Component-Based Security Under Partial Compromise

Martin Dehnel-Wild

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2018

Abstract

In our increasingly connected world, daily life depends on the correct and unimpaired operation of the computers controlling our infrastructure. These systems are increasingly complex, both within the computers themselves, and in the communications between them that create larger systems. This complexity increases both functionality and potential vulnerability. In turn, this makes ensuring the security of these critical systems simultaneously harder and more important than ever.

Ensuring that systems are not meaningfully vulnerable to attack requires a wide range of techniques. During design and implementation, the foremost of these is formal verification. Large, real-world systems and protocols made up of multiple computers or components are rarely designed with verification or analysis in mind. By the same token, these same systems have very often proved tricky to analyse formally.

Can we meaningfully verify the security properties of large, real-world, multi-component systems, in a reasonable amount of time? Is there value in making the modelling and analysis as fine-grained as we can, or does the extra effort fail to buy us anything? Can these systems still operate correctly and securely even when partially compromised by an attacker? Can we improve the languages and tools we use to model systems under partial compromise?

In this thesis, we address these questions through a mixture of theory and practice. We first consider a major power grid communications standard, DNP3. We show that its "Secure Authentication" protocols meet their security goals, and that a previously claimed attack is not possible. We then consider the security of 5G, and its main Authentication and Key-Agreement protocol. Our analysis reveals that 5G-AKA's security relies on unstated assumptions; in practice this means 'correct' implementations can be vulnerable to a security-critical race condition.

Our analyses show how far we have to go in terms of resilience under partial compromise. Neither studied system fares well against attackers which control any one of its components: this is no longer acceptable in elements of critical national infrastructure, as these systems increasingly come under sophisticated attack.

We then consider the formalisation of partial compromise, building the idea of an attacker controlling part of a system into models from the start. Our new techniques give approachable yet powerful ways to model a wide range of multi-component systems and protocols against fine-grained threat models.

We finish with two main conclusions. First, the time is now right for formal methods. We show that precise, fine-grained modelling of complex, multi-component protocols is both possible and valuable in the real world. Second, partial compromise resilience must be built in from the start. With little hope of securing all end-points completely, it is essential that resilience is also built into the network protocols.

Martin Dehnel-Wild, New College
A thesis submitted for the degree of Doctor of Philosophy, Trinity Term 2018

Acknowledgements

There are so many people without whom this thesis would never have happened. I will undoubtedly miss acknowledging many, but will try my best regardless.

First and foremost, Cas: thank you for your unending expertise, patience, knowledge, ideas, and enthusiasm. You have taught me so much about science, about how to be a researcher, and about how to get things done — in spite of my best efforts to the contrary. Your ever-cheerful attitude, combined with your mantras "For Science!" and "If you do good science, the rest will sort itself out" have helped me through many of the ups and downs this DPhil threw at me; I will hang on to them for a long time to come. Thank you for your belief in me, right from our very first meeting.

Thank you to all of the group — Katriel, Kevin, Luke, Dennis, and Nick — you've made the last four years an absolute blast, and so enjoyable. Thank you all for your willingness to discuss any and every crazy idea or stupid question I had, whether related to work or (more likely) not. And for all the board games.

Thank you to Professors Andrew Martin, David Gavaghan, and Andrew Simpson for your hands-off-but-supportive-and-encouraging roles of co-supervisor, college advisor, and patient receiver of rants respectively; knowing I could rely upon you to help me see the bigger picture when it all seemed impossible was invaluable.

Thank you to Chad and Beth Heitzenrater: for your wonderful friendship, for all the sanity-restoring conversations, whether over a beer, dinner, or a G&T, and for the great discussions about work, science, and the wider world.

Thank you to so many other friends made and/or continued through the department: I hope we all stay in touch for many years, but especially Drs. (or soon to be) Justin King-Lacroix, Andrew Paverd, Yang Liu, Max Whitby, David Mestel, Rich Baker, Matt Smith, and Jassim Happa.

Thank you especially to Julie Sheppard (and Lyn and Sarah) without whom I wouldn't have even *started* my DPhil, never mind completed it; long live the JCCG. Thank you to Suzanna Marsh and Dr. Rosanna Cretney for inspiring my interest in Outreach, for all the amazing work you do, and for being an oasis of peace and calm on so many occasions.

Thank you to Peter C, Tom K, Tom W, Adam B, and Catherine "High Speed Rail" H2 and all the others in the various academic teams for your support in many forms — but of course primarily for the funding and belief in the project. Thank you to Steve Babbage and Tim Evans at Vodafone Group for giving up your time and expertise to help me with the 5G project.

Thank you to all the friends I've made and re-discovered at Oxford outside of the department: for reminding me that there's life beyond my thesis. Especially thank you to Drs. David Bowe and Jennifer Thorp, Will Dawes and Dr. Katie Bank, Edward Smyth, Daniel

Tate, Edward Carroll, Tom Dixon, and all those from the various choirs of Worcester College (especially Alex, Ben, and Dan), The The Queen's College, New College, ChCh, and SMV. Thanks to Michael Peyton Jones, David Newman, Tim Muller, and Kati (again) for all the squash. Thank you to all friends from Canford, Durham, and beyond, but especially (in no particular order) to Pete, Pete, Pete, Tom, Tom, Leo, Karen, Cordie, Tash, Tasha, Matt, Farrah, Ella, Guy, Dan, Adam, James, Oli, Danny, Alan, Magnus, Sarah, Alice, Andrew, Alexander, Natalie, Kenneth, and Kim.

Thank you to my wonderful family: Mum, Alex, Alan, Margaret, Rob, Nicky, Jonny, Helen, and Simon — for all the encouragement in many forms, the dinners, the love, and friendship.

Finally, Heather. You have been an unwavering rock of support, love, and patient encouragement through all the time I've had the privilege of knowing you, but especially during the last four years. With all my heart, thank you.

Contents

List of Figures

"Lots of people working in cryptography have no deep concern with real application issues. They are trying to discover things clever enough to write papers about."
— Whitfield Diffie

1 Introduction

In our increasingly connected world, daily life depends on the correct and unimpaired operation of the computers controlling our infrastructure. These systems are increasingly complex, both within the computers themselves, and in the communications between them that create larger systems. This complexity increases both functionality and potential vulnerability. In turn, this makes ensuring the security of these critical systems simultaneously harder and more important than ever.

Ensuring that systems are not meaningfully vulnerable to attack requires a wide range of techniques. During design and implementation, the foremost of these is formal verification. Large, real-world systems and protocols made up of multiple computers or components are rarely designed with verification or analysis in mind. By the same token, these same systems have very often proved tricky to analyse formally. It is perhaps not hard to see why large parts of industry have avoided formal analysis tools and techniques, due to both their impracticality and limited perceived value. We do not pretend to fix this problem, but we do believe that this thesis pushes in the right direction.

As a result of slow-moving, legacy-influenced design, many elements of Critical National Infrastructure (CNI) such as power grids and mobile telephony networks end up becoming large, unwieldy, multi-component systems. Security for such overall systems, and trust in the communications between their constituent components is very often only considered long after the initial architecture is set in stone (or more literally, concrete). As a result, security mechanisms and protocols are frequently bolted on top, or crudely designed to mimic a system's structure, engaging and requiring input from many different elements. More modern designs un-encumbered by legacy requirements regularly implement pairwise security and authentication between elements, secured using well studied two-party asymmetric cryptography-based mechanisms such as TLS and IPsec. We recognise that for whatever reasons, changing existing real-world systems to reflect this is very often not possible.

Formal analysis of network security protocols has historically focussed on two-party protocols, and this field is relatively mature. There have been many successful studies and analyses of major protocols, achieving verification of desired properties in many cases, and finding (and fixing) attacks in others [34, 37, 66, 68, 73, 88]. Within academia it is now broadly expected that newly proposed network security protocols and similar constructions are presented with security analyses or proofs; this level of rigour is very rarely considered (much less required) for protocol standards within industry.

While we cannot say with any authority why this is the case, we believe that the ostensible requirements of vast expense, time, and expertise to achieve what are perceived to be relatively weak results (whether this view is justified or not) combined with legacy-based limitations on designs and modifications have severely hindered uptake of formal methods and successful collaborations between academia and industry in this realm. This leads to our first research question.

> **Question 1: Can we achieve meaningful verification results for large, real-world, multi-component systems in a reasonable time frame?**

We answer this question with a resounding 'Yes!', presenting the research in both Chapters 3 and 4 as our evidence, gaining meaningful results for large, real-world projects within sensible time frames. We use a range of component-based modelling and analysis techniques within the Tamarin Prover [138] to achieve these results.

Aside from addressing a research question, we hope that the work contained within this thesis will further strengthen the argument that the time is now right for formal analysis of modern, large-scale systems and protocols.

It is worth stating from that outset that we have chosen to use *symbolic* methods of formal verification for this research, rather than computational. Symbolic models consider data as terms, and any party (honest or malicious) can either know a full term, or nothing about it at all — they cannot know half a term, or just a single bit of a term. This is necessarily a more simplistic model of the data computed upon and transmitted in protocols than considering each individual bit involved. The benefit of making this choice is that it significantly reduces the state space searched, allowing us to achieve more readily the termination of (and useful results from) automated techniques and tooling for analysis.

We made this choice because we believe many protocols and systems within the CNI/ICS realm are more suited to symbolic analysis. These systems and protocols broadly follow the pattern of using simpler cryptographic primitives in a relatively well understood manner; the high level of complexity (and possible cause of vulnerabilities) of these protocols normally comes from the large number of cases and statefulness. This type of setup is therefore not suited to analysis by the computational, pen-and-paper approach, so in this thesis we focus on automated symbolic verification. We briefly discuss computational analysis techniques in Chapter 6.

In modelling and analysis of distributed systems it is often both possible and reasonable to combine morally similar and even geographically co-located elements or components of a system into a single end-point. All three areas of protocol design, formal modelling, and analysis are significantly more mature for two-party protocols than for multi-party protocols; given this disparity, it could reasonably make sense to amalgamate parties or components within protocols where possible. This would reduce the number of parties (ideally down to just two) over which the analysis has to reason, and thus limit the state space for any verification methods which use search algorithms. Pushing back against this, our second research question is as follows.

> **Question 2: Is there tangible value in fine-grained modelling and analysis of real-world, multi-component systems?**

We first begin to address this question in Chapter 3, by demonstrating that a previously claimed attack against a major power grid protocol suite is not possible: we conclude that the cause of this false positive was overly coarse modelling, which did not represent the specification sufficiently accurately.

We then take a slightly different approach, and consider multi-component protocols and systems where we could reasonably amalgamate potentially co-located components for ease of analysis. This could either be due to genuine historic amalgamation (and where the studied protocol is a slightly more complex successor to previous versions), or just similarity of roles and functionality within a protocol. There are then two natural paths: we can make two models, one with amalgamated components and one without; or, alternatively we could consider systems where others have already performed the analysis with some components combined from the outset. We happily encounter the latter scenario, and discuss the context, our modelling and results for this particular research in Chapter 4.

Much work has already considered different types of compromise in two-party protocols such as forward secrecy (compromising long-term keys *after* a 'secure' session), post-compromise security (compromising long-term keys *before* a 'secure' session), session-state revelations, and the resultant implications for the security properties each party can achieve [34, 62, 66, 127, 139]. In the two-party context, entirely compromising one of a protocol's two actors is understandably not considered 'partial compromise', so instead, compromise actions are variously limited to single keys (often with temporal restrictions), session state data compromise, and even access-based compromise, such as allowing an adversary illegitimate access to an oracle for (but not possession of) a key, using e.g., a hardware security module or trusted platform module. Many modern two-party protocols have been designed in such a manner as to be demonstrably resilient to these types of compromise.

Far less research has considered real-world, multi-component systems (distributed or discrete) under partial compromise. In this context we posit that the 'partial' in partial compromise can more readily refer to an entire component, rather than just a single term or key. Recognising that components within the systems we consider are distinctly heterogeneous in functionality, we expect that compromise of any one individual component could have wildly varying implications depending on the compromise in question.

Following on from component-based modelling with implicitly honest components (but with vulnerable or adversarially-controlled network channels), we therefore consider what happens when individual components within multi-component systems can be compromised by an attacker.

Question 3: Are real-world, component-based systems resilient to partial compromise?

We answer this again through considering our two major systems in Chapters 3 and 4. After analysing their resultant security properties under their stated threat models (i.e., without partial compromise), we take another look at these properties under varying levels of partial compromise.

The first of these (DNP3's Secure Authentication v5 protocol suite in Chapter 3) falls at the first hurdle. While in our models it achieves all of its stated security goals under the precise threat model described by the standard, compromising any one of the system's three major components completely destroys all properties of authentication, secrecy, and replay protection.

Secondly, we consider the security of 5G's main Authentication and Key Agreement protocol (5G-AKA) in Chapter 4, which leverages a range of assumptions on both the security of various trusted components, as well as internal network channel properties. Compromising these components and channels in a variety of ways is not as completely devastating as with Chapter 3, but the results are still not pretty. In each case, we provide recommendations for how to lessen the impact of partial compromise, but as these broadly require major changes to the primitives and protocols used, we recognise that these specific changes are fairly unlikely to be adopted soon.

Finally, having explored a range of security properties and threat models for multi-component systems in a semi-structured but broadly case-study specific manner, we explore how to improve upon this. We would ideally like to consider analysis of systems under partial compromise in a more systematic and rigorous manner, creating the theoretical tools and framework necessary to achieve this.

Question 4: Can we provide a formal framework to analyse partial compromise systematically?

We provide the first formal framework for modelling and analysis of multi-component systems under partial compromise in Chapter 5. This builds significantly upon the execution framework provided by Cremers and Mauw in [74], newly giving the modeller the built-in ability to specify fine-grained partial compromise abilities and restrictions for both components and network channels within multi-component systems. We make significant progress here in providing a working operational semantics, as well as giving multiple examples in these semantics. However, as described initially, and discovered in previous chapters (especially Chapter 3), real-world protocols involve many complicating factors such as unbounded looping and significant reliance on statefulness, for which these specific semantics are not yet mature enough. This naturally invites future work to augment our presented framework and address these limitations.

Lots of research considers protocols when the network channels are adversarially controlled; large amounts of separate research plays whack-a-mole with preventing compromise of the protocols' end-points themselves. In this thesis, we focus on some of the consequences of interaction between these two areas.

1.1 Contributions

In addressing these research questions and writing this thesis, we make contributions encompassing both theory and practice. These provide both specific project-based results, recommendations, and impact which stand on their own, as well as higher level observations and answers to our posed research questions.

In Chapter 3 we formally model and analyse DNP3's Secure Authentication v5 protocol suite (IEEE 1815-2012 [110]), and conclude that it meets its claimed security goals. We provide the most comprehensive analysis of the full DNP3 Secure Authentication v5 protocol yet, leveraging automated tools for the symbolic analysis of security protocols. In particular:

- We provide the first formal models of two of the SAv5 sub-protocols that had not been modelled previously. We additionally provide the first models of both the symmetric and asymmetric modes of the highest level key transport sub-protocol.
- We provide the first analysis of the complex combination of the three sub-protocols, thereby considering cross-protocol attacks as well as attacks on any of the sub-protocols. The security properties that we model capture the standard's intended goals in much greater detail than previous works.
- Despite the complexity of the security properties and the protocol, and in particular its complex state-machine and key updating mechanisms, and considering unbounded sessions and loop iterations, we manage to verify the protocol using the Tamarin prover. We conclude that the standard meets its intended goals if implemented correctly, increasing confidence in this security-critical building block of many power grids (Question 1).
- Notably, our findings contradict a claimed result by an earlier analysis; in particular, our findings show that an attack claimed by other work is not possible in the standard as defined. We believe this incorrectly claimed attack was the result of overly simplistic modelling (Question 2).
- We demonstrate that when under partial compromise (outside of its allowed threat model), the protocol does not uphold any of its security requirements (Question 3).
- Our analysis naturally leads to a number of recommendations for improving future versions of the standard.

This research has been accepted for publication at the European Symposium on Research in Computer Security (ESORICS) 2017 [70], and won the "Best Paper Award". The significantly extended journal version of this research has been accepted for publication in the Journal of Computer Security.

In Chapter 4, we formally model and analyse the 5G-AKA protocol from within the 5th Generation mobile telephony standards, created by 3GPP. We discover a potential vulnerability within the modelled standard, and have worked with 3GPP to ensure that the vulnerability is mitigated and corrected.

- First, we propose a fine-grained formal model of the 5G-AKA standard that enables a detailed view of the interaction between the various security-critical components. This models all four major components described by the standard, the standard's desired security properties, and the allowed threat model.

- Second, we perform symbolic analysis of this model with respect to a range of threat models (Question 1). Our analysis confirms many already discovered issues and subtle assumptions and requirements to achieve security in 5G-AKA.

- Third, our analysis reveals that the security of 5G-AKA critically relies on unstated assumptions on the inner workings of the underlying channels. In particular, the automated analysis discovers an attack that exploits a potential race condition. We additionally show that solving the race condition for the honest case does not necessarily prevent the attack. In practice this means that solely based on the 5G-AKA specification, a provider can implement the standard insecurely. We propose fixes and prove that they prevent the attack (Question 1). This race condition lay un-discovered in simpler, three-party models. We believe this demonstrates the value of fine-grained component-based modelling and analysis (Question 2).

- Fourth, we provide the first analysis of the 5G-AKA protocol under partial compromise (Question 3): we give the adversary the ability to perform varying levels of compromise on individual network components, as well as the previously secure internal network channels, and then explore and discuss the results and their implications for 5G.

  We have reported our findings to the 3GPP SA3 working group and are currently working with Vodafone to integrate a fix to the standard.

A conference paper version of this chapter has been accepted for publication at the Network and Distributed Systems Security Symposium (NDSS) 2019.

In Chapter 5 we consider and build upon some of the lessons learned from the analysis of real-world systems, and provide a new operational semantics for component-based systems under partial compromise, addressing Question 4. This more rigorous framework and approach to modelling and analysing partial compromise builds significantly upon the network-focussed operational semantics of Cremers and Mauw [74].

- This framework incorporates novel and fine-grained adversarial compromise capabilities allowing for exceptionally easy control over specific component and network-channel compromises. It additionally introduces significantly more expressive and precise control over the allowed threat models than previously seen within this type of framework. Together, we believe this provides a solid theoretical foundation for modelling and analysing component-based systems under partial compromise.

- We demonstrate the utility of this framework by modelling a system representing a simplistic ATM (cash machine) under partial compromise. We show two different component configurations and protocols, and their resultant security and insecurity under different threat models. We demonstrate basic analysis in this framework by providing a manual proof that one of these systems meets its security goals under a specified threat model.

1.2 Statement of originality

All work is my own (supervised by Professor Cas Cremers) except Chapter 2, describing necessary background material on Tamarin and symbolic modelling, and Chapter 3, modelling and analysing IEEE 1815–2012 (DNP3: SAv5).

Chapter 2 (Background) is drawn almost completely from other sources, mostly the theses of Milner [140], Meier [137], and Schmidt [159], as well as content from the Tamarin Manual [35] and associated papers [89, 138, 160]. This is explicitly **not** claimed as a contribution, but instead included for completeness.

Chapter 3 (DNP3: SAv5) is joint work with Cas Cremers and Kevin Milner; I proposed and led this project from start to finish. Kevin Milner completely re-factored the final Tamarin models, which significantly helped with termination issues. These were based upon my original models, and among other things, the re-factoring manually identified invariants over unbounded loops to achieve termination. Kevin wrote the section "Analysis in Tamarin" (Section 3.4.2) describing this; the remainder is my work.

Chapter 5 is my own work, but it builds significantly upon the starting point of the execution model and network-focussed operational semantics provided by Cremers and Mauw in [74]. The structure of the framework and basic execution model is similar to Cremers and Mauw's, but the semantics as a whole have been significantly extended and modified to incorporate partial compromise, fine-grained adversarial control, and the associated threat modelling.

1.3 Overview

We start by introducing relevant background material in Chapter 2; this covers notational preliminaries, the Tamarin prover, and the symbolic modelling assumptions needed to understand later chapters. We then consider our first major real-world, component-based system, the Distributed Network Protocol 3: Secure Authentication v5 protocol suite in Chapter 3. We explore both the protocol's rules and desired security properties, before modelling and analysing this protocol under the standard's precise threat model, and then under partial compromise.

We next look at 5G in Chapter 4: we consider the mobile telephony ecosystem's main authentication and key-agreement protocol, 5G-AKA, considering its message rules, precise threat model, and desired security goals. We then model and analyse the protocol, including all four major components specified by the standard. This analysis uncovers potential issues with the specification, and we work with the standards body who wrote the specifications (3GPP) to ensure these issues are correctly mitigated or fixed. We then consider how this large, multi-component system behaves under partial compromise, and which of its desired security properties it still upholds under a variety of threat models.

Then, having analysed two major, real-world systems under partial compromise, we explore how we can make this type of modelling and analysis more rigorous and systematic in Chapter 5. We provide the first operational semantics for component-based systems under partial compromise, and give examples throughout, demonstrating how it can be used to model and analyse protocols.

We consider related work, discussing a range of systems under varying levels of compromise, and how historic approaches have tried to tackle this problem in Chapter 6, as well as providing discussions of earlier results relating to the security of DNP3 and 5G. Finally, we conclude in Chapter 7 with a discussion of our results, and next steps for future research.

"If someone steals my laptop while I'm logged in, they can read my email, take my money, and impersonate me to my friends. But at least they can't install drivers without my permission."

— Randall Munroe

2 Background

This chapter is not a contribution, but instead included to give the reader sufficient information and background knowledge to understand the modelling and analysis described in later chapters more fully. We draw heavily and directly on the descriptions in [89, 138, 160], the theses of Meier [137], Schmidt [159] and especially Milner [140], the descriptions from the Tamarin Prover Manual [35] (Licensed under Creative Commons: Attribution-NonCommercial-ShareAlike 4.0 International License), and the notational preliminaries broadly follow the conventions in Cremers and Mauw [74]. All content in this chapter is drawn directly or indirectly from these sources (often quoting verbatim) unless stated otherwise; for more complete descriptions, proofs, and illuminating examples, we refer the reader to these sources.

2.1 Notational preliminaries

Let f be a function. We use $dom(f)$ and $ran(f)$ to denote the domain and range of f. We write $f[a \mapsto b]$ to denote f's update, i.e., the function f' such that $f'(x) = b$ when $x = a$, and $f'(x) = f(x)$ otherwise. We denote a partial function from X to Y by $f : X \nrightarrow Y$. For a set S, we denote the power set of S by $\mathcal{P}(S)$, and we denote the set of finite sequences of elements from S by S^*. Sequences of elements or terms t_0 to t_n are written $\langle t_0, \ldots, t_n \rangle$, and brackets are omitted where no confusion can occur. For s, a sequence of length $|s|$, and for $i < |s|$, we say that s_i indicates the i^{th} element of s, starting at $i = 0$. We write s^s' to denote the concatenation of the sequences s and s'. We abuse set-notation to write $e \in s \iff \exists i. s_i = e$.

When discussing multisets we reuse familiar set notation annotated with $\sharp$. The empty multiset is denoted by ${}^\sharp\emptyset$, and we annotate the usual set operations with $\sharp$ (on the left hand side) to denote the relevant multiset operations. For example, we use $^\sharp\backslash$ to refer to the multiset extension of the $\backslash$ operation, $^\sharp\{a, b, b\}$ for the multiset containing a once and b twice, etc. To refer to the set of elements of a multiset M, we write $\mathit{set}(M)$. We write $^\sharp\{s_1, \ldots, s_n\}$ for the multiset containing s_1 to s_n. $^\sharp S$ denotes the set of finite multisets with elements from S, i.e.,

$$^\sharp S \overset{\text{def}}{=} \left\{ ^\sharp\{s_1, \ldots, s_n\} \ \mid \ s_1, \ldots, s_n \in S \right\}$$

Let $\mathcal{S}ub$ be a set of substitutions; we then write $[t_0, \ldots, t_n / x_0, \ldots, x_n] \in \mathcal{S}ub$ to denote the substitution of t_i for x_i, for $0 \leq i \leq n$. We extend the functions *dom* and *ran* to substitutions. We denote the union of two substitutions by $\sigma \cup \sigma'$, which is defined when $\mathit{dom}(\sigma) \cap \mathit{dom}(\sigma') = \emptyset$, and write $\sigma(t)$ for the application of the substitution σ to t, omitting the brackets where no confusion can occur. For a binary relation R, we denote its reflexive transitive closure by R^*.

2.2 The Tamarin Prover

The TAMARIN prover is a tool for the symbolic modeling and analysis of security protocols [138]. It takes as input a security protocol model, specifying the actions taken by agents running the protocol in different roles (e.g., the protocol initiator, the responder, and the trusted key server), a specification of the adversary, and a specification of the protocol's desired properties. TAMARIN can then be used to automatically construct a proof that, even when arbitrarily many instances of the protocol's roles are interleaved in parallel, together with the actions of the adversary, the protocol fulfils its specified properties.

TAMARIN provides general support for modeling and reasoning about security protocols. Protocols and adversaries are specified using an expressive language based on multiset rewriting rules. These rules define a labeled transition system whose state consists of a symbolic representation of the adversary's knowledge, the messages on the network, information about freshly generated values, and the protocol's state. The adversary and the protocol interact by updating network messages and generating new messages. TAMARIN also supports the equational specification of some cryptographic operators, such as Diffie-Hellman exponentiation and bilinear pairings. Security properties are modeled as trace properties, checked against the traces of the transition system, or in terms of the observational equivalence of two transition systems.

A formal treatment of TAMARIN's foundations is given in the theses of Schmidt [159] and Meier [137]; we give an overview of it here. For an equational theory E defining cryptographic operators, a multiset rewriting system R defining a protocol, and a formula φ defining a trace property, TAMARIN can either check the validity or the satisfiability of φ for the traces of R modulo E. As usual, validity checking is reduced to checking the satisfiability of the negated formula. Here, constraint solving is used to perform an exhaustive, symbolic search for executions with satisfying traces. The states of the search are constraint systems. For example, a constraint can express that some multiset rewriting step occurs in an execution or that one step occurs before another step. We can also directly use formulas as constraints to express that some behavior does not occur in an execution. Applications of constraint reduction rules, such as simplifications or case distinctions, correspond to the incremental construction of a satisfying trace. If no further rules can be applied and no satisfying trace was found, then no satisfying trace exists. For symbolic reasoning, TAMARIN exploits the finite variant property [63] to reduce reasoning modulo E with respect to R to reasoning modulo AC with respect to the variants of R.

2.2.1 Term rewriting

In TAMARIN, cryptographic messages and operations are abstracted as terms with rewrite rules. This approach derives many of its methods from the more generic notion of term rewriting, and here we briefly recall some standard notation used (as in, e.g. [92]) and how it applies to TAMARIN.

Order sorted term algebras An order-sorted signature $\boldsymbol{\Sigma} := (S, \leq, \Sigma)$ consists of a set of sorts S, a partial order $\leq$ on S, and a set of function symbols Σ associated with sorts such that the following two properties are satisfied. First, for every $s \in S$, the connected component C of s in $(S, \leq)$ has a top sort denoted $top(s)$ such that $c \leq top(s)$ for all $c \in C$. Second, for every $f : s_1 \times \ldots \times s_k \rightarrow s$ in Σ with $k \geq 1$, $f : top(s_1) \times \ldots \times top(s_k) \rightarrow top(s)$ is in Σ. We assume that there are pairwise disjoint, countably infinite sets of variables $\mathcal{V}_s$ and constants C_s for each sort $s \in S$. We define the set of all variables as $\mathcal{V} := \bigcup_{s \in S} \mathcal{V}_s$ and the set of all constants as $C := \bigcup_{s \in S} C_s$. We use $x : s$ to denote variables from $\mathcal{V}_s$. For an arbitrary set $A \subseteq \mathcal{V} \cup C$ of variables and constants, $\mathcal{T}_\Sigma(A)$ denotes the set of well-sorted terms constructed over $\Sigma \cup A$. If there is only one sort in S, we say that $\boldsymbol{\Sigma}$ is unsorted and we identify $\boldsymbol{\Sigma}$ with its set of function symbols Σ. We write Σ^k for the set of all k-ary function symbols in Σ.

In TAMARIN, cryptographic messages are modelled using an order-sorted term algebra comprising a top sort *msg* and two mutually incomparable sorts *fr* and *pub* such that both $fr \leq msg$ and $pub \leq msg$. The fresh sort *fr* is used for terms such as nonces and random numbers that are freshly generated, while the public sort *pub* is used to model terms like message tags that are known to all participants (including the adversary).

In implementation, the sort of a variable is expressed using prefixes:

- `~x` denotes $x : fr$
- `$x` denotes $x : pub$
- `#i` denotes $i : temp$ (discussed in 2.2.4)
- `m` denotes $m : msg$

A string constant `'c'` denotes a public name in *pub*, which is therefore a fixed, global constant known to all, including the adversary.

Terms and subterms A term is constructed of subterms with positions, where a position is a sequence of natural numbers representing the path taken to reach the subterm when viewing the term as a tree.

A position p in a term t is a finite sequence of integers, the empty sequence being denoted by $[]$, and we write $t|_p$ for the subterm of t at position p, where

1. if $p = []$, then $t|_p = t$,
2. if $p = [i]\hat{}p'$, and $t = f(t_1, \ldots, t_n)$ for $f \in \Sigma$ and $1 \leq i \leq n$ then $t|_p = t_i|_{p'}$, and
3. otherwise $t|_p$ is not defined and p is not a valid position.

We define a partial ordering relation between two positions p and q such that $p \leq q$ if $\exists p'.(q = p\hat{}p')$, i.e., p is a prefix of q (note that p' may be an empty sequence). Positions p and q are called incomparable if neither $p \leq q$ nor $q \leq p$. For a term t, we write $t[u]_p$ for the term t where $t|_p$ is replaced by u.

Subterms t' of a term t are written $t' \sqsubseteq t$, e.g., $t_1 \sqsubseteq \langle t_1, t_2 \rangle$, and $t_2 \sqsubseteq \langle t_1, \langle t_2, t_3 \rangle \rangle$. For a term t, the set $Subterms(t)$ is the set of all subterms of t. We define $Var(t) = Subterms(t) \cap \mathcal{V}$, the set of all variables in t. A term t is said to be ground if $Var(t) = \emptyset$.

Equational theory An equation over an order-sorted signature Σ is written $s \simeq t$ for terms $s, t \in \mathcal{T}_\Sigma(\mathcal{V})$. A signature Σ and a set of equations E over that signature induce a congruence relation $=_{\Sigma,E}$ over the terms $\mathcal{T}_\Sigma(\mathcal{V})$, which we call an equational theory. We identify the equational theory $=_{\Sigma,E}$ with the set of equations E when the signature is clear from context.

In TAMARIN, cryptographic operations are modelled through the choice of an arbitrary signature Σ and a restricted equational theory E which formalises the semantics of operations. For example, TAMARIN's built-in symmetric encryption corresponds to $\Sigma = \{senc(\cdot,\cdot), sdec(\cdot,\cdot)\}$ for the function signature, and $E = \{sdec(senc(m,k),k) \simeq m\}$ for the equational theory. TAMARIN includes built-in function symbols with corresponding equational theories for several common cryptographic operations which can be found in [35]. These include hashing, symmetric- and asymmetric-encryption, signing, message-revealing signature schemes, Diffie-Hellman groups, bilinear pairing, and multisets.

Substitutions and unification A substitution σ is a map from some set of variables $dom(\sigma) \subseteq \mathcal{V}$ to terms $range(\sigma) \subseteq \mathcal{T}_\Sigma(\mathcal{V})$ such that $x : s \in dom(\sigma)$ implies $\sigma(x) : s \in range(\sigma)_s$. By convention, we write $t\sigma$ for a homomorphic extension of σ applied to the term t, and assume that substitutions are idempotent, such that $(t\sigma)\sigma = t\sigma$. A substitution is called a renaming if there exists an inverse (idempotent) substitution σ^{-1} such that $\forall t \in dom(\sigma) . (t\sigma)\sigma^{-1} = t$. A unifier of two terms s and t over a relation $=_E$ is a substitution σ such that $s\sigma =_E t\sigma$.

Rewriting modulo A rewrite rule over an order-sorted signature Σ is written $l \to r$ for terms $l, r \in \mathcal{T}_\Sigma(\mathcal{V})$. A rewrite system R is a set of rewrite rules defining a relation $\to_R$, such that $s \to_R t$ if there is a rule $l \to_R \in R$, a substitution σ, and a position p in s such that $s_p = l\sigma$ and $s[r\sigma]_p = t$. A rewrite system R modulo an equational theory E is defined similarly by a relation $\to_{R,E}$ where instead $s_p =_E l\sigma$ and $s[r\sigma]_p =_E t$. Tamarin decomposes user-defined equational theories into a rewrite system, which is only possible for a restricted class of equational theories. We do not go into detail about these restrictions or how they help with verification here, as they are not relevant to our work. For detailed discussion see the original description in [160] and recent work relaxing these restrictions in [89].

2.2.2 Multiset rewriting in Tamarin

Tamarin uses labeled multiset rewriting to model security protocols that communicate over a public channel controlled by a Dolev-Yao style adversary [86]. Such an adversary can see and block every message sent on the channel they control. They can furthermore send any message deducible from the messages that they have seen on this channel.

We briefly recall some definitions from [137, 159] specifying facts and rules, how they define a trace set, and how trace properties are specified in Tamarin. We assume there are two countably infinite sets FN and PN of fresh and public names respectively, which we will use for assigning names to terms of sort fr and pub in evaluating trace properties. We write $\mathcal{T}$ to refer to $\mathcal{T}_\Sigma(FN \cup PN \cup \mathcal{V})$, and $\mathcal{M}$ for $\mathcal{T}_\Sigma(FN \cup PN)$, where Σ will be clear from context.

Facts A fact is a tag drawn from finite signature $\Sigma_\mathcal{F}$ and a sequence of terms from $\mathcal{T}$. Fact tags each contain a name, an arity, and a multiplicity. The multiplicity of a fact may be either linear, in which case it is removed from state when consumed by a rewrite rule, or persistent, remaining in state even if consumed by a rule. Linear facts are used to model, for example, limited resources and mutable state; persistent facts are used to model things like read-only memory, adversary knowledge, and causal relationships. A linear fact Fa/k containing terms $\langle t_1, t_2, \ldots, t_k \rangle$ is written $\mathsf{Fa}(t_1, t_2, \ldots, t_k)$, while persistent facts are denoted in Tamarin's syntax by prefixing their name with an exclamation mark, e.g., $!\mathsf{Fa}(t_1, t_2, \ldots, t_k)$. We use $\mathcal{F}$ to refer to the set of all facts, and $\mathcal{G}$ to refer to all ground facts (i.e., all terms are ground).

Certain fact tags are given special meaning by TAMARIN:

- `Fr/1` to model the generation of fresh names,
- `In/1` to model receiving from a Dolev-Yao network, and is produced only by adversary construction rules,
- `Out/1` to model sending to a Dolev-Yao network, as a premise to adversary deconstruction rules, and
- `!K/1` to model adversary knowledge.

Facts can be intuitively seen as predicates storing information about the state given by their arguments. In implementation, facts must start with a capital letter, and all facts of the same name must have the same arity: e.g., `!Ltk(A, ltkA)` and `!Ltk(B, ltkB)` is acceptable in the same model, but `!Ltk(A, ltkA)` and `!Ltk(ltk)` is not.

Rewriting rules A labelled multiset rewriting rule is defined by three fact sequences: its premises (or left-hand side) l, its actions a, and its conclusions (or right-hand side) r, together written $[\ l\ ] - [\ a\ ] \rightarrow [\ r\ ]$. For legibility when l, a, and r are longer sequences of facts, we may write $[l_1, l_2, \ldots, l_{|l|}] - [a_1, a_2 \ldots, a_{|a|}] \rightarrow [r_1, r_2, \ldots, r_{|r|}]$ equivalently as

$$\frac{l_1, l_2, \ldots, l_{|l|}}{r_1, r_2, \ldots, r_{|r|}}[a_1, a_2, \ldots, a_{|a|}]$$

following the notation for inference rules. In TAMARIN, each rule also has some associated rule information, including annotations like a rule name that do not affect the semantics of the rule.

The term ‘labelled’ in labelled multiset rewriting refers to labelling of rules with actions. The actions of a rule are sequences of facts that are used to express execution properties; we will discuss this further in Section 2.2.4. All multiset rewriting rules in TAMARIN are labelled (though the actions may be an empty sequence), so we generally drop the ‘labelled’ qualifier and refer to them just as multiset rewriting rules. Finally, given a rule ru we use $prems(ru)$, $acts(ru)$, and $concs(ru)$ to refer to the rule's premises, actions, and conclusions respectively.

Multiset rewriting rules in TAMARIN are divided into two sets: protocol rules and message deduction rules. Protocol rules are used to specify both the behaviour of honest participants as well as model-defined adversary capabilities like key reveal. Protocol rules in TAMARIN must meet six conditions (from [159]).

A protocol rule is a multiset rewriting rule $[\ l\ ] - [\ a\ ] \rightarrow [\ r\ ]$ such that:

P1. l, a, and r do not contain fresh names,

P2. l does not contain `!K` and `Out` facts,

P3. r does not contain `!K`, `In` and `Fr` facts,

P4. The argument of a `Fr`-fact is always of sort fr

P5. r does not contain the function symbol $*$, and

P6. $[l] - [a] \rightarrow [r]$ satisfies both
   a) $vars(r) \subseteq vars(l) \cup \mathcal{V}_{pub}$ and
   b) l only contains irreducible function symbols from $\Sigma \setminus \Sigma_{\mathcal{DH}}$ (or it is an instance of a rule that does).

Condition P1 ensures that fresh names are not created directly by protocol rules; instead, fresh names are accessed through the `Fr` fact generated by a special rule. P2 and P3 ensure that special facts occur only in their intended places. P4 is not strictly necessary, as `Fr` facts can only be generated in a particular way and always contain a fresh name, but simplifies later definitions. Condition P5 restricts Diffie-Hellman product terms and is not relevant to our work, but included for completeness. Finally, P6 prevents rules from referencing non-public names which do not exist in their premises.

Message deduction rules determine how terms may be manipulated by the adversary to create `In` facts. The semantics of the fact symbols `In(m)`, `Out(m)`, and `K(m)` is given by the following set of message deduction rules:

$$\begin{aligned}
\mathit{pub}: &\quad [\,] - [\,] \rightarrow [\texttt{!K}(x:\mathit{pub})]\\
\mathit{gen\_fresh}: &\quad [\texttt{Fr}(x:\mathit{fr})] - [\,] \rightarrow [\texttt{!K}(x:\mathit{fr})]\\
\mathit{cf}: &\quad [\texttt{!K}(x_1)\ldots\texttt{!K}(x_k)] - [\,] \rightarrow [\texttt{!K}(f(x_1,\ldots,x_k))] \text{ for all } f \in \Sigma\\
\mathit{isend}: &\quad [\texttt{!K}(x)] - [\texttt{K}(x)] \rightarrow [\texttt{In}(x)]\\
\mathit{irecv}: &\quad [\texttt{Out}(x)] - [\,] \rightarrow [\texttt{!K}(x)]
\end{aligned}$$

The adversary knowledge is represented by `!K()` facts. The rules *pub* and *gen_fresh* model the ability of the adversary to learn any public name and generate their own fresh names respectively. The *cf* rule allows an adversary to combine terms by applying functions. Finally, *isend* and *irecv* allow the adversary to receive and send messages respectively. Note that *isend* is also labelled with $\texttt{K}(x)$ so that messages sent by the adversary appear in the protocol trace.

Example 2.2.1. We give a simplified example protocol rule, the initiator's first message from the Needham-Schroeder-Lowe protocol [130].

```
rule I_1:
  let m1 = aenc{'1', ~ni, $I}pkR
  in
    [ Fr(~ni)
    , !Pk($R, pkR) ]
  --[ ]->
    [ Out( m1 )
    , St_I_1($I, $R, ~ni) ]
```

In this rule, the initiator generates a fresh nonce `~ni` (`ni` : *fr*), and receives the public key of the responder with public ID `$R`, (`R` : *pub*) from a server in the persistent fact `!Pk($R, pkR)`. The initiator then outputs the term `m1` to the network, which is defined to be the asymmetric encryption of the string `'1'`, the nonce `~ni`, and the initiator's public identity `$I`. These terms are asymmetrically encrypted with the responder's public key, `pkR`. The rule additionally outputs the private fact `St_I_1($I, $R,~ni)`; intuitively, we use this fact to store the internal state of the initiator after it has executed the first step. This fact will then be consumed by the initiator in a later rule. The 'let' binding allows a modeller to specify a short representation for a term that will be written several times in the rule.

We omit the actions in the above rule as they are only relevant in the context of lemmas, which we will discuss in Section 2.2.4.

In addition to the basic message deduction rules, TAMARIN currently implements several extensions to these rules, including Diffie-Hellman rules as described in [138] and rules generated to support a larger set of equational theories as described in [160], but these are not relevant for our research.

For a set of rules R, we write $insts(R)$ and $ginsts(R)$ for the set of instances and ground instances of R respectively. To implement fresh name generation, we define a special rule which is the unique source of `Fr` facts, $\mathtt{Fresh} = [\,] - [\,] \rightarrow [\mathtt{Fr}(x : fr)]$.

2.2.3 Execution

The state of the transition system is modelled as a finite multiset of facts, which is modified step by step by executing rules.

A valid step of a rewrite system R with respect to an equational theory E is defined by the transition relation $\Rightarrow_{R,E}$:

$$\frac{l - [\,a\,] \rightarrow r \in_E ginsts(R \cup \{\mathtt{Fresh}\}) \quad lfacts(l) \subseteq^{\sharp} S \quad pfacts(l) \subseteq set(S)}{S \xRightarrow{set(a)}_{R,E} ((S \setminus^{\sharp} lfacts(l)) \cup^{\sharp} r}$$

where $lfacts(l)$ and $pfacts(l)$ respectively denote the linear and persistent facts in l. This transition can apply ground instances of rules if their premises are included in the state S, consumes the linear facts in the premises, and adds the rule conclusions to the state.

From this we define the set of all traces generated by the multiset rewriting system R modulo an equational theory E in terms of the labels appearing in rules.

$$traces_E(R) := \Big\{ \langle A_1, \ldots, A_n \rangle \;\Big|\; \exists S_1, \ldots, S_n \,.\, \varnothing \xRightarrow{A_1}_{R,E} S_1 \xRightarrow{A_2}_{R,E} \ldots \xRightarrow{A_n}_{R,E} S_n$$
$$\wedge \forall i, j, x \,.\, (i \neq j) \wedge (S_{i+1} \setminus^{\sharp} S_i) = {}^{\sharp}\{\mathtt{Fr}(x)\} \implies$$
$$(S_{j+1} \setminus^{\sharp} S_j) \neq {}^{\sharp}\{\mathtt{Fr}(x)\} \Big\}$$

The final condition enforces that each instance of Fresh is used at most once in a trace, thereby ensuring that each fresh name is unique.

Note that the sequence of rule instances in an execution is sufficient to characterize the execution, as both the trace and the sequence of global states can be reconstructed from it.

2.2.4 Trace properties

The Tamarin multiset rewriting rules define a labeled transition system. The system's state is a multiset of facts and the initial system state is the empty multiset. The rules define how the system can make a transition to a new state. Here we focus on the action facts, which are used to reason about a protocol's behaviour.

A rule can be applied to a state if it can be instantiated such that its left hand side is contained in the current state. In this case, the left-hand side facts are removed from the state, and replaced by the instantiated right hand side. The application of the rule is recorded in the trace by appending the instantiated action facts to the trace.

A trace property is therefore a set of traces. We define a set of traces in Tamarin using first-order logic formulas over action facts and timepoints. More precisely, Tamarin's property specification language is a guarded fragment of a many-sorted first-order logic with a sort for timepoints. This logic supports quantification over both messages and timepoints.

Tamarin makes use of a first-order logic with sorts to specify security properties. Temporal properties are supported through a sort *temp* for timepoints, with variable drawn from $\mathcal{V}_{temp}$; this logic supports quanitification over both messages and timepoints. Properties are specified as first-order formulas over trace atoms, comprising either:

1. falsehood, $\perp$, or
2. term equality, e.g., $t_1 \approx t_2$,
3. timepoint ordering, e.g., $i <$ j,
4. timepoint equality, e.g., $i \doteq j$,
5. and actions f @ i.

A trace formula is therefore a first-order formula over trace atoms. In order to evaluate trace formulas, we associate a valuation to variables. To ensure valuations respect sorts, we associate a valid domain D_s to each sort s : $\mathrm{D}_{temp} \subseteq \mathbb{Q}$, $\mathrm{D}_{fr} \subseteq FN$, $\mathrm{D}_{pub} \subseteq PN$, and $\mathrm{D}_{msg} \subseteq M$. A valuation itself is a function θ from $\mathcal{V}$ to $\mathbb{Q} \cup \mathcal{M}$ such that $\theta(\mathcal{V}_s) \subseteq \mathrm{D}_s$ for every sort, i.e., it respects these domains. As with substitutions, we write $t\theta$ for the homomorphic extension of θ applied to a term t. For a sequence s we write $idx(s)$ for the set of all indices of s. For an equational theory E, a satisfaction relation $(tr, \theta) \models_E \varphi$ is defined, relating a trace *tr* and a valuation θ with a trace formula φ:

$$
\begin{array}{ll}
(tr,\theta) \models_E \bot & \text{never} \\
(tr,\theta) \models_E f@i & \text{iff } \theta(i) \in idx(tr) \wedge f\theta \in_E tr_{\theta(i)} \\
(tr,\theta) \models_E i \lessdot j & \text{iff } \theta(i) < \theta(j) \\
(tr,\theta) \models_E i \doteq j & \text{iff } \theta(i) = \theta(j) \\
(tr,\theta) \models_E t_1 \approx t_2 & \text{iff } t_1\theta =_E t_2\theta \\
(tr,\theta) \models_E \neg\varphi & \text{iff } \neg((tr,\theta) \models_E \varphi) \\
(tr,\theta) \models_E \varphi \wedge \psi & \text{iff } ((tr,\theta) \models_E \varphi) \wedge ((tr,\theta) \models_E \psi) \\
(tr,\theta) \models_E \exists x : s \,.\, \varphi & \text{iff } \exists u \in \mathrm{D}_s \,.\, (tr,\theta[x \mapsto u]) \models_E \psi
\end{array}
$$

We extend this relation over trace sets and say that a trace formula φ is valid for a trace set Tr modulo E if and only if $(tr,\theta) \models_E \varphi$ for every trace $tr \in Tr$ and every valuation θ, and that a trace formula φ is satisfiable for a trace set Tr modulo E if and only if there exists a trace $tr \in Tr$ and valuation θ such that $(tr,\theta) \models_E \varphi$. We write $Tr \models_E^{\forall} \varphi$ and $Tr \models_E^{\exists} \varphi$ for validity and satisfiability respectively. Importantly, $Tr \models_E^{\forall} \varphi$ if and only if $\neg(Tr \models_E^{\exists} \neg\varphi)$, which allows us to convert from solving validity of a trace property to solving satisfiability of the negated property.

In most cases, we are interested whether a trace formula φ is valid (satisfiable) for all (some) traces of a multiset rewriting system. Overloading notation, we thus define that φ is R,E-valid, written $R \models_E^{\forall} \varphi$, iff φ is valid for $traces_E(R)$ modulo E. We define that φ is R,E-satisfiable, written $R \models_E^{\exists} \varphi$, iff φ is satisfiable for $traces_E(R)$ modulo E. When analyzing security protocols, we use satisfiability claims to formalize that there exists at least one execution of a protocol satisfying certain properties, e.g., that there exists an execution with honest agents only. We use validity claims to formalize that all executions of a protocol satisfy certain properties, e.g., that the session-key computed by two honest agents is never known to the adversary. For an example of this, please see [137].

In implementation, the syntax within Tamarin's specification language for security properties is defined as follows:

- `All` for universal quantification, temporal variables are prefixed with `#`
- `Ex` for existential quantification, temporal variables are prefixed with `#`
- `==>` for implication
- `&` for conjunction
- `|` for disjunction
- `not` for negation
- `f @ i` for action constraints
- `i < j` for temporal ordering

- `#i = #j` for an equality between temporal variables ‘`i`’ and ‘`j`’
- `x = y` for an equality between message variables ‘`x`’ and ‘`y`’

Example 2.2.2. We give an example lemma from the Needham-Schroeder-Lowe public key protocol model [130], describing the property of nonce secrecy:

```
lemma nonce_secrecy:
  "not(
       Ex A B s #i.
         Secret(A, B, s) @ #i
       & (Ex #j. K(s) @ #j)
       & not (Ex #r. RevLtk(A) @ #r)
       & not (Ex #r. RevLtk(B) @ #r)
  )"
```

Informally, this says “If actors A and B are both honest, then secret term `s` is not known to the adversary”. In more depth, `Secret(A, B, s) @ #i` is an action fact recorded in $traces_E(R)$ (where R is the protocol model) at time-point `#i`, and `K(s) @ #j` states that the adversary has learnt the secret term `s` at some arbitrary time-point `#j`, recorded in $traces_E(R)$ by `K(s)`. This example assumes a threat model where *some* actors can be compromised, and when this occurs, this is recorded by the “`RevLtk(ActorName)`” fact. Here, we require that there does not exist one of these action facts recorded in $traces_E(R)$ for either actor A or B; this enforces that the parties we consider here have not been compromised.

The lemma overall states that if someone claims to have set up a shared secret `s`, then there cannot exist any time-point `#j` such that the adversary knows this secret unless the adversary has performed a key reveal action on either the initiator or responder, i.e., `RevLtk(A)` or `RevLtk(B)`.

2.2.5 Constraint solving

Satisfiability claims (or in implementation, lemmas) are proved or disproved through a constraint solving algorithm, which incrementally builds a satisfying trace through a series of constraint reduction steps. Validity claims can also be solved using the same process, by proving or disproving satisfiability of the negated claim. The constraint solving algorithm exhaustively searches for satisfying traces by beginning with the constraints introduced by a claim and solving these through a series of constraint reduction steps.

Throughout this section, we use a convention for variable names. We use f to range over facts, i and j over temporal variables, ri over multiset rewriting rule instances, and u and v over natural numbers, unless otherwise specified.

2.2.5.1 Dependency graphs

TAMARIN makes use of dependency graphs to represent protocol executions in a way that captures causal dependencies between the rules. Dependency graphs comprise nodes, labelled with rule instances, and edges representing the dependencies between nodes. The edges of a dependency graph show causal dependencies between nodes: an edge from a conclusion of one node to a premise of another marks that a fact is generated by the former and consumed by the latter. Persistent facts (prefixed with !) are allowed to have multiple outgoing edges. The rules at each node are ground instances in the equational theory. Meier defines dependency graphs formally in [137, §8.1] as follows:

For an equational theory E and a multiset rewriting system R, $dg := (I, D)$ is an R, E-dependency graph if the nodes I are a sequence of ground instances of $R \cup \{\texttt{Fresh}\}$ (i.e., $I \in ginsts_E(R \cup \{\texttt{Fresh}\})^*$), the edges $D \subseteq \mathbb{N}^2 \times \mathbb{N}^2$, and dg satisfies the conditions DG1-4 below. In the following, we refer to the premises and conclusions of dg as pairs (i, u) such that i is a node of dg and $u \in idx(prems(I_i))$ or $u \in idx(concs(I_i))$ respectively. These have associated premise and conclusion facts $prems(I_i)_u$ and $concs(I_i)_u$ respectively.

DG1. for every edge $((i, u), (j, v)) \in D, i < j$, and the conclusion fact of (i, u) is equal modulo E to the premise fact of (j, v);
DG2. every premise of dg has exactly one incoming edge;
DG3. every conclusion of dg with a linear conclusion fact has at most one outgoing edge;
DG4. instances of `Fresh` in I are unique.

The trace of a dependency graph (I, D) where $I := \langle ri_0, \ldots, ri_n \rangle$ is defined as

$$\langle set(acts(ri_0)), \ldots, set(acts(ri_n)) \rangle$$

The set of all R, E-dependency graphs is denoted $dgraphs_E(R)$. In order to use dependency graphs as an alternative formulation of the multiset rewriting semantics above, it is necessary to prove equivalence between the two. Meier proves the following theorem in [137]:

For every multiset rewriting system R and every equational theory E,

$$traces_E(R) =_E \{trace(dg) \mid dg \in dgraphs_E(R)\}$$

Dependency graphs in TAMARIN are also extended to support message deduction over associative and commutative equational theories like Diffie-Hellman (see [159, §3.2.3]) though this is not directly relevant to our work.

2.2.5.2 Guarded trace properties

In order to ensure we can solve a trace property, we must restrict the set of allowed properties. We must ensure that they are closed under negation so that we can convert between satisfiability and validity. Further, we must ensure that when evaluating the trace formula it is sufficient to consider only the subterms that appear in a given trace. To do this, we first define the notion of guarded trace property, a class of trace properties which meet these restrictions.

A trace formula φ is a guarded trace formula if all logical operators are $\wedge, \vee, \forall$, and $\exists$, negation is applied only to trace atoms and $\perp$, and all of its quantifiers are of the form $\exists x \,.\, g \wedge \psi$ or $\forall x \,.\, \neg g \vee \psi$ such that:

G1. $x \subseteq \mathcal{V}_{msg} \cup \mathcal{V}_{temp}$

G2. either

a) g is an action $f \ @ \ i$ and $x \subseteq \mathit{vars}(f \ @ \ i)$

b) or g is an equality $s \approx t$, $\mathit{vars}(s) \cap x = \emptyset$, and $x \subseteq \mathit{vars}(t)$.

A guarded trace formula φ is a guarded trace property if it is closed and for all subterms t of φ, $\mathit{root}(t)$ is a variable, a public name, or an irreducible function symbol in Σ.

Tamarin can automatically convert most trace properties that a modeller may write to guarded trace properties, so it is not typically necessary to keep these requirements in mind when writing a model. They are, however, necessary for the constraint solving performed by Tamarin. A constraint is either a guarded trace formula or,

1. a node constraint, written $i : ri$ for an index i and rule instance ri,
2. a premise constraint, written $f \blacktriangleright_v i$ for a fact f occurring as the v^{th} premise at index i, or
3. an edge constraint, written $(i,u) \rightarrowtail (j,v)$ for an edge between the u^{th} conclusion at index i and the v^{th} premise at index j.

A constraint system is a finite set of constraints. We define a constraint satisfaction relation between a dependency graph $dg = (I,D)$ with valuation θ and a constraint γ, written $(dg,\theta) \Vdash_E \gamma$, as

$$
\begin{array}{ll}
(dg,\theta) \Vdash_E i : ri & \text{iff } \theta(i) \in \mathit{idx}(I) \text{ and } ri\theta =_E I\theta(i) \\
(dg,\theta) \Vdash_E f \blacktriangleright_v i & \text{iff } \theta(i) \in \mathit{idx}(I) \text{ and } f\theta =_E \mathit{prems}(I\theta(i))_v \\
(dg,\theta) \Vdash_E (i,u) \rightarrowtail (j,v) & \text{iff } (\theta(i),u) \rightarrowtail (\theta(j),v) \in D \\
(dg,\theta) \Vdash_E \varphi & \text{iff } \mathit{trace}(dg,\theta) \vDash_E \varphi
\end{array}
$$

For a constraint system Γ, we extend this relation such that $(dg,\theta) \Vdash_E \Gamma$ iff

$$\forall \gamma \in \Gamma \,.\, (dg,\theta) \Vdash_E \gamma$$

A R,E-solution of a constraint system Γ is an R,E-dependency graph dg such that there is a valuation θ with $(dg,\theta) \Vdash_E \Gamma$.

2.2.5.3 Constraint reduction

The constraint solving algorithm used by TAMARIN broadly builds a constraint-reduction relation $\rightsquigarrow_{R,E}$, refining the initial constraint system defined by a guarded trace property into a set of refined constraint systems. This is performed until either a dependency graph with a valuation satisfying a refined constraint system is found, or all resulting constraint systems contain contradictory constraints. This implies either that a trace violating the specific security property has been found, or the property has been verified as upheld respectively.

The constraint reduction relation is made up of two parts: case distinction rules, which add additional constraints by splitting a constraint system into a set of constraint systems representing possible cases, and simplification rules which remove redundant constraints or constraint systems entirely. Each constraint reduction rule in the relation must be proven to be both complete and sound, i.e., for a rule $\Gamma \rightsquigarrow_{R,E} \{\Gamma_1, \ldots, \Gamma_n\}$, an R,E-solution of Γ is an R,E-solution of at least one of the constraint systems $\{\Gamma_1, \ldots, \Gamma_n\}$, and vice versa respectively. Please see [137, Figure 8.3] for the complete figure showing the constraint reduction rules that form the basic $\rightsquigarrow_{R,E}$ relation. In [137, §8.2.3], Meier proves that these rules are sound and complete, and that from every solved constraint system it is possible to extract a solution trace.

2.2.6 Implementation

These constraint reduction rules are implemented in TAMARIN as three categories of rules: case distinction rules, contradiction rules, and simplification rules. Case distinction rules are those which may result in more than one constraint system, and are represented by goals in TAMARIN. Contradiction rules are rules which immediately reduce a constraint system to $\perp$. Simplification rules are all the remaining rules, which reduce a constraint system to a single other constraint system.

Proofs are created by incrementally solving applicable goals until either all goals are solved (and therefore a satisfying trace has been found) or the initial constraint system has been reduced to $\perp$. Both contradiction and simplification rules are applied eagerly by TAMARIN at each step between solving goals, and the proof is thus represented by the series of goals solved and contradictions applied, omitting simplification rules that would make proofs difficult to read.

Goals are chosen at each step according to a heuristic function, which attempts to assign a useful priority order to different goals. Finally, TAMARIN precomputes a limited set of case distinction rules. This precomputation simplifies some common constraint-reduction steps that are applied repeatedly in a protocol, contracting several constraint reductions into one proof step. Specifically, this step precomputes the resulting constraint systems from solving a premise goal, for all facts that appear as premises in protocol rules and message deduction rules.

For more detail, intuition, examples, and instruction on how this works in implementation, please see [137, 159] and [35].

2.3 Symbolic modelling assumptions

We use the Tamarin prover in this thesis to build models of large protocols and their desired security properties, and then perform symbolic analysis of these models. Symbolic modelling necessarily has a large number of restrictions on the results achieved, based upon the underlying assumptions which we discuss briefly here.

Symbolic analysis does not consider computational attacks on a protocol, instead focusing on the logic of protocol interactions. This requires us to make assumptions about the primitives used in the protocol, which restricts the power of the analysis. We make the following assumptions:

- Dolev-Yao Adversary: the adversary controls the network channels [86].
- Symbolic Representation: as described above, information is contained in terms. Any party (including the adversary) can either know a term in its entirety, or not know it at all. A party cannot learn e.g., a single bit of a term, or half a term.
- Perfect Cryptography: we assume that the cryptographic primitives used are perfect. This means that e.g., an adversary can learn the term m from the symmetrically encrypted $\{\!| m |\!\}_k^s$ term if and only if it knows the key, k.
- Hash Functions: we assume that hash functions are one-way and injective.
- Randomness: we assume all fresh random terms generated with e.g., `Fr(~x)` unpredictable, and unique (as described, no two fresh terms generated separately are equal).

We give examples from our Tamarin models and security properties written as lemmas in the relevant chapters, and complete models are attached electronically.

"Cyber-attacks have plunged entire cities into darkness."
— Barack Obama

"The thing about infrastructure is that everyone uses it. If it's secure, it's secure for everyone. And if it's insecure, it's insecure for everyone. This forces some hard policy choices."
— Bruce Schneier

3 Secure Authentication in the Grid: A Formal Analysis of DNP3: SAv5

3.1 Introduction

Most of the world's power grids are monitored and controlled remotely. In practice, power grids are controlled by transmitting control and monitoring messages, between authorised operators ('users') that send commands from control centers ('master stations'), and substations or remote devices ('outstations'). The messages may be passed over a range of different media, such as direct serial connections, ethernet, Wi-Fi, or un-encrypted radio links. As a consequence, we cannot assume that these channels guarantee confidentiality and authenticity.

The commands that are passed over these media are critical to the security of the power grid: they can make changes to operating parameters such as increases or decreases in voltage, opening or closing valves, or starting or stopping motors [83]. It is therefore desirable that an adversary in control of one of these media links should not be able to insert or modify messages. This has motivated the need for a way to authenticate received messages.

The DNP3 standard, more formally known as IEEE 1815-2012, the "Standard for Electric Power Systems Communications – Distributed Network Protocol" [110], is used by most of the world's power grids for communication, and increasingly for other utilities such as water and gas.

Secure Authentication version 5 (SAv5) is a new protocol family within DNP3, and was standardised in 2012 (Chapter 7 of IEEE 1815-2012 [110], based on IEC/TS 62351-5 [109]). SAv5's goal is to provide authenticated communication between parties within a utility grid. For example, this protocol allows a substation or remote device within a utility grid to verify that all received commands were genuinely sent by an authorised user, that messages have not been modified, and that messages are not being maliciously replayed from previous commands.

Given the security-critical nature of the power grid, one might expect that DNP3: SAv5 would have attracted substantial scrutiny. Instead, there has been very little analysis, except for a few limited works. One possible explanation is the inherent complexity of the DNP3: SAv5 protocol, as it consists of several interacting sub-protocols that maintain state to update various keys, which results in a very complex state machine for each of the participants. Such protocols are notoriously hard to analyse by hand, and the complex looping constructions pose a substantial challenge for protocol security analysis tools. Moreover, it is not sufficient to analyse each sub-protocol in isolation. While this has been known in theory for a long time [118], practical attacks that exploit cross-protocol interactions have only been discovered more recently, e.g., [46, 134]. In general, security protocol standards are very hard to get right, e.g. [38, 77, 147].

Chapter overview We start by describing the Secure Authentication v5 standard in Section 3.2. We describe the sub-protocols' joint modelling in Section 3.3, and their analysis and results in Section 3.4. We consider DNP3: SAv5 under partial compromise and its associated results in Section 3.5, before presenting our overall recommendations in Section 3.6, We finally consider claimed attacks from the literature against DNP3 in Section 3.7, before concluding this chapter in Section 3.8.

A public archive with our protocol models can be found at [71], but they are also included electronically with this thesis.

3.2 The DNP3 standard

The DNP3 standard [110] gives both high level and semi-formal descriptions, to serve as an implementation guide, as well as providing an informal problem statement and conformance guidelines. The Secure Authentication v5 protocol is described in Chapter 7 of [110]. We give an overview of the system and its sub-protocols, before describing the threat model from SAv5.

3.2.1 System and Sub-Protocols

There are three types of actor in SAv5: the (single) **Authority**, the **Users** (operating from a Master station), and the **Outstations**. The Authority decides who are legitimate users, and generates new (medium-term) Update Keys for these users. Users send control packets to outstations, who act upon them if they are successfully authenticated. Outstations send back (similarly authenticated) monitoring packets. Each user can communicate with multiple outstations, and each outstation can communicate with multiple users. Users regularly generate new (short-term) **Session Keys** for each direction of this communication, and transport these keys to the outstations. These are updated and distributed using so-called **Update Keys**, which can be considered as medium-term keys. In the symmetric mode, update keys are distributed and updated using long-term **Authority Keys**. In the asymmetric mode, update keys are distributed and updated using long-term public/private key pairs for the Authority and each User and Outstation. These keys are used by four sub-protocols: the symmetric *Update Key Change Protocol*, the asymmetric *Update Key Change Protocol*, the *Session Key Update Protocol*, and the *Critical ASDU Authentication Protocol*. See Figure 3.1 for an overview of the sub-protocols' relationships.

Initial Key Distribution: Before any protocols are run, a long-term Authority Key and an initial medium-term update key must be pre-distributed to each party. If the parties support the optional asymmetric mode of the *Update Key Change Protocol*, public/private key pairs will be generated and the private halves securely distributed to their respective owners; the public keys are distributed to all involved parties. These keys are distributed "over a physical channel" (e.g., via USB stick) to the respective parties. N.B. Session Keys are *not* pre-distributed.

The *Session Key Update Protocol*: Before parties can exchange control or monitoring messages, the user and outstation must initialise session keys. This sub-protocol initialises (and later updates) a new, symmetric Session Key for each communication direction.

After ~15 minutes or ~1,000 critical messages (both configurable) the session keys will expire. The user and outstation run the *Session Key Update Protocol* again, where the user generates fresh symmetric session keys, and sends them to the outstation, encrypted with their current update key. These session keys *must* remain secret, but the secrecy of new keys importantly does not rely on the secrecy of previous session keys.

All sub-protocols use sequence numbers and freshly generated Challenge Data with the aim of preventing replay attacks.

The *Critical ASDU Authentication Protocol*: Outstations use this sub-protocol to verify that received control packets were genuinely sent by a legitimate user. Vice-versa, this sub-protocol allows a user to confirm that received monitoring packets were genuinely sent by a legitimate outstation. These packets are called 'Critical ASDUs', or Application Service Data Units. As this is an authentication-only protocol, **Critical ASDUs are not confidential.**

After this sub-protocol's first execution, the faster 'Aggressive Mode' may be performed: this cuts the non-aggressive mode's three messages to just one by sending the ASDU and a keyed HMAC in the same message.

The *Update Key Change Protocol*: After a longer time, the update key may expire. The user and outstation (helped by the Authority) will execute the *Update Key Change Protocol*. A new update key is created by the Authority, and sent to both the user and outstation.

Protocol 'Purpose': The high-level purpose of the protocol suite as a whole is to provide authentication for the validity of critical DNP3 packets (ASDUs); it uses three sub-protocols to achieve this. The sub-protocols that make up this protocol suite each have distinct but inter-related purposes: the *Critical ASDU Authentication Protocol* aims to verify (or reject) correctly the authenticity of Critical ASDUs, but to run this protocol, users and outstations must be in possession of valid session keys. The *Session Key Update Protocol* aims to distribute (and maintain the confidentiality and integrity of) new pairs of regularly changing session keys for use by the *Critical ASDU Authentication Protocol*; to distribute these, it encrypts the session keys with medium-term update keys. Finally, the *Update Key Change Protocol* aims to distribute (and maintain the confidentiality and integrity of) new, rarely changing update keys, for use by the *Session Key Update Protocol* (encrypted with a pre-distributed Authority Key, or the recipient's public key).

From the outset, it seems apparent that the security of the *Critical ASDU Authentication Protocol* probably depends on the security of the *Session Key Update Protocol* (as one distributes the session keys for the other), and by extension, the *Update Key Change Protocol*: we investigate this relationship in more detail throughout this chapter. See Figure 3.1 for a visual overview of how the protocols relate to each other.

3.2.2 Protocol Descriptions

We now give more detailed descriptions of the four sub-protocols in Secure Authentication v5. By way of notational preliminaries, $\{\!|\, m \,|\!\}_k^s$ denotes the symmetric encryption of term m under key k; similarly HMAC$_k(m)$ denotes the HMAC of term m symmetrically keyed by k. Likewise, $\{\!|\, m \,|\!\}_k^a$ denotes the asymmetric encryption of term m under public key k, and $\{\!|\, m \,|\!\}_{sk}^{asig}$ denotes the asymmetric signature of term m under private key sk.

3.2.2.1 The *Session Key Update Protocol*:

See Figure 3.2. This is also the first sub-protocol run after a system restarts, to initialise the shared session keys.

S1. The user sends a Session Key Status Request. The user moves from "Init" to the state "Wait for Key Status".

S2. The outstation generates fresh challenge data CD$_j$, and increments its Key Change Sequence Number, KSQ. It sends a Session Key Status message (SKSM$_j$) to the user, containing the KSQ value, user ID, USR, Key Status, and CD$_j$. The outstation moves from "Start" to the state "Security Idle".

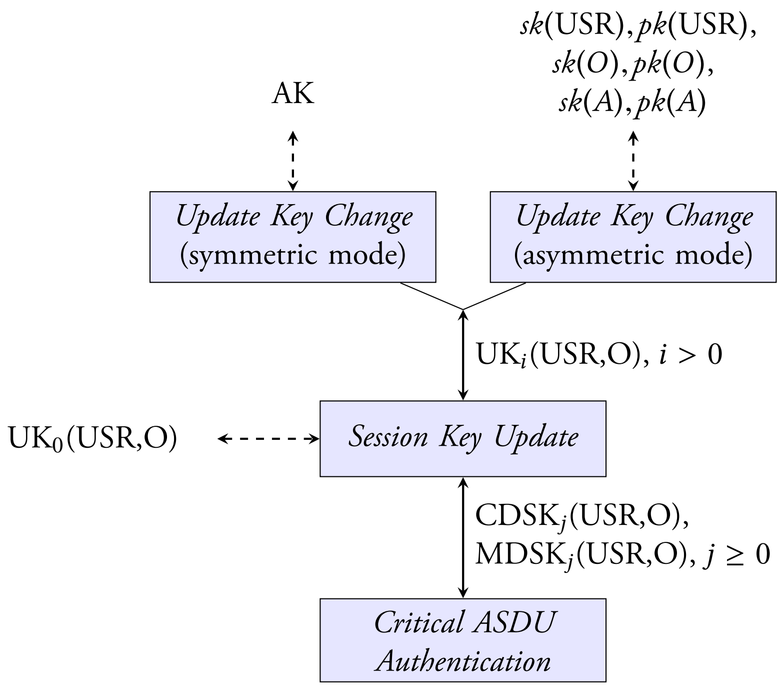

$sk(\text{USR}), pk(\text{USR}),$
$sk(O), pk(O),$
$sk(A), pk(A)$
AK
Update Key Change (symmetric mode)
Update Key Change (asymmetric mode)
$\text{UK}_i(\text{USR,O}),\ i > 0$
$\text{UK}_0(\text{USR,O})$
Session Key Update
$\text{CDSK}_j(\text{USR,O}),$
$\text{MDSK}_j(\text{USR,O}),\ j \geq 0$
Critical ASDU Authentication

Figure 3.1: Relationships between sub-protocols, the flow of keys between them (continuous lines), and required pre-shared keys (dashed lines).

S3. The user generates two new session keys (one for each direction), CDSK and MDSK, and sends a Session Key Change Message to the outstation (SKCM_j). This contains the KSQ and USR values, and the encryption of the new keys and the previously received SKSM_j message from the outstation, encrypted with the current symmetric update key. The user moves to the state "Wait for Key Change Confirmation".

S4. The outstation decrypts this with the shared update key, and checks that SKSM_j is the same as it previously sent. If so, the outstation increments KSQ, and generates new challenge data, CD_{j+1}; it sends another Session Key Status Message (this time SKSM_{j+1}), but as session keys have been set, the message now also includes an HMAC of SKCM_j, keyed with the MDSK.

S5. The user verifies that the received HMAC was generated from SKCM_j. If so, the user and outstation start to use the new session keys. If not, the user and outstation mark the keys as invalid, and retry the protocol. The user state moves to "Security Idle".

3.2.2.2 The *Critical ASDU Authentication Protocol*:

See Figure 3.3. This is the main data authentication protocol, and is used to verify the authenticity of critical ASDUs. This can only run *after* the first execution of the *Session Key Update Protocol*, and it can run in both the control and monitoring directions, User→Outstation and Outstation→User respectively. Here we present it in the control direction; the direction

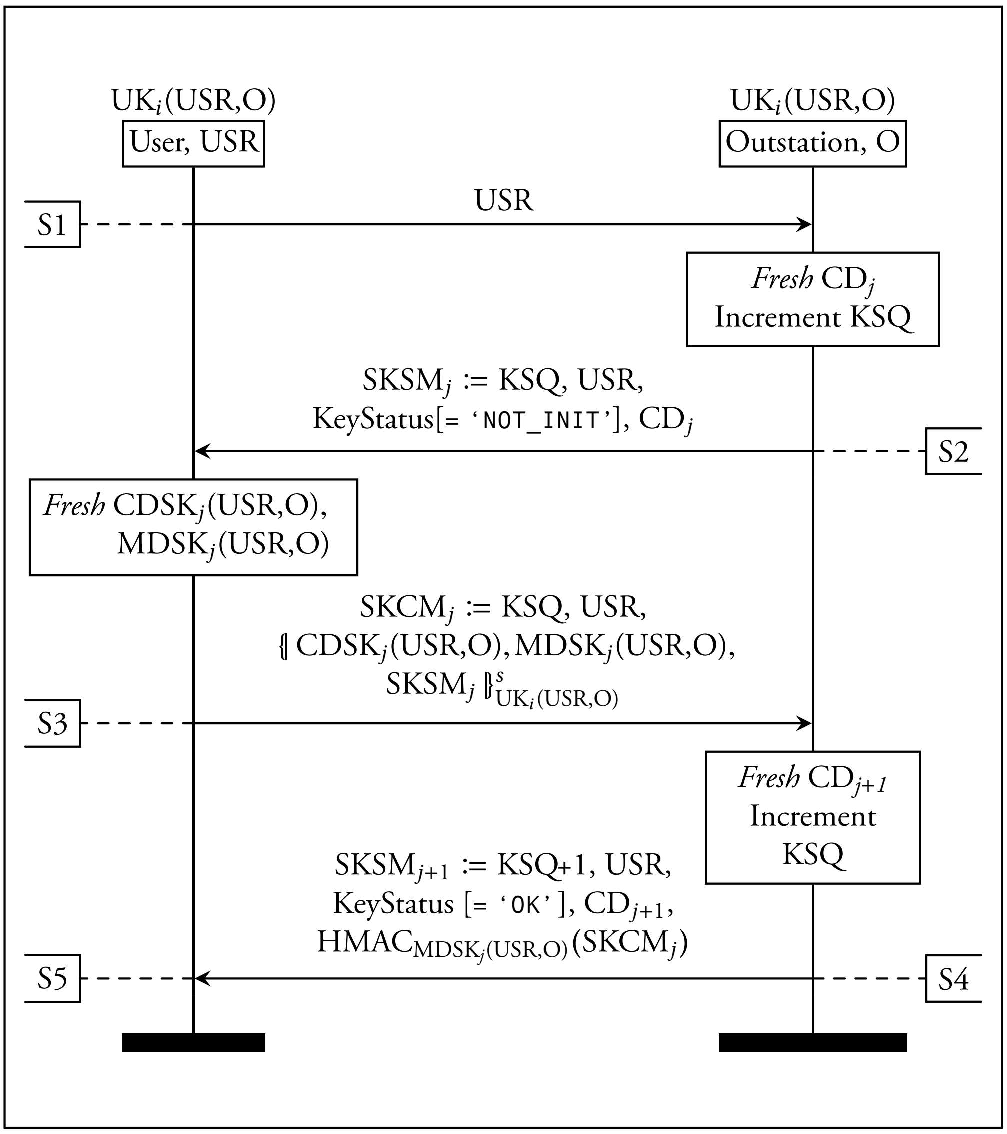

UK_i(USR,O)
User, USR
UK_i(USR,O)
Outstation, O
USR
S1
Fresh CD_j
Increment KSQ
$SKSM_j$:= KSQ, USR,
KeyStatus[= 'NOT_INIT'], CD_j
S2
Fresh $CDSK_j$(USR,O), $MDSK_j$(USR,O)
$SKCM_j$:= KSQ, USR,
$\{\!| CDSK_j(USR,O), MDSK_j(USR,O), SKSM_j |\!\}^s_{UK_i(USR,O)}$
S3
Fresh CD_{j+1}
Increment KSQ
$SKSM_{j+1}$:= KSQ+1, USR,
KeyStatus [= 'OK'], CD_{j+1},
$HMAC_{MDSK_j(USR,O)}(SKCM_j)$
S4
S5

Figure 3.2: The *Session Key Update Protocol*. The labels S1–5 identify the protocol rules described in Section 3.2.2.1

determines which key is used for the HMAC in the final message (A3), i.e., CDSK or MDSK. First, the non-aggressive mode; both parties start in the state "Security Idle":

A1. The user sends a critical ASDU, which the outstation must authenticate.

A2. On receipt of this ASDU, the outstation increments its Challenge Sequence Number, CSQ, and sends an Authentication Challenge (AC), which contains the user's ID, USR, fresh challenge data, CD, and the CSQ value. The outstation moves to the state "Wait for Reply".

A3. The user sends an Authentication Reply message, which contains the CSQ, USR, and an HMAC of the previously received Authentication Challenge message, AC, and the critical ASDU it seeks to authenticate. This HMAC is keyed with the Control Direction Session Key, CDSK.

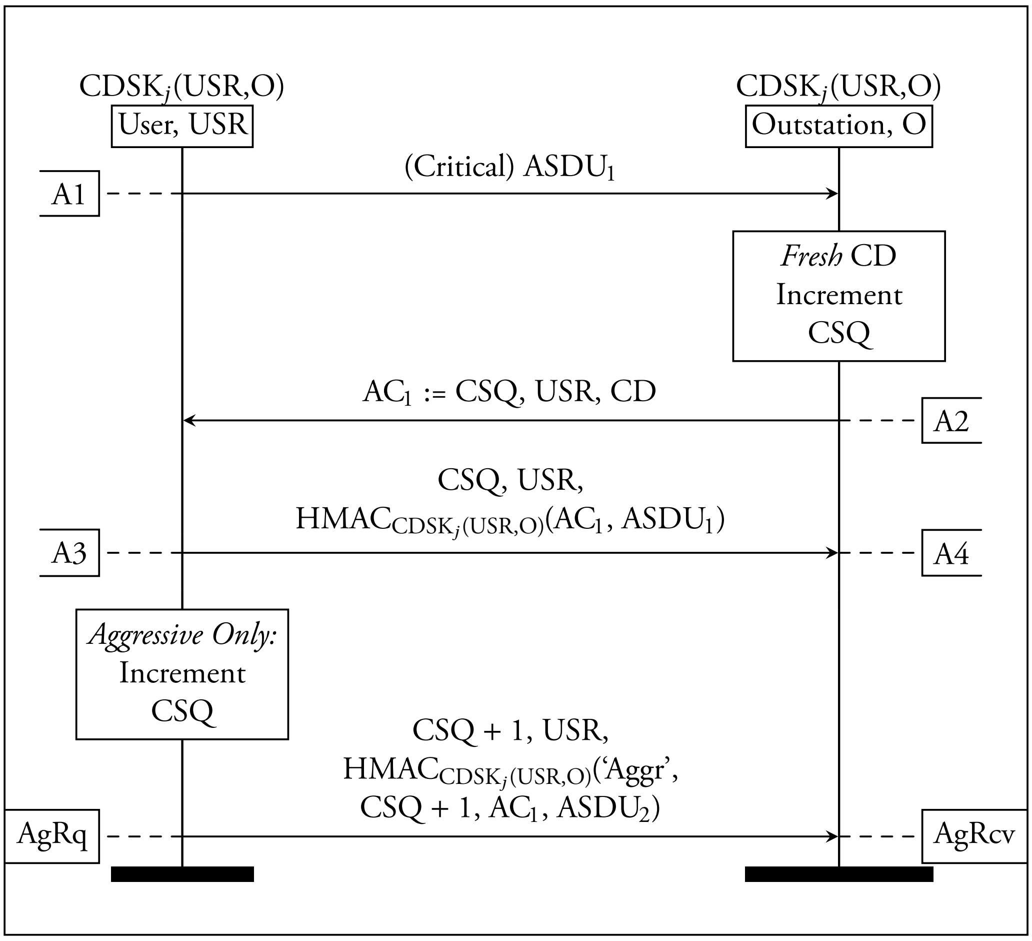


Figure 3.3: The *Critical ASDU Authentication Protocol*, Control Direction, Non-Aggressive *and* Aggressive Modes. The labels A1–4 identify the protocol rules described in Section 3.2.2.2

A4. The outstation verifies that the HMAC was constructed with the AC message it sent, the critical ASDU, and keyed with the current CDSK. If it succeeds, the outstation acts upon this critical ASDU; if it fails, it does not execute it. Regardless of the outcome, the outstation returns to the state "Security Idle".

Aggressive Mode: Once the non-aggressive sub-protocol has run once, the user may send an Aggressive Mode Request ('AgRq' in Figure 3.3). This contains both the new ASDU to be authenticated, the incremented CSQ, and an HMAC in the same message. This HMAC is calculated over the last Authentication Challenge message the user received, and the entire preceding message it is being sent in.

The outstation then checks ('AgRcv' in Figure 3.3) that the HMAC was constructed with the last Authentication Challenge, and that the CSQ is incremented from the last message. If so, it accepts and acts upon the ASDU.

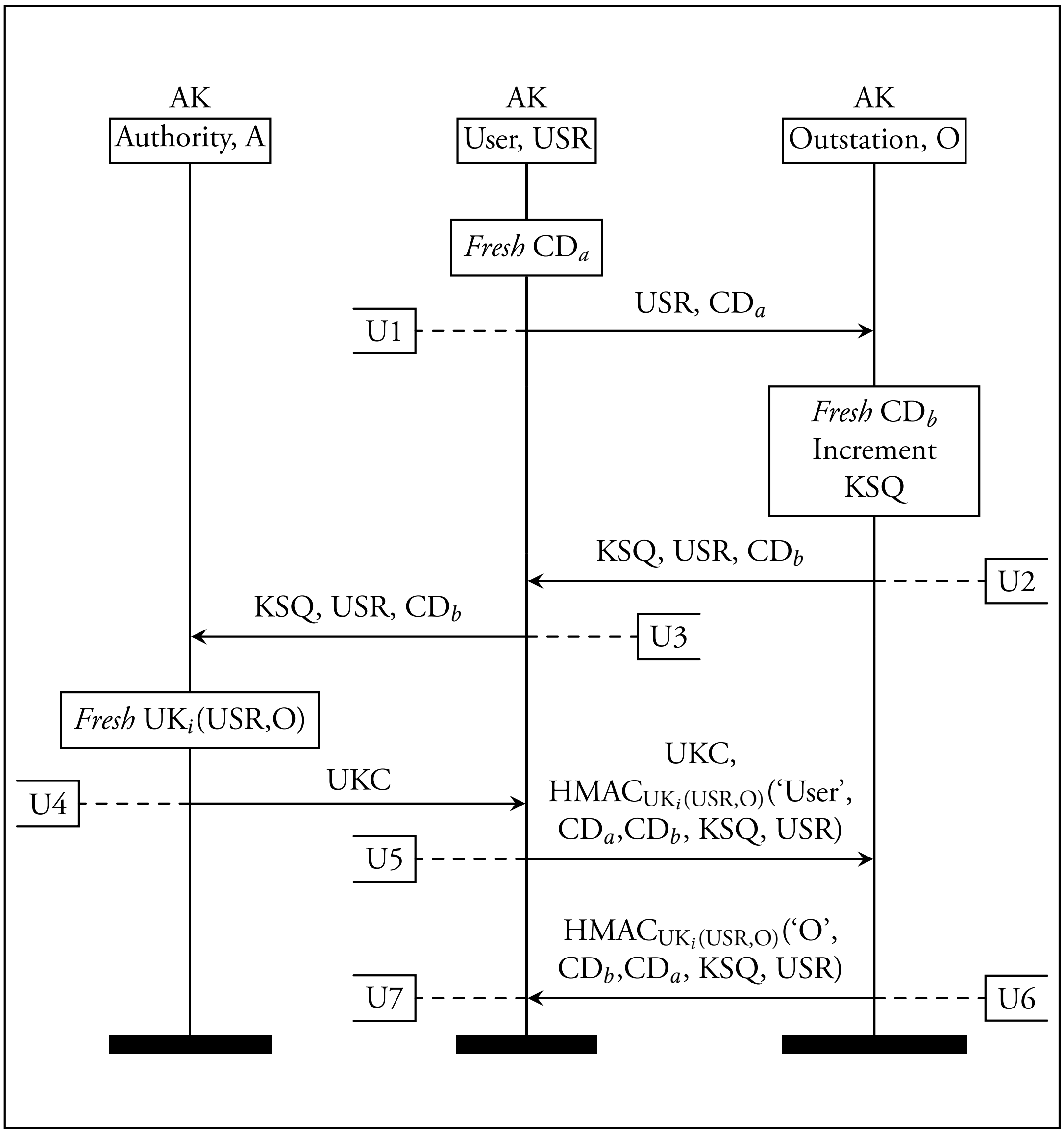


Figure 3.4: The *Update Key Change Protocol*, symmetric mode. The labels U1–7 identify the protocol rules described in Section 3.2.2.3. In U4 and U5, UKC is the tuple KSQ, USR, $\{\!|\, USR, UK_i(USR,O), CD_b \,|\!\}^{s}_{AK}$

3.2.2.3 The *Update Key Change Protocol* (Symmetric Mode):

See Figure 3.4. This key-transport sub-protocol has two modes; symmetric (default) and asymmetric (optional; described in Section 3.2.2.4). Both allow users and outstations to change the symmetric update key used by the previous protocol. Both devices start in "Security Idle"; the outstation always remains here. The symmetric mode of this protocol proceeds as follows:

U1. The user sends an Update Key Change Request message, containing the user's ID, USR, and freshly generated challenge data, CD_a. The user moves to the state "Wait for Update Key Reply".

U2. Upon receipt of this message, the outstation increments its Key Change Sequence Number (the same variable as in the previous sub-protocol), and also generates fresh challenge data, CD_b. It sends the new value of KSQ, USR and CD_b to the user in an Update Key Change Reply message.

U3. The user forwards this message on to the Authority.[1]

U4. The Authority creates a new update key. It encrypts the key, USR, and CD_b with the Authority Key, and transmits it, KSQ, and USR back to the user.

U5. The user decrypts this, and forwards both this message (Update Key Change), and an Update Key Change Confirmation (UKCC) message to the outstation. This is an HMAC of the user's full name, both challenge data (CD_a and CD_b), KSQ, and USR, and it is keyed with the *new* update key. The user moves to the state "Wait for Update Key Confirmation".

U6. The outstation decrypts the first part of the message to learn the new update key, and verifies that the UKCC HMAC was created with the correct challenge data and KSQ from step U2. If so, it sends back its own UKCC message (also keyed with the new update key), but with the order of the challenge data swapped, and with its name, rather than the user's.

U7. If the user can validate this HMAC (by checking that it was created with the challenge data and KSQ values from this same protocol run, keyed with the new update key), then it accepts the message, and both parties start to use the new update keys. If this fails, the parties retry the protocol. Regardless of outcome (except timeout), the user moves back to the state "Security Idle".

3.2.2.4 The *Update Key Change Protocol* (Asymmetric Mode):

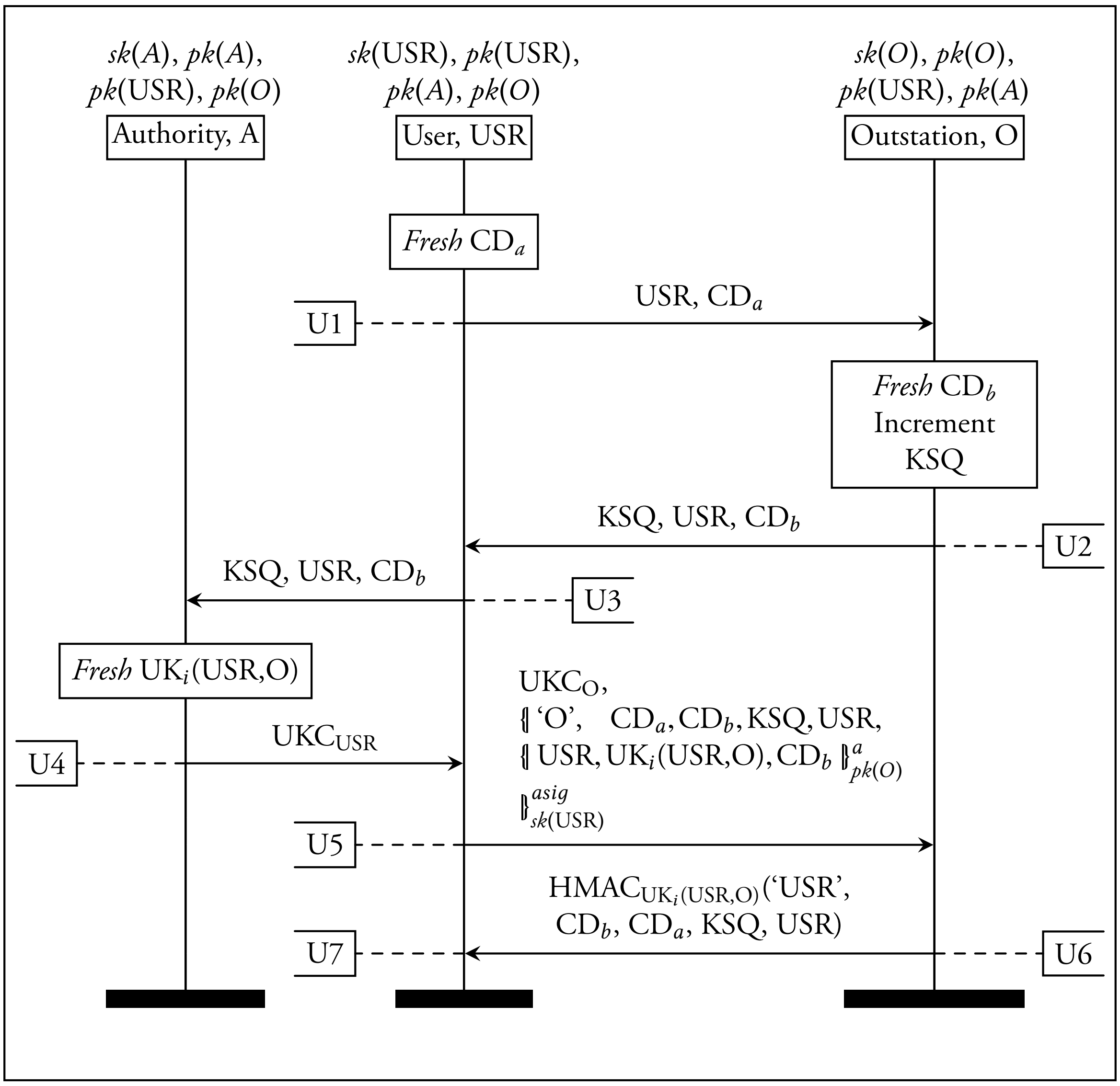
The asymmetric mode of the *Update Key Change Protocol* is very similar to the symmetric mode. See Figure 3.5 for the message sequence chart of the Asymmetric mode of the *Update Key Change Protocol*. The overall message flows are similar to the symmetric mode, and messages are effectively identical for U1–3. In message U4, the message transmitted is encrypted asymmetrically, not symmetrically. In message U5, the UKC object within the message (`g120v13`) is asymmetrically encrypted with the Outstation's public key. The overall U5 message also includes an Update Key Change Signature object (`g120v14`), rather than a (symmetrically encrypted) Update Key Change Confirmation object (`g120v15`). This Update Key Change Signature object is the signature over:

– Outstation Name
– User Challenge Data (CD_a in the MSC)
– Outstation Challenge Data (CD_b in the MSC)

[1]U3 and U4 are technically out of scope for DNP3: SAv5.

– Key Change Sequence Number (KSQ)
– User Number (USR)
– Encrypted Update Key Data (from UKC)

This object is signed by the User's private key, $sk(\text{USR})$, which allows the Outstation to verify the signature with the User's public key, $pk(\text{USR})$. In message U6, the Outstation then sends back to the User an object signed by the Outstation's private key, i.e., $sk(\text{O})$. This signature is over all the same data as the previous signature (from U5), but as it is signed by the Outstation's private key, an adversary should not be able to forge this only from knowledge of the terms listed above and the previously seen signature (from U5).



$sk(A), pk(A), pk(\text{USR}), pk(O)$
Authority, A
$sk(\text{USR}), pk(\text{USR}), pk(A), pk(O)$
User, USR
$sk(O), pk(O), pk(\text{USR}), pk(A)$
Outstation, O
Fresh CD_a
U1
USR, CD_a
Fresh CD_b Increment KSQ
KSQ, USR, CD_b
U2
KSQ, USR, CD_b
U3
Fresh $\text{UK}_i(\text{USR,O})$
U4
UKC_{USR}
UKC_{O}, $\{\!|\, \text{'O'}, \text{CD}_a, \text{CD}_b, \text{KSQ}, \text{USR}, \{\!|\, \text{USR}, \text{UK}_i(\text{USR,O}), \text{CD}_b \,|\!\}^{a}_{pk(O)} \,|\!\}^{asig}_{sk(\text{USR})}$
U5
$\text{HMAC}_{\text{UK}_i(\text{USR,O})}(\text{'USR'}, \text{CD}_b, \text{CD}_a, \text{KSQ}, \text{USR})$
U7
U6

Figure 3.5: The *Update Key Change Protocol*, asymmetric mode. The labels U1–7 identify the protocol rules described previously. In messages U4 and U5, UKC_x is equal to the tuple: KSQ, USR, $\{\!|\, \text{USR}, \text{UK}_i(\text{USR,O}), \text{CD}_b \,|\!\}^{a}_{pk(x)}$, where x is the intended recipient of the message (i.e., the User in message U4, and the Outstation in message U5).

3.2.3 Threat Model and Security Properties

In this section we describe how we arrived at the threat model and security properties that we formally analyse. This is not as straightforward as one might think, as security properties are often informally and minimally described in protocol standards. For transparency, we will quote the original standards where possible. We use colored boxes to denote verbatim quotations from other documents.

The standard has a "Problem description" section [110, p.13] that describes "the security threats that this specification is intended to address". We reproduce this section *in its entirety* below:

5.2 Specific threats addressed (from IEEE 1815-2012 p.13)

This specification shall address only the following security threats, as defined in IEC/TS 62351-2:

- spoofing;
- modification;
- replay
- eavesdropping — on exchanges of cryptographic keys only, not on other data.

Additionally, the general principles section contains a subsection "Perfect forward secrecy" that suggests an implicit security requirement. We could not determine any other sections that would imply security requirements.

The wording of the above section suggests that all listed terms are defined in IEC/TS 62351-2 [108]. This is not the case: [108] defines only some of these concepts. In particular, "modification" and (perfect) "forward secrecy" are not defined. We address the listed concepts in turn, starting from the ones which are defined.

Spoofing. The standard specifies that spoofing is defined through [108] as:

2.2.191 Spoof (from IEC/TS 62351-2 p.39)

Pretending to be an authorized user and performing an unauthorized action.
[RFC 2828]

While this definition references RFC 2828 [169], there is a difference, in that [169] equates spoofing and masquerading, but does not reference unauthorized actions:

spoofing attack (from RFC 2828)

(I) A synonym for "masquerade attack".

where masquerade is defined in the RFC as

masquerade attack (from RFC 2828)

a type of attack in which one system entity illegitimately poses as (assumes the identity of) another entity. (see: spoofing attack.)

Thus, the RFC equates spoofing and masquerading. Analogously, the DNP3 standard directly relies on [108], which defines masquerading as

2.2.131 Masquerade	(from IEC/TS 62351-2 p.30
The pretence by an entity to be a different entity in order to gain unauthorized access. [ATIS]	

Here, ATIS [19] is a glossary from which this particular definition is taken. Hence it seems that within the context of DNP3, spoofing and masquerading are interchangeable, similar to the statements in RFC 2828. However, the definitions in the DNP3 standard [109] are closer to [19] than to [169], since they additionally include the aspect of unauthorized access/action. Note that the DNP3 standard has no explicit concept of authorization; this seems out of the standard's scope.

Replay

2.2.159 Replay Attack	(from IEC/TS 62351-2 p.35)
1. A masquerade which involves use of previously transmitted messages. [ISO/IEC 9798-1:1997]	

This is a verbatim copy of a similar section in the reference ISO/IEC 9798-1:1997 [111], and suggests that replay is a special case of masquerading/spoofing.

Eavesdropping

2.2.92 Eavesdropping	(from IEC/TS 62351-2 p.25)
Passive wiretapping done secretly, i.e., without the knowledge of the originator or the intended recipients of the communication. [RFC 2828]	

This is a verbatim copy from the definition in the reference RFC 2828 [169]. However, DNP3 adds the specific restriction to the confidentiality of keys, as the main purpose of the standard is to authenticate messages that are not confidential.

Modification There is no explicit definition: we interpret this as an integrity requirement: adversaries must not be able to modify transmitted messages.

Perfect Forward Secrecy The general design text contains:

5.4.10 Perfect forward secrecy	(from IEEE 1815-2012 p.16)
This specification follows the security principle of perfect forward secrecy, as defined in IEC/TS 62351-2. If a session key is compromised, this mechanism only puts data from that particular session at risk, and does not permit an attacker to authenticate data in future sessions.	

Surprisingly, IEC/TS 62351-2 [108] does not mention the concept of (perfect) forward secrecy. However, the informal explanation suggests that the loss of some session keys should not affect authentication of future sessions with, presumably, different session keys.

Adversary Capabilities The standard states that communications might be performed over insecure channels, and this suggests the threat model includes adversaries that can manipulate or insert messages.

The standard additionally states that "if update keys are entered or stored on the device in an insecure fashion, the entire authentication mechanism is compromised" [110, p.21]. This suggests that some forms of compromise might be considered (e.g., of session keys), but not the full compromise (in which all stored data is compromised) of a party involved of a session.

3.3 Formal model of SAv5 in Tamarin

Our modelling and analysis of Secure Authentication v5 used the Tamarin security protocol verification tool [138], as described in Chapter 2.

3.3.1 Complexity of the Protocol

Each of the protocols within Secure Authentication v5 are individually straight forward; however, much more complexity becomes apparent when they interact. To give an indication of the state machines, see Figure 3.6 for a diagram showing the state transitions performed by the user. The system starts in state 0; each node is the state the user is in before it executes a rule along one of the outgoing edges. These edges are labelled with the name of the rule which the user executes during the transition into another state (these names are the same as in the Message Sequence Charts). This diagram demonstrates how multiple loops can occur in many different orders, with very little determined structure, and how little of the relevant state is represented by the standard's state machines. Each protocol can loop many times (below certain large thresholds), making the possible routes through the state machines and state-space very large and complex indeed. As there is stored data associated with each of these states, we do not get injective correspondence with the named states from the SAv5 specification.

Much of the complexity within Secure Authentication v5 comes from the transitions between different stateful sub-protocols as well as the multiple directions of the *Critical ASDU Authentication Protocol*. Each of these sub-protocols updates some part of agent state, and each of them rely on parts of state updated by other sub-protocols. For example, the ability for an outstation to receive an aggressive mode request depends on:

1. their pairing to a user, set at initialization and invariant across all sub-protocols;
2. their current state machine state, which must be set to `SecurityIdle` or `WaitForReply`. This status is invariant through the both the symmetric and asymmetric versions of the *Update Key Change Protocol* but modified by the other two sub-protocols;
3. their current session keys, set by the previously completed *Session Key Update Protocol*, and invariant across the other sub-protocols;

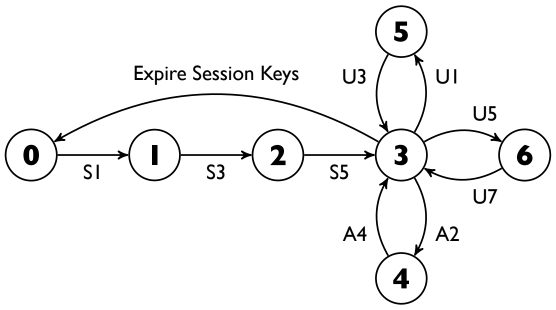


Figure 3.6: A simplified version of the user's state machine as defined in the standard, excluding error transitions and the monitoring direction of the *Critical ASDU Authentication Protocol*. Note that although many transitions occur from the same state, they are conditional on additional state that is not represented in the state machine as described by the standard.

4. the status of those session keys, which is invariant across all sub-protocols but can be changed at any time through a key expiry state transition;
5. the 'current' challenge from the user, modified when a non-aggressive mode request was successfully verified by the outstation, which is invariant in the aggressive mode protocol variant and all other sub-protocols;
6. and the number of aggressive mode requests received by the outstation since the current challenge was set, which is invariant in all other sub-protocols.

Each of these parts of state can be updated independently of each other, and each is invariant over several sub-protocols.

The state machine described in the protocol specification [110] captures very little of the protocol logic; the current state machine state is only one of the six dependencies listed above, and in fact is invariant for the outstation through both the *Update Key Change Protocol* and *Session Key Update Protocol* sub-protocols. An accurate TAMARIN model must include this much larger range of transitions, as well as their associated errors and timeouts; it is not possible to directly map the states and transitions in the specification to those in the model. This greatly increases the difficulty of interpreting the specification.

The state machines described in [110] (corresponding to the transitions discussed in Section 3.2.2) capture very little of the protocol logic, as the allowed transitions depend more on values in memory than on the current state machine 'state'. As an example, the outstation remains entirely in the named state "Security Idle" throughout the *Update Key Change Protocol*; however, the outstation can only respond to certain messages from the user dependent on data from previously sent or received terms. Our TAMARIN models include this much larger range of transitions, as well as their associated errors and timeouts.

3.3.1.1 Tamarin Example Rule

We have described the theory behind TAMARIN's modelling and solving in depth in Chapter 2, but we now give a specific example of a DNP3: SAv5 rule modelled in TAMARIN.

Example 3.3.1.1. The rule is an abstracted implementation of the A3 'Send Authentication Reply' message rule, in the control direction, from the *Critical ASDU Authentication Protocol* described in Section 3.2.2.2.

```
rule A3_C:
    let AC = < CSQ, USR, CD >
        AR = < CSQ, USR, hmac(<CSQ,AC,ASDU>, CDSK) >  in
    [ UserState(USR, OS, MDSK, CDSK, LastControlChallenge, [...],
                'SecurityIdle')
    , In(AC) ]
  --[ SentASDU(USR, OS, AR, 'NonAggressiveMode', 'ControlDirection')
    , UsingSessionKeys(CDSK,MDSK)
    ]->
    [ UserState(USR, OS, MDSK, CDSK, AC, [...], 'SecurityIdle')
    , Out(AR) ]
```

The A3 rule defines the behaviour when a user (identified by `USR`) receives a challenge in response to a critical ASDU sent to an outstation (here identified by `OS`). The challenge message `AC` contains a `CSQ` sequence number, the `USR` identifier, and some challenge data `CD`. The user generates a reply, `AR`, including an HMAC of the challenge message, `AC`, and the ASDU under the relevant session key.

The actions contain two labels to refer to later, one recording that the user `USR` sent an authenticated reply `AR` intended for the outstation `OS`, in the non-aggressive mode and in the control direction. The other records the session keys that the user had in their state at the time of the rule's execution.

The conclusions of the rule output an updated user state, in which the term `LastControlChallenge` used for aggressive mode is changed to the new challenge just received, as well as the message to the network `AR` defined above.

The full model and associated theorems are contained in the file `dnp3.m4`, which can be found at [71].

3.3.1.2 Modelling Choices, and Issues with the Specification

The finer details of SAv5's sub-protocols in [110] and [109] are very often unclear, underspecified, and open to interpretation. We now describe the larger issues we encountered, and how we chose to model them.

Challenge Sequence Numbers: These values are simple counters, used to distinguish between runs of a protocol. We encounter questions around when and where they should be stored and incremented, and what to do in case of rollover. The specification states that parties should keep one CSQ per direction, and *not* on a per-user basis [110, p.211-g]. It also implies that parties should keep count of the number of Authentication Replies and Aggressive Mode Requests it has sent since the last Authentication Challenge it has received, on a per-user basis [110, p.211-d]. The purpose of the CSQ is to match messages, and to ensure that replays are not possible [110, pp.207 & 211].

Instead of modelling precisely as described, we keep one CSQ per user, per direction (control and monitoring). If we do not do this, the universal CSQ values in a model must depend on *all* of the state machines running from the same station, which makes analysis infeasible.

Modelling CSQs in this manner is analagous to the specification: both keep a single value which allows the Challenger to check whether received messages contain the correct CSQ or not; the specification keeps a universal total *and a difference* on a per-user basis, we simply keep a per-user total; both interpretations require this incrementing value to be in Authentication Replies or Aggressive Mode Requests, to prevent replay attacks.

The specification has clear rules for when the *sender* of a CSQ should increment this value [110, p.211], but nothing about when a recipient should accept the value; it is not clear whether a received CSQ (e.g., in an Authentication Challenge) can be any value, whether it must be strictly increasing, or whether it must be precisely one higher than its last seen value. In our model, recipients of CSQ values check that they are strictly increasing.

Sequence Number Rollovers: CSQs and KSQs explicitly rollover to 0 after they reach $4,294,967,295$ ($2^{32} - 1$); what a recipient of a rolled-over CSQ does is not defined. If CSQs are not strictly increasing, this is not an issue. If not, the way rollovers are handled needs to be done so correctly, or this might allow (very slow) replay attacks. We avoid this issue by not modelling rollovers.

Challenge Data: These values are equivalent to the traditional idea of a nonce, used within the protocols (in most situations) to determine freshness of a protocol run. When parties should set and erase certain values, and how many historic values parties should save is not clear. Challenge data from previous messages is saved to enable alternate modes, e.g., saving the last Authentication Challenge (AC) in the (non-aggressive) *Critical ASDU Authentication Protocol* so that its Challenge Data can be used in Aggressive Mode Requests.

It is not clear when this should be stored, and when each party should erase its previous 'last sent challenge'. As an illustration: after the outstation (here the Challenger) has sent an AC, the user might not receive this AC message; if the user gets bored of waiting for an AC (which might never appear), it might send an Aggressive Mode Request with a critical ASDU. The user knows that it must construct this request with the last AC it received, which will not be the same as the last AC the outstation sent: should the outstation accept this Aggressive

Mode Request? Figure 7-28 of [110] implies it should, (if in the 'WaitingForReply' state) but what about after an invalid reply is received, or a message times out?

How to construct HMACs from the user's side is clear (both for the non-aggressive and aggressive modes of CAAP), but it is not so clear how to work out what construes a valid HMAC from the outstation's side.

We modelled this as follows: after sending an Authentication Challenge message, the outstation saves the challenge it is currently expecting to receive in an Authentication Reply. Upon timeout, other error, or successful reply, this challenge gets saved as the last sent challenge. This means an outstation can receive an aggressive mode request between the Authentication Challenge and a timeout, and still prevent it from being over-written.

Asymmetric Mode of the *Update Key Change Protocol*: The DNP3 specification says [110, p.753] "If the Key Change Method is asymmetric, the Update Key Data shall be encrypted using the outstation's public key". Unfortunately, it does not specify whether the Authority is the one encrypting it, or the User. For our interpretation, we can choose one of two paths: either, we assume the Authority sends two copies of UKC to the User, one encrypted with the User's public key, and the other encrypted with the Outstation's public key; the User then forwards on the second one to the Outstation. Alternatively, the Authority could send one UKC message to the User, encrypted with the User's public key. The User could then decrypt this message, and re-encrypt its contents under the Outstation's public key, before transmission.

As the communications between Authority and User are considered 'out of scope' for the DNP3 SAv5 spec, it does not make any comment on which of these two situations is the case. Secondly, the specification makes various comments about keeping the required bandwidth to a minimum. As such, we model it as the latter case, i.e., the User decrypts the received message, and re-encrypts under the Outstation's public key before forward transmission.

We address many of these issues by making recommendations for improvements to the specification in Section 3.6; this section addresses these and other issues encountered through our modelling, analysis, and understanding of best cryptographic practice.

3.4 Analysis and results

3.4.1 Modelling the Threat Model and Security Properties

Modelling Adversary Capabilities As described in Section 3.2.3, the standard assumes that communication channels are not secure, so we assume the worst: the adversary fully controls the network, i.e., it can drop and inject arbitrary messages, and eavesdrop all sent messages. As we describe in Chapter 2, this is known within symbolic verification as the network part of the Dolev-Yao attacker model [86].

Based on the general principle of perfect forward secrecy, we additionally provide the adversary with the ability to compromise some (but not all) keys. In particular, when considering authentication or confidentiality properties, we will allow the adversary to compromise all session keys except for the CDSK/MDSK used for this particular critical ASDU. As a result, our model also considers any attacks on the authentication property that are based on the compromise of (different) earlier session keys, as described in the standard.

Modelling the Security Properties We now revisit each of the properties defined in Section 3.2.3 and describe how we interpret them for modelling purposes, resulting in three properties called AUTH1, AUTH2, and CONF.

Spoofing: AUTH1 The main security goal of SAv5 seems to be to prevent spoofing, i.e., to ensure that all critical ASDUs originate from the intended parties. This is classically specified as an authentication property. However, there is no canonical notion of authentication; instead, there are many subtly different forms (see, e.g., [131]). In this particular case, we choose a form of agreement, i.e., if party A receives a critical ASDU, then this exact message was sent by some B who agrees on the message and some additional parameters. In particular, the additional parameters we include here are the mode ("aggressive" or "non-aggressive") and the direction ("control" or "monitoring").

One complication is that classical authentication properties link identities: if Alice receives a message, she associates the sender with an identity (say, Bob), and the authentication property then encodes that Bob sent the message. However, in the case of SAv5, there are not always clear identities for parties, e.g., outstations. Instead, pairs of users and outstations are effectively linked through their initial (pre-distributed) update keys. Thus, the best we can hope to prove is that upon receiving a message, apparently from someone that initially had update key k, then the message was indeed sent by someone whose initial update key was k.

We thus model the following (relatively weak) agreement property, which we refer to as **AUTH1**: if an outstation or a user receives an Authentication Reply or Aggressive Mode Request message m in a mode x (where x is either "aggressive" or "non-aggressive") in direction y (where y is "control" or "monitoring"), then this message m was sent in mode x for direction y by a party that had the same initial (pre-distributed) update key.

We consider the following adversary capabilities for this property: the adversary can compromise all session keys (CDSK or MDSK) except for the one used in the message m. This covers the "perfect forward secrecy" general principle. Additionally, we allow the adversary to compromise all update keys other than that used to assign the current session keys. Furthermore, if the current update key (used to assign the current session keys) was assigned in the symmetric mode of the *Update Key Change Protocol*, we allow the adversary to compromise all (asymmetric) private keys; if the current update key was assigned in the asymmetric mode, we allow the adversary to compromise the (symmetric) Authority Key.

If neither of these is the case, the current update key must be the initial, pre-distributed update key, and the *Update Key Change Protocol* will not yet have successfully run.

Replay: AUTH2 Classically, replay refers to multiplicity: if Bob apparently completes N sessions with Alice, then Alice in fact ran at least N sessions with Bob. Phrased differently, an adversary should not be able to complete more sessions with Bob than Alice actually ran with Bob. However, the definitions in the standard suggest that replay should be interpreted as a special case of masquerading (and thus spoofing), which uses previously transmitted messages. From this we infer that some form of multiplicity or recentness is intended to be part of the anti-spoofing guarantee. We encode this as AUTH2, which is strictly stronger than AUTH1.

Thus, **AUTH2** additionally models so-called *injective* authentication, which captures the classical notion of replay prevention. Informally, it states that for each received message, there is a unique message sent. Thus, an attack in which an adversary tricks Bob into receiving a message twice which Alice only sent once violates the property.

Eavesdropping: CONF Since the standard considers non-confidential ASDU messages, there is no clear confidentiality requirement. However, the authentication guarantees can only be satisfied against an active adversary if the relevant keys remain confidential. Hence, a subgoal is to require confidentiality of keys. This should in particular hold against weaker adversaries, such as eavesdroppers.

We note that the prevention of spoofing attacks (as per the first requirement) implies that all the relevant keys (Authority Key, Update Key, and MDSK or CDSK) are confidential with respect to eavesdroppers. If they are not, the active adversary can trivially use them to spoof a message. We can still model these confidentiality requirements separately. This is useful for protocols that do not satisfy the authentication guarantees directly.

If the user chooses, encrypts, and transmits a new Session Key (e.g., `CDSK_1`) it is important that the adversary does not learn it. However, it is equally important that the adversary cannot e.g., block the transmission of `CDSK_1`, impersonate the user, and transmit different, adversary-chosen keys (e.g., `CDSK_2`) to the outstation. In the second case, `CDSK_1` might still be secret, but the adversary can still issue 'authentic' commands to the outstation, HMAC'd with `CDSK_2`. Since there are different key types, **CONF** is modelled as a set of confidentiality properties, one of each type of key and each perspective (role).

Modification. As stated before, this is not defined in the standard, and we interpret it as an integrity requirement. As such, it will be covered by our authentication guarantees AUTH1 and AUTH2.

Perfect forward secrecy. As noted in Section 3.2.3, this general principle indicates an intended resilience against the compromise of other session keys, and is covered by our adversary capabilities for the three properties.

3.4.1.1 Properties in Tamarin

We now explore the full properties from our Tamarin analysis. As described in Chapter 2, with each lemma, Tamarin attempts to construct a counter-example to the stated property; if it cannot, (assuming termination) it concludes that the property is upheld by the protocol model.

First, we consider the confidentiality properties. This first property models the secrecy of the Update Keys:

```
lemma update_key_secrecy:
"(All id uk #i.
    not(Ex #r. UpdateKeyReveal( uk ) @ #r)
    & NewUpdateKey( id, uk, 'Initial', 'usb_stick' ) @ #i
    ==> not(Ex #j. K( uk ) @ #j)
  )
& (All id uk ak #i.
    not(Ex #r. AuthorityKeyReveal( ak ) @ #r )
            & not(Ex #r. UpdateKeyReveal( uk ) @ #r )
    & NewUpdateKey( id, uk, 'Symmetric', ak ) @ #i
    ==> not(Ex #j. K( uk ) @ #j)
  )
& (All id uk oprk uprk #i.
    not(Ex #r. OutstationPrivateKeyReveal( oprk ) @ #r)
    & not(Ex #r. UpdateKeyReveal( uk ) @ #r)
    & NewUpdateKey( id, uk, 'Asymmetric', < oprk, uprk > ) @ #i
    ==> not(Ex #j. K( uk ) @ #j)
)"
```

This lemma is a conjunction of three properties: each of these three sections must be true for the overall lemma to be true. The lemma specifies:

- If the initial update key itself is not explicitly revealed, this implies the adversary does not know it (`K(...)`), and,
- If an update key encrypted using symmetric cryptography (under 'ak') is not revealed directly, and the authority key 'ak' used to encrypt it also was not revealed, this implies the adversary does not know it, and finally,
- If an update key transported using asymmetric cryptography is not revealed directly, and the private key used (by the outstation) to decrypt it is not revealed, this implies the adversary does not know it.

As described, Tamarin attempts to construct a counter-example to this statement. If Tamarin is able to conclude that it cannot construct a trace which would violate the secrecy of the update key, it concludes that the model of the protocol upholds this property.

The second confidentiality property models the secrecy of the Session Keys.

```
lemma session_key_secrecy:
"(All id uk cdsk mdsk #i.
      not( Ex #r . UpdateKeyReveal( uk ) @ #r )
    & not( Ex #r . CDSKReveal( cdsk ) @ #r )
    & not( Ex #r . MDSKReveal( mdsk ) @ #r )
    & Sourced_SKs( id, uk, cdsk, mdsk, 'Initial', 'usb_stick' ) @ #i
    ==> not(( Ex #j . K( cdsk ) @ #j ) | ( Ex #j . K( mdsk ) @ #j ))
  )
& (All id uk ak cdsk mdsk #i.
      not(Ex #r. AuthorityKeyReveal( ak ) @ r )
    & not( Ex #r . UpdateKeyReveal( uk ) @ #r )
    & not( Ex #r . CDSKReveal( cdsk ) @ #r )
    & not( Ex #r . MDSKReveal( mdsk ) @ #r )
    & Sourced_SKs( id, uk, cdsk, mdsk, 'Symmetric', ak ) @ #i
    ==> not(( Ex #j . K( cdsk ) @ #j ) | ( Ex #j . K( mdsk ) @ #j ))
  )
& (All id uk cdsk mdsk oprk #i.
      not(Ex #r. OutstationPrivateKeyReveal( oprk ) @ r )
    & not( Ex #r . UpdateKeyReveal( uk ) @ #r )
    & not( Ex #r . CDSKReveal( cdsk ) @ #r )
    & not( Ex #r . MDSKReveal( mdsk ) @ #r )
    & Sourced_SKs( id, uk, cdsk, mdsk, 'Asymmetric', oprk ) @ #i
    ==> not(( Ex #j . K( cdsk ) @ #j ) | ( Ex #j . K( mdsk ) @ #j ))
)"
```

This lemma is again a conjunction of three properties; each of these three sections must be true for the overall lemma to be true. We describe each of the three parts of the *Session Key* secrecy lemma in turn:

- Firstly, for when the current *Update Key* was distributed via the 'Initial' distribution method (i.e., USB stick): if the outstation accepts the Session Keys stated a `Sourced_SKs` Action Fact, then, assuming that the Update Key and the new Session Keys themselves haven't been directly revealed to the adversary in this trace, this implies that there is no time point `#j` in this trace when the adversary knows either of the new session keys.
- Secondly, for when the current update key was distributed via the *Symmetric Update Key Change Protocol*: if the outstation accepts the new session keys named in the `Sourced_SKs` action fact, then, assuming that the Authority Key ('`ak`') used to transmit the current update key, the Update Key, and the Session Keys themselves weren't compromised by (or revealed to) the adversary in this trace, then this implies that there is no time point `#j` in this trace when the adversary knows either of the new session keys.
- Finally, for when the update key was distributed via the *Asymmetric Update Key Change Protocol*: if the outstation accepts the new session keys named in the `Sourced_SKs` action fact, then, assuming that the private key ('`oprk`') of the private / public key-pair used to transmit the current Update Key, the Update Key, and the Session Keys themselves weren't compromised by (or revealed to) the adversary in this trace, then

this implies that there is no time point `#j` in this trace when the adversary knows either of the new session keys.

We then prove AUTH1 and AUTH2 for both of these protocols, before proving it for the *Critical ASDU Authentication Protocol*. First, the agreement lemma for the *Update Key Change Protocol*:

```
lemma update_key_agreement:
"(All id id2 uk ak ak2 update_key_method #i #j.
         not(Ex #r. AuthorityKeyReveal( ak ) @ #r & #r < #i )
        & not(Ex #r. UpdateKeyReveal( uk ) @ #r & #r < #i )
        & Sourced_UpdateKey( id, uk, 'Symmetric', ak ) @ #i
        & NewUpdateKey( id2, uk, update_key_method, ak2 ) @ #j & #j < #i
        ==> ( id = id2 ) & ( ak = ak2 )
          & ( update_key_method = 'Symmetric' )
  )
& (All id id2 uk oprk oprk2 uprk uprk2 update_key_method #i #j.
         not(Ex #r. UserPrivateKeyReveal( uprk ) @ #r & #r < #i )
        & Sourced_UpdateKey( id, uk, 'Asymmetric',
                                          < oprk, uprk > ) @ #i
        & NewUpdateKey( id2, uk, update_key_method,
                                    < oprk2, uprk2 > ) @ #j & #j < #i
        ==> ( id = id2 ) & ( oprk = oprk2 )
          & ( uprk = uprk2 ) & ( update_key_method = 'Asymmetric' )
)"
```

As the initial UK_0(USR,O) is pre-distributed by USB key, we do not need to prove agreement on it (and its secrecy has already been proven by the lemma `update_key_secrecy`). This lemma models:

- For all '`Sourced_UpdateKey`' events for an update key encrypted using *symmetric* cryptography (under '`ak`'), where neither the authority key ('`ak`') used to encrypt it, or the update key itself ('`uk`') is revealed directly by the adversary, where there was a `NewUpdateKey` action fact (before the '`Sourced_UpdateKey`') with the same update key, they must agree on ID, mode of encryption used to transmit the update key, and the authority key used to encrypt it.

- For all '`Sourced_UpdateKey`' events for a new update key encrypted using *asymmetric* cryptography, and the secret signing key '`uprk`' is not revealed directly, where there was a `NewUpdateKey` action fact (before the '`Sourced_UpdateKey`') with the same update key, they must agree on ID, mode of encryption used to transmit the update key, and the user's private key. N.B. This property does NOT require secrecy of the new Update Key for agreement to hold, making it strictly stronger than the symmetric property above.

Note we have also proven (before this lemma, in the lemma `update_key_sourced`) that for each '`Sourced_UpdateKey`' action fact there exists at least one `NewUpdateKey` action fact which agrees on the same attributes. This second property (above) is now stronger, saying that for *all* pairs of `NewUpdateKey` and `Sourced_UpdateKey` with the same value '`uk`', they must agree on all other listed attributes as well.

We then prove agreement on the *Session Key Update Protocol*:

```
lemma skiup_agreement:
"(All id id2 uk uk2 cdsk mdsk type source type2 source2 #i #j.
      not( Ex #r . UpdateKeyReveal( uk ) @ #r & #r < #i)
    & ( not(type = 'Asymmetric')
      | not(Ex #r. OutstationPrivateKeyReveal(source) @ #r))
    & ( not(type = 'Symmetric')
      | not(Ex #r. AuthorityKeyReveal(source) @ #r))
    & Sourced_SKs( id, uk, cdsk, mdsk, type, source ) @ #i
    & NewSKs( id2, uk2, cdsk, mdsk, type2, source2 ) @ #j & #j < #i
    ==> ( id = id2 ) & ( uk = uk2 )
      & (type = type2) & (source = source2)
)"
```

This lemma models that for all traces where the update key used to transmit the session keys is not revealed, the session keys themselves are not revealed, and either the outstation's private key or the authority key (depending on the method by which the update key was originally encrypted) weren't revealed, then where there was a `Sourced_SKs` action fact and a `NewSKs` action fact before this with the same session keys ('`cdsk`' and '`mdsk`'), they must agree on ID, update key, method of update key encryption ('`type`'), and authority key used to encrypt the update key ('`source`').

```
lemma asdu_agreement_implies_mode_agreement:
" not(Ex ak #r. AuthorityKeyReveal( ak ) @ #r )
& not(Ex oprk #r. OutstationPrivateKeyReveal( oprk ) @ #r )
& not(Ex uprk #r. UserPrivateKeyReveal( uprk ) @ #r )
==>
( All linkid ar mode direction linkid2 mode2 direction2 #i #j.
    ( All cdsk mdsk uk type source.
        UsingSessKeys( cdsk, mdsk, uk, type, source ) @ #i
    ==>
        not( Ex #kr. UpdateKeyReveal( uk ) @ #kr & #kr < #i )
        & ( direction = 'control' ==>
            not( Ex #skr. CDSKReveal( cdsk ) @ skr & #skr < #i ) )
        & ( direction = 'monitor' ==>
            not( Ex #skr. MDSKReveal( mdsk ) @ skr & #skr < #i ) ) )
    & AuthASDU( linkid, ar, mode, direction ) @ #i
    & SentASDU( linkid2, ar, mode2, direction2 ) @ #j & #j < #i
    ==> ( mode = mode2 )
      & ( direction = direction2 )
      & ( linkid = linkid2 )
)"
```

This lemma only considers traces where there were no Authority Key Reveal actions, and no private keys revealed. Then, for all traces where an ASDU was authorised (in the rule containing the `AuthASDU` action fact), using the session keys named in `UsingSessKeys(...)` (distributed by the named update key '`uk`', which cannot have been revealed, and itself was distributed via the method in '`type`', with the keys in '`source`'), and where the session keys for the direction of the received message (i.e., '`control`' or '`monitor`') were not revealed, and where there was a `SentASDU` action fact with the same message '`ar`' before the `AuthASDU` action fact, then they these two action facts must agree on '`linkid`' (the internal ID of the communications channel), '`mode`' (aggressive or non-aggressive), and '`direction`' (control or monitor).

Informally, this models that all authorised ASDUs cannot have been originally transmitted over a different link, in a different direction, or in a different direction; this as yet makes no claim about injectivity (or whether it was replayed over the same link, in the same direction and mode). It is worth nothing that while this proves agreement, it does not yet prove injective agreement.

In the `asdu_aliveness` lemma we prove (in almost exactly the same way) that (with the same above conditions) wherever there is a `AuthASDU` action fact, there must exist at least one `SentASDU` action fact with the same message, `linkid`, `mode`, and `direction`.

Then, finally, we prove injective agreement in the `asdu_injective_agreement` lemma:

```
lemma asdu_injective_agreement:
"not( Ex ak #r. AuthorityKeyReveal(ak) @ #r )
& not(Ex oprk #r. OutstationPrivateKeyReveal( oprk ) @ #r )
& not(Ex uprk #r. UserPrivateKeyReveal( uprk ) @ #r )
==>
( All linkid ar mode direction #i #j.
    ( All cdsk mdsk uk type source.
        UsingSessKeys( cdsk, mdsk, uk, type, source ) @ #i
    ==>
        ( All uk #k. UpdateKeyUsedForSKs( linkid, uk,
                              cdsk, mdsk, type, source ) @ #k
            ==> not( Ex #kr. UpdateKeyReveal( uk ) @ #kr & #kr < #i ) )
        & ( direction = 'control' ==>
            not( Ex #skr. CDSKReveal( cdsk ) @ #skr & #skr < #i ) )
        & ( direction = 'monitor' ==>
            not( Ex #skr. MDSKReveal( mdsk ) @ #skr & #skr < #i ) ) )
    & AuthASDU( linkid, ar, mode, direction ) @ #i
    & SentASDU( linkid, ar, mode, direction ) @ #j & #j < #i
    ==> not( Ex #k. AuthASDU( linkid, ar, mode, direction ) @ #k
      & not( #k = #i ) )
)"
```

In this lemma, we prove (again, in almost exactly the same way) that with the same above conditions, whenever there is a `AuthASDU` action fact with matching `SentASDU` action fact and associated terms, then there does not exist another (different) `AuthASDU` action fact at any point in the trace with the same message, `linkid`, `mode`, and `direction`. We can be confident that any honestly transmitted ASDU will have an incremented counter (and challenge data), and so any actor receiving a message with these same terms can be confident it is a replay, and refuse to authorise it. Proving this series of lemmas now demonstrates that the protocol upholds this property of injective agreement on authorised ASDUs.

3.4.2 Analysis in Tamarin

As described in Chapter 2, Tamarin makes use of backwards reasoning, starting from trace constraints that correspond to the negation of the specified property, and building up further constraints from the possible solutions to an open proof goal. This has the invariant that all complete traces that fulfil the original constraints also fulfil at least one of the new sets of constraints. For example, if the current state contains a rule with an unsolved premise fact, then when Tamarin solves this premise it splits the current state into several states, each containing one of the possible conclusions which may have been the source of that fact.

For example, to prove that a particular property holds in all traces (such as "In all traces, X is preceded by Y"), Tamarin begins with the trace constraints from its negation ("There exists a trace in which X is not preceded by Y"). Goals are solved until either there is a case with no goals remaining, which is a completed trace and thus a counter-example to the property, or all possible states for this transition system contradict the constraints. In the latter case, this returns a proof that no trace of the transition system can satisfy the constraints of the negated property, and thus the property holds in all traces.

This backwards reasoning makes Tamarin very efficient in many protocols, but is ill-suited to a naïve model of the SAv5 protocol. The specification relies not only on shared state between each constituent sub-protocol, but also a shared state machine which dictates which transitions are allowable at particular times. Further, the majority of state transitions occur from and return to the same state, `Security Idle`.

The consequence is that the SAv5 model contains many unbounded loops. Naïvely, an attempt by Tamarin to solve a premise requiring the `Security Idle` state may find that many rules are potential sources for the preceding state, and attempt to solve each of these possibilities separately. Worse, many may introduce new unsolved premises that also require the `Security Idle` state, creating a loop. One can see how this might cause Tamarin to indefinitely unroll loops backwards if it cannot already conclude that the constraints cannot be met. Within Tamarin's framework, this issue can be countered by making use of its forward inductive reasoning and proving properties that resemble loop invariants. For example, Tamarin can prove inductively that each instance of a *Critical ASDU Authentication Protocol* rule of a particular role must have been preceded by an instance of the *Session Key Update Protocol* rule for the same role. After it proves such invariants inductively, Tamarin can use them in the backwards search to avoid unrolling loops and instead reason about their starting point immediately.

More technically, the key to analysing a protocol like this is to identify invariants over particular transitions and prioritize solving for the origin point of these terms as necessary. For example, an outstation running the *Critical ASDU Authentication Protocol* is making use of session keys that were set during the last *Session Key Update Protocol* (rule S4, as labelled in Figure 3.2) and are invariant in all other rules. We therefore add a premise to any rule making use of the session keys so that it directly relies on the current "session key invariant", represented by a persistent fact that is output when the session keys are changed, along with a fresh identifier so that it cannot unify to any other session key invariant. In solving the premises, we can prioritize finding the origin point of the current invariants, as the properties of the current protocol often depend only on the circumstances around the relevant invariants.

In the *Critical ASDU Authentication Protocol* example, the authentication properties depend on the properties of the last *Session Key Update* and the original pairing of the user to outstation, and in the Aggressive Mode, on the last generated challenge data. Each of these is included as an invariant. When proving that all traces have the AUTH1 property, this allows TAMARIN immediately to solve for the origin of the invariants, which adds constraints to, for example, where the session keys were generated and assigned.

As described, the key to analysing a protocol like this successfully is to identify invariants over certain transitions, and to prioritize solving for the source of these.
Invariants within the model include:

1. the authority key and relevant identifiers for both the user and outstation as assigned during the initial key distribution,
2. update key invariants for both the user and outstation,
3. session key invariants for both the user and outstation,
4. and the last challenge sent or received in each direction for both the user and the outstation.

Additionally, there is a 'last key status message' which is stored by the outstation in both the S2 and S4 rules. Although this is invariant through all other rules, it is only used in rules where it is also modified, so we can efficiently represent it with a linear fact consumed and output by those two rules. Finally, there are three 'keys to reveal' facts output whenever new keys are generated, which are used to model adversary compromise and represented with persistent facts. The combined invariant relations are shown in Figure 3.7.

Please see Milner's thesis [140] for more discussion and detail on the use of injective facts and handling invariants within TAMARIN.

3.4.2.1 Asymmetric Mode Analysis

Adding the Asymmetric mode of the *Update Key Change Protocol* to the overall protocol models had significant impact on the proof burden and time required to prove the required properties of the models. As we detail in Section 3.4.3, adding one extra sub-protocol (the asymmetric variant of the *Update Key Change Protocol*) causes nearly a 4.5-fold increase in the total CPU time required to prove these same properties.

The main cause of the increased CPU time is that all proofs now need to consider many more case distinctions and possible sources for obtaining messages. Since there is no

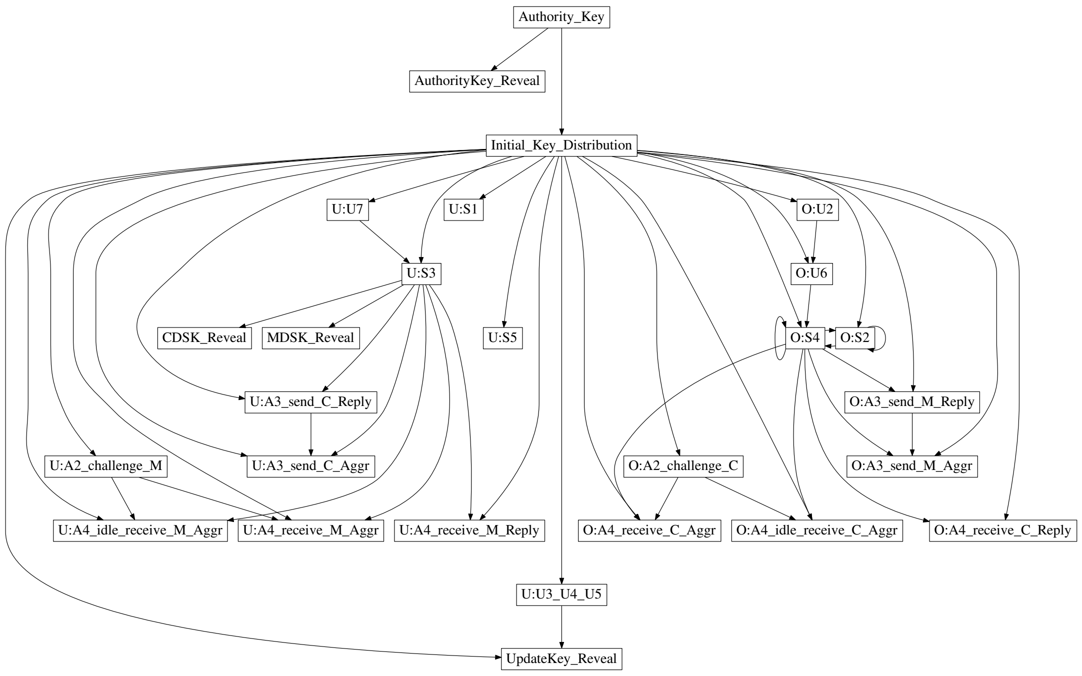


Figure 3.7: Protocol rules and the structure of loop invariants in the DNP3 model. Rules executed by the user and outstation are prefixed by 'U:' and 'O:' respectively, and invariants are represented by edges from the rules that set their value to the rules that use them. For example, the U:A3_send_C_Aggressive rule uses the CDSK invariant from the U:S3 rule, the challenge data invariant since the last U:A3_send_C_Reply rule, and the identifiers set up in the Initial_Key_Distribution rule.

compositional reasoning that can be directly applied here, and the additional sub-protocol can possibly precede most of the rule instances, the additional rules for the fourth sub-protocol exponentially increase the proof search space. Concretely, this meant we had to construct more complex invariants before TAMARIN could automatically prove the both the previous and newly added (i.e., specific to the asymmetric mode) security properties.

Rather than simply having to prove new properties just about the new asymmetric sub-protocol, we have additionally had to modify and update almost every lemma to take account of the fact that the asymmetric sub-protocol is now modelled. Proving properties about the secrecy of the update key now requires considering three possible methods of distribution rather than the previous two (i.e., USB Stick, Symmetric, and Asymmetric). Then, as session keys set by the *Session Key Update Protocol* could be compromised *indirectly* by compromise of the update key, or even compromise of the keys used by the *Update Key Change Protocol* (i.e., either the authority key or the user and/or outstation's private keys), properties about the *Session Key Update Protocol* must now also consider each possible method by which the update key could have been set or potentially compromised. This extra burden of proof on the origin of all keys used to encrypt or authenticate messages creates significantly more work and complexity for each lemma.

3.4.3 Results

Section 3.4.1 described how the specification requires the protocol be resilient to Spoofing, Modification, Replay, and Eavesdropping, and how these properties translated into more formal security properties AUTH1, AUTH2, and CONF. Our analysis in Tamarin has formally verified all three of these properties for our model of DNP3: Secure Authentication v5; in particular, they hold for any (unbounded) number of sessions and loop iterations. These results can be automatically verified by Tamarin from the model and properties in `dnp3.m4`, which can be found at [71]. On a modern PC (2.8 GHz Intel Core i7 from 2014 with 16GB RAM), these theorems in total prove in ~9m. We additionally proved several sanity checking properties, e.g., to show that our model correctly allows for expected behaviours.

It is worth noting the significant extra computational effort required to prove these properties when introducing even only one extra sub-protocol. The results from [70] proved the same properties on DNP3: SAv5 with only the symmetric mode of encryption available for the *Update Key Change Protocol*; introducing only one extra sub-protocol (the asymmetric variant of the *Update Key Change Protocol*) causes nearly a 4.5-fold increase in the total CPU time required to prove these same properties.

Table 3.1 details the time taken to achieve these results, and Table 3.2 details the results of each security property.

Model	**CPU Time**	**Wall clock time**
Symmetric only, from [70]	420.55s (~7m 00s)	123s (2m 03s)
Asymmetric *and* Symmetric	1846.53s (~30m 46s)	550s (9m 10s)

Table 3.1: Times for proving lemmas within the symmetric-only and combined models

Security Property	**Result**
AUTH1	verified
AUTH2	verified
CONF	verified

Table 3.2: Results of verification of security properties

As stated in the introduction, our results seemingly contradict an attack claimed in previous analysis; we will return to this in detail in Section 3.7.

3.5 Analysis under partial compromise

Having modelled and analysed the DNP3: SAv5 protocol suite under the standard's stated threat model, we now consider the implications of partial compromise, i.e., compromise of any of the system's components. We emphasise that consideration of partial compromise and our analysis here is strictly outside of IEEE 1815-2012's stated threat model.

As described, each of the outstation, user (at a master station), and authority all share the long term symmetric Authority Key, AK. While a system may implement the asymmetric mode of the *Update Key Change Protocol*, this is optional; systems implementing DNP3: SAv5 are required to support the symmetric mode of the *Update Key Change Protocol* regardless of whether they implement the asymmetric mode or not. As such, all three components of user, outstation, and authority must be in possession of the Authority Key.

No mention or requirement for trusted hardware or e.g., Hardware Security Modules is made in IEEE 1815-2012 [110], so we believe that physical compromise or gaining a 'root' on any one of these three components will give the adversary this Authority Key, and with it the ability to impersonate any of the three components in the system. This compromise does not require long-term access or control of any of the components; an adversary only needs a single time point of access to one of the components to learn the Authority Key. The standard notes that changing the Authority Key is out of scope for any of the defined protocols.

This new knowledge will then allow the adversary to invoke the *Update Key Change Protocol*, either receiving (if choosing to impersonate the user or outstation) or setting (if choosing to impersonate the authority) a new update key. With knowledge of this new update key, the adversary will in turn be able to receive or set a new session key through the *Session Key Update Protocol*; they will then successfully be able to authenticate any critical ASDUs which they choose to send to either user or outstation.

Partial Compromise Results: Compromise of any one of the system's three components therefore violates all three properties of AUTH1, AUTH2, and CONF.

Additionally, the standard implies that this Authority Key is universal across any particular network of users and outstations, and not different per triple of $\langle \mathit{outstation}, \mathit{user}, \mathit{authority} \rangle$.[2] This means that compromise of *any* component in a particular power grid network (e.g., a random, poorly guarded outstation in the middle of nowhere) could potentially lead to complete compromise of that whole power grid, and not just of the direct components with whom that outstation directly communicates. Alternatively, the memory of the computer(s) within a decommissioned outstation might not be securely deleted before being sold or disposed of publicly: in this case, the power grid operator would not even have evidence

[2]Or even just per user—outstation pair, as there should only be one authority per network.

of a physical break-in as an indication of compromise of the hard-coded Authority Key, and therefore the potentially complete compromise of their network.

This is clearly a design failing: while introducing individual hard-coded 'authority keys' per $\langle$*outstation*, *user*, *authority*$\rangle$ triple would both create a small amount of extra administrative overhead, and has many other issues, it would at least significantly reduce the impact of compromise of any single outstation or user within a power grid.

3.6 Recommendations

While our analysis was succesful in showing that the main properties hold under the standard's threat model, it also naturally leads to several recommendations. To aid clarity of implementation, to avoid possible misinterpretation, and to allow the protocol to meet stronger security guarantees, we propose the following changes to future versions of the specification.

Recommendations Based upon Modelling and Analysis:

- Update Key Change messages (`g120v13`) should contain a clear indication of intended recipient (i.e., outstation ID). This would allow for a stronger authentication property that only relies on the secrecy of the Authority key, not additionally on the secrecy of the new update key.

  In the *Update Key Change Protocol*, the Update Key Change object (`g120v13`) contains the KSQ, User Name, update key, and outstation Challenge Data, but not an outstation identifier (in contrast to the asymmetric version in `g120v14`). Thus, an outstation cannot ensure the Authority agrees on the outstation identity when receiving a newly encrypted update key. It is only through the HMAC in the Update Key Confirmation message (`g120v15`) that the outstation can authenticate the destination of the update key, but this HMAC is computed under that same new update key being distributed. Concretely, there is potential to attack the *Update Key Change Protocol* without knowledge of the Authority's key using only knowledge of the new update key. The adversary can present a challenge from outstation A to the user as if it were from outstation B, receive an Update Key Change object intended for outstation B encrypted under the Authority key, and re-compute the Update Key Confirmation message so that it is incorrectly accepted by outstation A. This potential 'attack' (under a very specific out-of-threat model scenario) was generated automatically by TAMARIN.

  This has only minor impact, as the update keys are assumed to be secret, and the attack requires two outstations to be running the *Update Key Change Protocol* with the same user concurrently. Nonetheless, it implies achieving agreement on a new update key requires a weaker adversary than is strictly necessary.

We discovered this during the analysis phase, as we attempted to prove a range of additional, stronger properties not explicitly required by the desired security properties or threat model. This included e.g., the above described authentication properties where the adversary could compromise the secrecy of the new update key; this property is violated in this scenario, but the simple modification to the protocol as described would strengthen the protocol significantly, precluding this attack.

- The specification must clarify the use of Challenge Sequence Numbers:
  - It is not clear whether CSQ values (per direction) should be kept on a per Master-Outstation pair basis, or whether each device should keep one universal CSQ value (per direction).
  - The specification must clarify whether recipients of CSQ values from the network (whether Responder or Challenger) should expect CSQ values to be strictly increasing. The sender's behaviour (whether in an Authentication Challenge, Authentication Reply, or Aggressive Mode Request) is clear, but it is not clear under which conditions a device should accept a CSQ as valid from another party. If CSQ values are not required to be strictly increasing, then replay attacks of Aggressive Mode Requests become possible.
  - Further discussion and reasoning about the use of CSQs may be found in Section 3.3.1.2. These issues were mainly discovered during the modelling phase: formally modelling a protocol forces the modeller to be completely explicit about the protocol's possible behaviours, and the English-language descriptions within the standard of how Challenge Sequence Numbers are to be used (and when they are to be accepted) is not clear enough to answer the above queries. During the analysis phase we were able to generate attack traces in Tamarin for the Aggressive Mode by removing the requirement for strictly increasing CSQ values.

Recommendations Based upon Best Cryptographic Practice:
These recommendations do not necessarily follow directly from formal Tamarin analysis and results, but are based upon our greater understanding of the protocols as a result of performing the modelling and analysis, combined with our knowledge of best practice for cryptography and protocol design.

- The specification should strongly recommend (or even require) that devices support asymmetric authenticated key exchange, rather just than symmetric key-transport with an optional asymmetric key-transport mode for the *Update Key Change Protocol*. This should be recommended for **both** the *Update Key Change* and *Session Key Update* Protocols, and completely disabling the symmetric modes of these sub-protocols should be permitted. Use of Elliptic Curve Cryptography (ECC) would allow stations to benefit from the added security of asymmetric cryptography, without significantly increasing the total amount of data transmitted. Asymmetric cryptography crucially only requires each private key to be in one location, and ECC is viable on low-power devices [103].

- The specification should deprecate the universal Authority Key per network in the symmetric mode of the *Update Key Change Protocol*, instead implementing at minimum a unique symmetric Authority Key per outstation.
- Deprecate HMAC-SHA-1. The SHA-1 algorithm is dangerously weak, and a collision has been found [172]. HMAC-SHA-256 should be required at minimum.
- Within both the symmetric and asymmetric modes, the protocol should perform some form of key-exchange (incorporating randomness from all involved components), rather than (a)symmetric key-transport [54]. This would significantly reduce the protocol's dependence on the raw output of any one CSPRNG [45].

Other Recommendations:

These recommendations are based upon our greater understanding of the engineering aspects of the protocols as a result of reading the standards in depth, and performing the modelling and analysis. We then combined this with our knowledge of best practice for protocol engineering and implementation.

- The standard must clarify how recipients of messages should parse them, and the standard must clearly and precisely state how recipients should calculate HMACs (e.g., to compare to received Authentication Replies and Aggressive Mode Requests). This must clarify which Sequence Numbers (for both Challenges and Key Changes) should be valid under which conditions, and which Challenge Data should be valid in which situations.
- The standard must clearly state when various data should be kept until (e.g., Challenge Data), when it should be overwritten, and how many previous instances of this data should be kept per User-Outstation pair.

Communication with IEEE, National Grid

During the modelling and analysis phases, we attempted to communicate with the standard's authors (via the IEEE) to learn more about the design decisions, and how the standard would be used in practice. We also attempted to contact potentially relevant users of the standard at National Grid (via NCSC), but got no response from either National Grid or IEEE. While the purpose of our research has always been focussed on the correctness of the *standard* rather than any particular implementation, it would have been beneficial to us to have a real, working implementation of the Secure Authentication v5 protocol suite, so that we could confirm (or reject) our interpretations of the imprecision in the standard. Gaining access to multiple implementations of the standard would be especially beneficial for future analyses of the standard's security; this would allow us see whether our interpretations of underspecification were viewed in the same light by multiple implementers, and to see whether this crucial lack of detail allows different implementations to inter-operate successfully and securely in practice.

3.7 Claimed attacks against DNP3: SAv5

While we consider broader work related to DNP3 in Section 6.4, here we discuss in context some previous work which claims to have found attacks against DNP3: SAv5's *Critical ASDU Authentication Protocol*.

Amoah et al., 2014 and 2016 (in [26] and [24] respectively) use Colored Petri-Nets to model and analyse both the non-aggressive and aggressive modes of the *Critical ASDU Authentication* sub-protocol. They discover a denial of service attack in the non-aggressive mode [26], and a "replay attack" when the aggressive and non-aggressive modes are combined [24]. Both papers only consider the *Critical ASDU Authentication Protocol* in isolation.

According to [24, p.353], the attack works as follows: after a non-aggressive critical ASDU request (A1 in Figure 3.3), the attacker blocks the Authentication Challenge message (A2) to the user, and sends a new one with the same challenge data, *but with an artificially incremented CSQ*. The user creates an Authentication Reply (A3, containing an HMAC) with this incremented CSQ value, which the outstation now rejects (A4). The attacker then replays this Authentication Reply with the critical ASDU prepended, to match the format of an Aggressive Mode Request (without modifying the HMAC), which, they claim, the outstation will now accept: valid Aggresive Mode Requests should have both the same challenge data as the last sent Authentication Challenge message, and a CSQ value incremented for each request sent since that challenge. As the user never sent an Aggressive Mode Request (only a non-aggressive request), [24] claims this violates agreement.

This attack does not work, as an outstation will not accept a non-aggressive mode message replayed into the Aggressive Mode. Our reasoning is as follows: HMACs within an Aggressive Mode Request must be calculated over "The entire Application Layer fragment that this object is included in, including the Application Layer header, all objects preceding this one, and the object header and object prefix for this object" [110, p.742, Table A-9]. An Aggressive Mode HMAC must therefore include the "Object Header `g120v3` Authentication Aggressive Mode Request", and the "Object Header `g120v9` Authentication MAC"; these two object headers must both be included in the HMAC calculation [110, A.45.9, p.741]. In contrast, the calculation of an HMAC within an Authentication Reply message (`g120v2`) from a *non-aggressive mode request* contains no such Aggressive Mode objects or headers. Assuming the attacker cannot successfully modify the HMAC without access to the session key, an HMAC for an Aggressive Mode Request will never match one calculated from the non-aggressive mode, regardless of whether the CSQ values and challenge data match.

We modelled this 'attack' in the file `dnp3-aggressive-amoah-attack.spthy`. For this to succeed, we had to under-approximate the original model significantly compared to the specification. Notably, in this model, we had to remove anything from the specification stating or implying the mode in both HMACs, as well as removing checks on the relationship between the CSQ in the body of the Aggressive Mode Request, and the CSQ within the Authentication Challenge included in the HMAC [110, pp.211 & 742].

We conclude that this claimed attack is an artefact of a model that is too coarse, and is not possible in faithful implementations of the standard.

After the conference version of this work was accepted for publication, we contacted the authors of [24] (Amoah, Çamtepe, and Foo) with our discovery, asking for comment or clarification. Amoah and Foo both replied to our email confirming that they did not model the HMAC correctly, and that therefore "the previously reported replay attack identified on the non-aggressive to the aggressive mode of operation will not be possible".

3.8 Conclusions

In this chapter we have performed the most comprehensive symbolic modelling and analysis yet of the DNP3 Secure Authentication v5 protocol; this analysis has considered all of the constituent sub-protocols (both symmetric and asymmetric), including cross-protocol and cross-mode attacks. We make use of novel modelling techniques in Tamarin, by identifying invariants in DNP3's state transitions to cope with analysis of the protocol's inherent complexity, extensive state, and unbounded loops and sessions.

Our findings notably contradict claimed results by earlier analyses; in particular, our findings show that the attack claimed in [24] is not possible in the standard as defined. We then perform analysis of the protocol under partial compromise (explicitly disallowed by the standard's threat model), and discover that it is especially vulnerable to compromise of any single one of its constituent network components; we provide recommendations for how to mitigate this.

While our analysis naturally leads to a number of recommendations for improving future versions of DNP3, we conclude that the core protocol of the standard meets its stated security goals if implemented correctly, increasing confidence in this security-critical building block of power grids.

"It is up to the network implementation to provide the security. You can see this very clearly in GSM: even though an authentication protocol is defined, the specification doesn't mandate running it. A GSM system works just as well without authentication.
[. . .]
Thus, the specifications permit building a working and secure system, but don't guarantee it. This allows differentiation of manufacturers (some get it right, some don't). As all of the UEs security depends on the network operators, the user has to trust the network operator to buy the right equipment."
— Alf Zugenmaier (Vice-Chair of 3GPP SA3)

4 Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion

4.1 Introduction

The 5$^{\text{th}}$ Generation (5G) mobile networks and telecommunications standard is currently under development, and are nearly finalised. A crucial building block in this standard is the "5G Authentication and Key-Agreement" (5G-AKA) protocol. This protocol is developed by 3GPP, and is an evolution of the AKA variants used in 3G and 4G, and is used to authenticate and establish keys between the involved parties. These parties include the subscribers, the networks within close range (referred to as Serving Networks), and the subscribers' carriers (referred to as Home Networks). The security of all 5G communication therefore crucially relies on 5G-AKA.

Traditionally, security protocols standards were not developed in tandem with rigorous security analysis, leading to many vulnerabilities being found after deployment. More recently, there has been a positive trend in which rigorous scientific analysis has been part of the development process; most notably IETF's TLS 1.3 protocol [154], which has benefited from being developed in tandem with a range of analysis methods [147]. Given the extremely wide deployment of 5G in the near future, it seems prudent to perform state-of-the-art analysis for this standard as well.

We perform a fine-grained formal analysis of 5G's main authentication and key agreement protocol (5G-AKA), and provide the first models that explicitly consider all parties defined by the protocol specification. Our formal analysis reveals that the security of 5G-AKA critically relies on unstated assumptions on the inner workings of the underlying channels. In practice this means that following the 5G-AKA specification, a provider can easily and 'correctly' implement the standard insecurely, leaving the protocol vulnerable to a security-critical race condition. We then provide the first models and analysis considering component and channel compromise in 5G, the results of which further demonstrate the fragility and subtle trust assumptions of the 5G-AKA protocol.

We propose formally verified fixes to the encountered issues, and we are working with 3GPP to ensure that this race condition is mitigated.

Methodology

Our work aims to provide rigorous formal analysis, and to improve the security of the 5G-AKA standard. As with Chapter 3, our approach uses formal symbolic modeling with the Tamarin prover [138].

Several aspects of the 5G-AKA protocol complicate formal analysis. The first of these is the sheer complexity of the documentation in which it is specified, which spans hundreds of pages. The second aspect is the complexity of the protocol, which involves four parties, depends on sequence counters for its security, and complex channel assumptions. The third is the informal nature of the security requirements, which mean the modeler has to make complex assumptions on the basis of the possible use cases.

We closely model the 5G-AKA specification: in particular, we explicitly model all four main parties in the specification, in which Home Networks include a separate component for credential storage that may be implemented in e.g., a hardware security module. We explicitly model possible assumptions on the channels connecting these four parties. We then analyse the resulting system model with respect to a range of threat models, including compromised components and channels.

Chapter overview: We structure our work in three main parts. In the first part, we describe the protocol (Section 4.2), the threat model implied by the standard (Section 4.3), security properties (Section 4.4), and its formalisation (Section 4.6 and 4.7.)

In the second part, we formalise and model a basic threat model, and perform analysis in Sections 4.8 to 4.12. We then consider the modelling, analysis, results, and consequences of stronger threat models that involve channel and component compromise in Section 4.13.

Finally, in the third part, we discuss the potential implications of these results, we discuss our interaction with the 3GPP working groups and upcoming changes to the standard (TS 33.501 [8]) in Section 4.14, before concluding in Section 4.16.

4.2 The 5G-AKA protocol

The 5G-AKA protocol is the flagship "Authentication and Key-Agreement" protocol within the newly proposed 5G standard, and therefore is the core building block for the security guarantees of the standard. 5G-AKA has evolved from the EPS-AKA protocol as used by 4G/LTE [11]. The 5G-AKA protocol is specified within §6.1.3.2 of 3GPP Technical Specification 33.501 [8]; here we model version v0.7.0.

We distinguish between the "home network", e.g., the network that the user signed up with, and the "serving network", which is the actual network that the phone connects to. While the home network can be the same as the serving network, they are different in a roaming scenario.

The 5G-AKA protocol establishes authentication and key-agreement between a mobile device, a serving network, and the device's home network. The standard additionally specifies a credential repository, which resides within the home network. The protocol is therefore specified as a sequence of communications between four roles:

- **UE**: the 'User Equipment'.
  This can be for example mobile phones or USB 5G dongles. Each UE is uniquely identified by its SUbscription Permanent Identifier (**SUPI**). In 5G, the SUPI performs the same role as the 'IMSI' in pre-5G standards.

- **SEAF**: the 'Security Anchor Function'.
  In the roaming context, this is within the serving network, e.g., the network that the phone connects to in a remote location.

- **AUSF**: the 'Authentication Server Function'.
  This falls within the home network, e.g., the network of the phone's service provider.

- **ARPF**: the 'Authentication credential Repository and Processing Function'.
  This also falls within the home network, and may typically reside within a secure location, such as a Hardware Security Module.

The UE and ARPF alone share the user's long-term secret symmetric key, **K**. In 5G-AKA, the SUPI should never be exposed publicly; a 'SUbscribption Concealed Identifier' or **SUCI** is used to achieve this. The **SIDF**, or the Subscriber Identity De-concealing Function is used to decrypt a SUCI value into a SUPI: this functionality is co-located with the ARPF.

At the end of a successful run of the protocol, all parties should all share or at least be able to derive and agree upon an 'anchor key' $\mathbf{K_{SEAF}}$, from which session keys for communication between the mobile device and base station(s) within the local network are derived. The secrecy of the K_{SEAF} key is therefore crucial to ensure the security of subsequent operations and communications.

See Figure 4.1 for a diagram illustrating the parties and channels involved in the 5G-AKA protocol; in this example, a mobile phone user is roaming in Norway, communicating back to their home network in the UK.

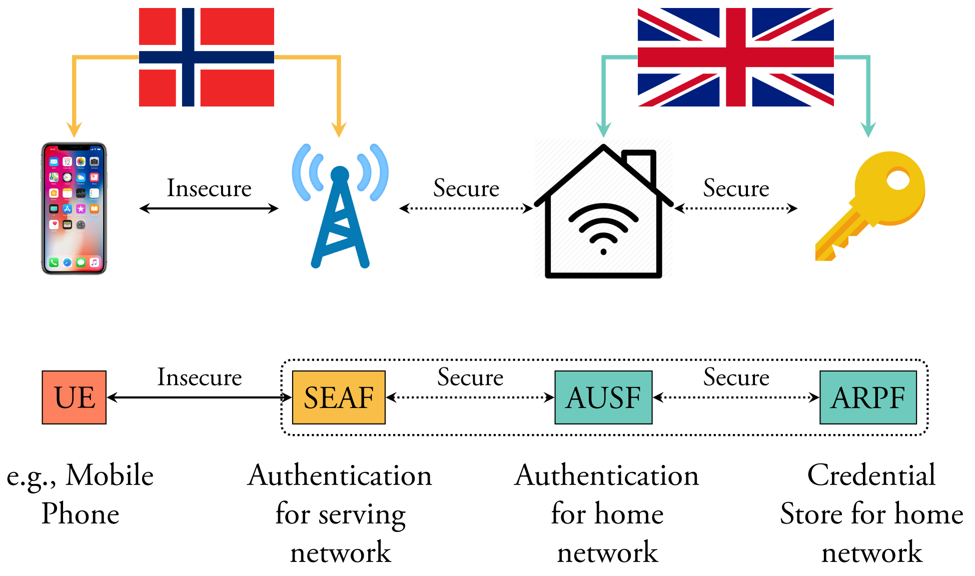

Insecure
Secure
Secure
UE
Insecure
SEAF
Secure
AUSF
Secure
ARPF
e.g., Mobile Phone
Authentication for serving network
Authentication for home network
Credential Store for home network

Figure 4.1: Parties and channels involved in the 5G-AKA protocol. Parties inside the dashed box are within the 5G core network; dashed channels are considered 'secure'. The left two parties (UE and SEAF) can be remote, or roaming; in this example, they are both in a separate country to that of the subscriber's home network – in this example, Norway. The right two parties are within the subscriber's home network, in this example, the UK. The icons (mobile phone, base station, home network, key) and country flags are for illustrative purposes only.

4.2.1 Normal execution of the 5G-AKA protocol

We now give a simplified overview of the 5G-AKA protocol execution, for illustrative purposes, as described in [8, §6.1.3.2]. See Figure 4.2 for a message sequence chart of the normal flow of the protocol, which we describe below. We omit the details of messages that are not needed to understand the vulnerabilities later.

1. The UE sends its ephemerally encrypted 'concealed ID' (SUCI, encrypted with a variant of the Elliptic Curve Integrated Encryption Scheme) and the name of its home network to the nearest SEAF. In the case that the SUPI is not concealed, it just sends the SUPI value unencrypted.

2. The SEAF sends a `5G-AIR` message containing the previous message and the name of the serving network to an AUSF in the relevant home network.
3. The AUSF then transmits this information in an '`Auth-Info Request`' message to the home network's ARPF.
4. The ARPF
   (a) Requests de-concealment of the SUCI into its respective SUPI from the Subscriber Identity De-concealing Function, or SIDF. The ARPF and SIDF are normally co-located.
   (b) The ARPF then retrieves the relevant K for this user, and
   (c) Generates a 128-bit random number 'RAND', a single AUTN value derived from RAND and the user's long-term key K, an 'Expected Response' value (XRES*), and a session key for the AUSF, K_{AUSF}. These are sent to the AUSF in an '`Auth-Info Response`' message. The 'Expected Response' value is a hash of the derived keys, RAND, and SNID; possession of it enables other parties (which may not know K) to verify that the user responded correctly.
5. The AUSF sends a `5G-AIA` message containing AUTN, a hash of the 'Expected Response' (i.e., HXRES*), the new 'anchor key' K_{SEAF} derived from K_{AUSF}, and the SUPI of the intended recipient.
6. The SEAF sends RAND and AUTN to the UE in an `Auth-Req` message.
7. The UE proves its identity, and implicitly, ownership of K, by responding to the SEAF with RES* (i.e., the 'Response') within an `Auth-Resp` message; the UE can now calculate the keys K_{AUSF} and K_{SEAF}.
8. The SEAF calculates the hash of RES* (i.e., HRES*) received from the UE, and checks that it matches with the hash of the '*Expected* Response', HXRES* value from the AUSF. If it matches, the SEAF considers the authentication to have been successful, and sends an Authentication Confirmation (`5G-AC`) message containing the user's SUPI, the serving network's ID, and the response, to the AUSF. The AUSF acknowledges this message from the SEAF by replying with a `5G-ACA` message (unspecified as of TS 33.501 v0.7.0).

4.3 Basic threat model: $\mathcal{A}_{Basic}$

The 5G-AKA documentation does not specify an explicit threat model. Section 5 of TS 33.501, "Security requirements and features" gives a mixture of threat models and desired security properties from the perspectives of the involved components, and we attempt to extract the most important points, discussing the threat model here, and the required security

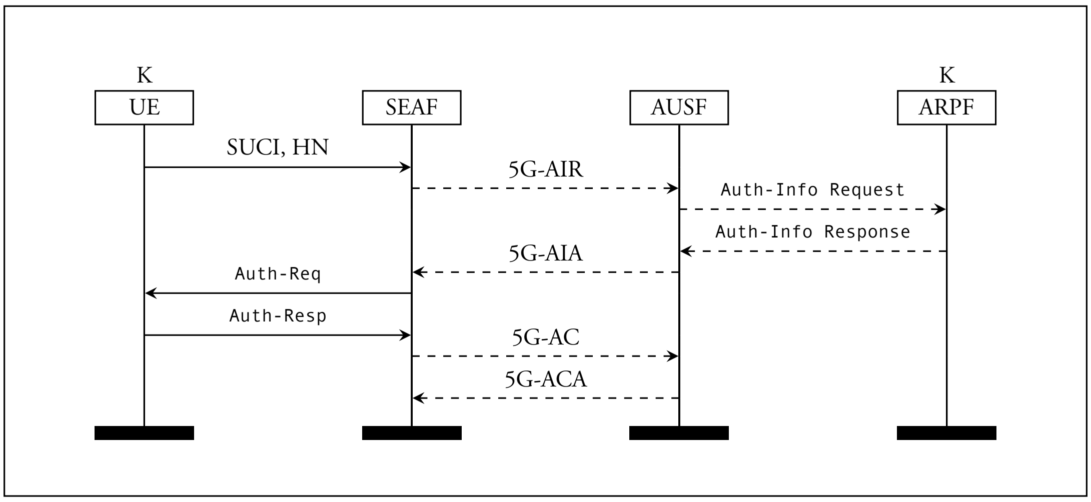


Figure 4.2: The normal flow of the 5G-AKA Protocol. Only the UE and ARPF know the user's long-term key, K. Dashed lines indicate messages sent over secure channels.

properties in Section 4.4. This is not an easy task, as security properties are often informally and minimally described in protocol standards. For transparency, we quote the original documentation where possible.

4.3.1 Channel threat model

The 5G-AKA protocol leverages the communications channels between the four involved parties. As shown in Figure 4.1, the channels specified by TS 33.501 for the 5G-AKA protocol are:

1. UE $\leftrightarrow$ SEAF
2. SEAF $\leftrightarrow$ AUSF
3. AUSF $\leftrightarrow$ ARPF

The standard specifies which of these connections should be secured in which way, and implicitly, the channels over which the adversary might have control. Concretely, the communications between SEAF, AUSF, and ARPF (i.e., channels 2 and 3) are within the "5G Core Network". In Figure 4.3 cite the precise properties required of "e2e core network interconnection" channels as described in TS 33.501 verbatim, which suggest that SEAF $\leftrightarrow$ AUSF and AUSF $\leftrightarrow$ ARPF form a type of secure channel. As a result, we consider the channels SEAF $\leftrightarrow$ AUSF and AUSF $\leftrightarrow$ ARPF as 'secure channels'; we describe their formal modelling later in Section 4.8.2.

Note that these channel properties do not explicitly require or guarantee delivery of messages, nor of ordering of the receipt of messages. We believe that these properties are analogous (or at least very close) to setting up and maintaining long-term IPSec, (D)TLS, or Diameter sessions over these channels, between the named parties. We return to the subtleties regarding the precise assumptions and formal modelling later in Section 4.8.2.

> **5.7.4: Requirements for e2e core network interconnection security (from TS 33.501 p.21)**
>
> A solution for e2e core network interconnection security shall satisfy the following requirements.
>
> - The solution shall provide confidentiality and/or integrity end-to-end between source and destination network for specific message elements identified in this specification. [...]
> - The destination network shall be able to determine the authenticity of the source network that sent the specific message elements protected [...]
> - The solution should be using standard security protocols.
> - The solution shall cover prevention of replay attacks.

Figure 4.3: Requirements for e2e core network interconnection security (from [8] p.21)

The standard does not specify any assumed security for the channel between UE and SEAF: in some sense, this is part of what 5G-AKA aims to provide. We therefore assume that channel between the UE and SEAF is considered insecure; we model it such that it is attacker-controlled, eavesdroppable, replayable, and the channel itself is without any cryptographic protections; individual messages may of course use their own cryptographic protections directly.

4.3.2 Component threat model

TS 33.501 v0.7.0 [8] does not explicitly describe whether it considers compromise of components within the system, and as such we conclude that compromise of any of the core network components (SEAF, AUSF, or ARPF) is not allowed in the basic threat model. The standard describes (cited verbatim in Figure 4.4) the protections required for the long-term key K within the USIM, and as such we assume that an adversary cannot compromise an honest user's key K. We do however assume that a persistent and capable adversary would be able to compromise the long-term key(s) of *other* USIMs, e.g., ones in its long-term possession.

We discuss the further implications of stronger channel properties (i.e., a weaker threat model) in Section 4.11, and then consider compromised channels, and separately compromised components (i.e., a stronger threat model) in Section 4.13.

4.4 Required security properties

Many academic papers have considered desirable security properties for authentication and key-agreement protocols. Some of the more commonly aspired to properties include: session key secrecy [117], long-term key secrecy [117], forward secrecy [139], post-compromise security [62], unknown key-share attack resistance [47], and a whole hierarchy of authentication properties [131]. This is by no means supposed to be a comprehensive list; instead it

5.1.4 Secure storage and processing of subscription credentials (from TS 33.501 p.16)

The following requirements apply for the storage and processing of the subscription credentials used to access the 5G network:

- The subscription credential(s) shall be integrity protected within the NG-UE using a tamper resistant secure hardware component.
- The long-term key(s) of the subscription credential(s) (e.g., K in EPS AKA) shall be confidentiality protected within the NG-UE using a tamper resistant secure hardware component.
- The long-term key(s) of the subscription credential(s) shall never be available in the clear outside of the tamper resistant secure hardware component. [...]

Figure 4.4: Secure storage and processing of subscription credentials (from [8] p.16)

6.1 Primary authentication and key-agreement (from TS 33.501 p.25)

The purpose of the primary authentication and key-agreement procedures is to enable mutual authentication between the UE and the network and provide keying material that can be used between the UE and network in subsequent security procedures. The keying material generated by the primary authentication and key-agreement procedure results in an anchor key called the K_{SEAF} provided by the AUSF of the home network to the SEAF of the serving network.

Keys for more than one security context can be derived from the K_{SEAF} without the need of a new authentication run. A concrete example of this is that an authentication run over a 3GPP access network can also provide keys to establish security between the UE and a N3IWF used in untrusted non-3GPP access.

The authentication run also results in an intermediate key called the K_{AUSF}. The K_{AUSF} may be left at the AUSF based on the home operator's policy on using such key.

Figure 4.5: Primary authentication and key-agreement (from [8] p.25)

is merely designed to illustrate a representative selection of reasonable, desirable properties for a modern key agreement protocol.

TS 33.501 v0.7.0 [8] details a number of security requirements on the various elements of the 5G ecosystem. We now detail the requirements directly affecting the 5G-AKA protocol, and the security properties which the standard states or implies 5G-AKA should uphold. TS 33.501 contains the text describing "security requirements and features for solutions", notably considering confidentiality and integrity requirements; we cite this in Figure 4.6. Section 5 of TS 33.501 further describes how we formalise these requirements on Authentication and Authorization; we include it verbatim in Figure 4.7.

4.4.1 Secrecy

TS 33.501 v0.7.0 [8] does not explicitly state a requirement for the secrecy of the session key K_{SEAF} (referred to as the "anchor key"); however possession of this key grants the bearer

5 Security requirements and features (from TS 33.501 p.15)

5.1.2 User data and signalling data confidentiality

5.1.2.1 Requirements on Support and Usage of Ciphering

- The UE shall support ciphering of user data between the UE and the gNB.
- The UE shall support ciphering of RRC and NAS-signalling. [...]
- Confidentiality protection of the user data between the UE and the gNB is optional to use.
- Confidentiality protection of the RRC-signalling, and NAS-signalling is optional to use.
- Confidentiality protection should be used whenever regulations permit.

5.1.3 User data and signalling data integrity

5.1.3.1 Requirements on support and usage of integrity protection

- The UE shall support integrity protection and replay protection of user data between the UE and the gNB.
- The UE shall support integrity protection and replay protection of RRC and NAS-signalling. [...]
- Integrity protection of the RRC-signalling, and NAS-signalling is mandatory to use, except in the following cases: [...]

Figure 4.6: Security requirements and features (from [8] p.15)

5.10 Authentication and Authorization (from TS 33.501 p.23)

The 5G system shall satisfy the following requirements.

Subscription authentication: The serving network shall authenticate the Subscription Permanent Identifier (SUPI) in the process of authentication and key-agreement between UE and network. [...]

Serving network authentication: The UE shall authenticate the serving network identifier through implicit key authentication. The meaning of 'implicit key authentication' here is that authentication is provided through the successful use of keys resulting from authentication and key-agreement in subsequent procedures. [...]

UE authorization: The serving network shall authorize the UE through the subscription profile obtained from the home network. UE authorization is based on the authenticated SUPI. [...]

Serving network authorization:

Serving network authorization by the home network: Assurance shall be provided to the UE that it is connected to a serving network that is authorized by the home network to provide services to the UE. This authorization is 'implicit' in the sense that it is implied by a successful authentication and key-agreement run. [...]

Access network authorization: Assurance shall be provided to the UE that it is connected to an access network that is authorized by the serving network to provide services to the UE. This authorization is 'implicit' in the sense that it is implied by a successful establishment of access network security. This access network authorization applies to all types of access networks.

Figure 4.7: Authentication and Authorization properties required by TS 33.501 (from [8] p.23)

access to a network on behalf of the UE which derived the key; Figure 4.5 alludes strongly to the importance of the K_{SEAF}, and its cryptographic parent, the K_{AUSF}. As such, we consider secrecy of session keys to be one of the primary goals of 5G-AKA, goal, even if unstated in the specification. We therefore interpret this information and the requirements from Figure 4.4 as the following **key secrecy** properties:

S1. The adversary must not be able to learn the long-term secret key K of an honest subscriber (stored within the UE/USIM).

S2. The adversary must not be able to learn an "anchor key" K_{SEAF} for an honest subscriber derived by 5G-AKA, or its cryptographic parent, K_{AUSF}.

4.4.2 Authentication and agreement

We note that the standard's interpretation of 'implicit key authentication' is not the same as normally considered within academic literature; the same description in line with academic literature would be closer to "The UE shall authenticate the serving network identifier through implicit key authentication. The meaning of 'implicit key authentication' here is that if the UE accepts a key for message encryption then only the intended peers can learn that key. Explicit authentication is provided through the successful use of keys resulting from authentication and key-agreement in subsequent procedures."

We intepret the requirements from TS 33.501, notably Figure 4.7, as the following **agreement** properties:

A1. The serving network and UE must agree on the identity of the UE.

A2. The UE and serving network must agree on the identity of the serving network.

A3. The home network and serving network must agree on the identity of the UE (and upon agreement, the home network confirms that the UE is a legitimate subscriber).

A4. The UE and home network must agree on the identity of the home network.

A5. The UE and home network must agree on the identity of the serving network (and this agreement implies that the serving network is authorised by the home network).

A6. The UE, serving network, and home network must agree on the anchor key, K_{SEAF}.

A7. The anchor key K_{SEAF} must not be replayable, i.e., the UE, home network, and serving network agree that the K_{SEAF} has only ever been accepted by one session.

Agreement property A7 is not directly stated as an explicit goal for K_{SEAF}, but replay protection for all data is required and indicated at multiple points, so we believe this is a reasonable goal for K_{SEAF} as well.

We describe how we interpret, model, and analyse these informally defined security requirements as more formal secrecy and authentication properties in Sections 4.7.1 and 4.7.2 respectively.

4.4.3 Additional security properties

While we could additionally consider forward secrecy and post-compromise security, 5G-AKA does not achieve these properties, and we are confident that the protocol in its present form will not (modulo trivial amendments) ever be able to achieve these remaining properties. For 5G-AKA to achieve, e.g., forward secrecy, the protocol would have to cease to rely solely upon a static shared symmetric secret key K between the UE and ARPF, representing a radical change in the design of the overall protocol. Due to legacy considerations we do not believe 3GPP would approve such major changes within 5G. It is our understanding that this was considered as a goal for 5G-AKA when TR 33.899 [5] was being prepared, but by the time of the first drafts of TS 33.501, this goal had been dropped. Similar, even greater changes would have to be introduced to achieve post-compromise security, and this is not stated or alluded to as a goal in either TR 33.899 or TS 33.501.

We do not formally analyse the secrecy of the SUPI when in concealed (SUCI) form beyond initial explorations (which found that the SUPI's secrecy was maintained in normal 5G-AKA execution and threat models) as this has been studied in more depth in concurrent work by Basin et al. in [40]; we discuss this work in more depth in Section 6.5. In the version of the standard which we analysed (TS 33.501 v0.7.0, [8]) the behaviour and SUCI de-concealment points within the 5G core network (i.e., revelation to the AUSF and SEAF) have not been determined.

4.5 Modelling and analysis roadmap

At this stage, it is worth laying out a roadmap for the remainder of this chapter. So far, we have informally described the 5G-AKA protocol, the basic threat model, and desired security properties as described in TS 33.501.

We model 5G-AKA as a four-party protocol in the Tamarin Prover. We model its channels, components, and interactions, describing our assumptions, modelling choices and limitations in Section 4.6. We give example rules from the protocol, discuss modelling limitations, considering e.g., XOR modelling and counter behaviour, and discuss our choice to model each component separately, creating a four-party protocol.

Then, in Section 4.7 we formalise the security properties, describing how we convert English-language properties from the standard (discussed in Section 4.4) into Tamarin lemmas. These lemmas are partially contingent on the specific threat model, but we describe per-threat-model modifications to these lemmas in the relevant analysis sections.

After modelling the 5G-AKA protocol's rules and security properties in Tamarin, the next step is to analyse the protocol against the threat model(s) and desired security properties: does the 5G-AKA protocol uphold the desired security properties in the stated threat model?

Actor	Partner	Threat Model	Property
UE	UE	$\mathcal{A}_{Stronger}$	Key Secrecy
SEAF	SEAF	$\mathcal{A}_{Basic}$	Agreement
AUSF	AUSF	$\mathcal{A}_{Weaker}$	
ARPF	ARPF		

Table 4.1: Query categories for 5G-AKA analysis

As we alluded to at the end of Section 4.3, we consider more than just the threat model as directly interpreted from TS 33.501. We consider three broad categories of threat model: $\mathcal{A}_{Basic}$, the basic threat model as described in TS 33.501 and Section 4.3, $\mathcal{A}_{Weaker}$, a group of weaker threat models, and $\mathcal{A}_{Stronger}$, a group of stronger threat models. This range of threat models in combination with the required security properties naturally directs our analysis, and we describe $\mathcal{A}_{Weaker}$ and $\mathcal{A}_{Stronger}$ in more detail in the coming sections.

Each question we wish to ask of the protocol model consists of three main elements: the viewpoint or actor(s) involved in the query, the threat model or adversarial power, and the property to be upheld. The actor "involved in the query" is the actor or component from whose perspective we make the query, and in the case of agreement properties, the other component with whom the actor seeks to agree upon a term.

Table 4.1 gives a high-level enumeration of these different groups. In reality, each of $\mathcal{A}_{Stronger}$ and $\mathcal{A}_{Weaker}$ encompasses multiple different threat models; Key Secrecy considers the secrecy of K, K_{AUSF}, and K_{SEAF}; and Agreement considers agreement (between the actor and partner) over 1) the identities of each of the components, and 2) the anchor key K_{SEAF}. Our queries of the protocol are therefore a series of Tamarin lemmas created overall by taking the cartesian product of these groups.

We proceed as follows, grouping the analysis phases by threat model:

- Firstly, starting in Section 4.7, we formalise the threat model $\mathcal{A}_{Basic}$, analysing the protocol model against the desired security properties to see which are upheld. This threat model does not allow any secure channel or component compromise. We present results and vulnerabilities under this threat model in Section 4.10.
- Secondly, as a result of the discovered issues and vulnerabilities, we consider whether there may be unstated or implicit assumptions on the underlying secure channels. We therefore explore the consequences of a strictly weaker threat model $\mathcal{A}_{Weaker}$, caused by stronger channel binding, in Section 4.11, giving analysis and results against the same desired security properties as before. We propose and analyse a range of fixes for the standard in Section 4.12, which then create the channel properties required to achieve 5G-AKA's desired security goals.

- Thirdly, as a multi-party, multi-stakeholder protocol, we consider what might happen if one or more of the involved parties behaved maliciously, and how much this protocol relies upon the security and honesty of each of its components and secure channels. We explore the consequences of adversary compromise of the secure channels and components, i.e., $\mathcal{A}_{Stronger}$ in Section 4.13. We give analysis and results against the same desired security properties for these scenarios.

4.6 Formal model of 5G-AKA in Tamarin

We formally model the four-party 5G-AKA protocol v0.7.0 [8] in the Tamarin Prover. These models have built upon and significantly diverged from initial models of 5G-AKA **v0.3.0** in the *three*-party setting, eventually leading to independent concurrent work: [40].

UMTS-AKA (3G) and LTE-AKA (4G) describe three components, rather than four. We believe it would have been reasonable to continue with the three-party configuration, amalgamating the AUSF and ARPF into a single 'home network' component, but in line with our specific desire to explore more fine-grained component-based models we opted to split them out into the two separate components as precisely as possible. We believe our results bear this decision out.

As in Chapter 3, our modelling and analysis of 5G-AKA takes advantage of the Tamarin security protocol verification tool [138]. We give a reminder of the symbolic modelling assumptions we made (in the context of 5G), and example 5G-AKA rules in Tamarin's syntax.

4.6.1 Example 5G-AKA rules

We give an example of a pair of protocol rules from within our 5G-AKA models in Figure 4.8. These two rules, `seaf_receive_attachReq` and `seaf_send_air` describe the SEAF receiving an attach request message from a UE over the insecure UE $\leftrightarrow$ SEAF channel (in this case containing the un-concealed SUPI and HN), and then constructing and transmitting a `5G-AIR` message to an AUSF within the 5G core network respectively.

This pair of rules for the SEAF uses both secure and insecure channels. The first rule receives an attach request message from a UE through the insecure, adversary-controlled, `In(…)` fact in its premise, containing the `SUPI` and `HN` terms. In the conclusion of the second rule, the SEAF takes advantage of the secure channel between itself and the AUSF, and transmits the `5G-AIR` message (which after 'let-binding' substitution will be `<'air',<SUPI,SNID,'3gpp_creds'>>`) using the 'Send Secure' `SndS(…)` construction described below, in Section 4.8.2. The next rule in the model (`ausf_receive_air`) will ordinarily receive the the message from the SEAF with a 'Receive Secure' `RcvS('seaf_ausf',SNID, AUSF,<'air',msg>)` fact.

```
rule seaf_receive_attachReq:
    [!SEAF(SNID),
     In(<SUPI, HN>),
     Fr(~SEAF_State_ID)]
  --[StartSeafSession(SNID),
     SEAF_SUPI(SNID, SUPI)]->
    [St_1_SEAF(~SEAF_State_ID, SNID, SUPI, HN)]

rule seaf_send_air:
    let
        msg = <SUPI, SNID, '3gpp_creds'>
    in
    [St_1_SEAF(~SEAF_State_ID, SNID, SUPI, ARPF), !AUSF(AUSF, ARPF)]
  --[Send_AIR_to(AUSF, ARPF)]->
    [St_2_SEAF(~SEAF_State_ID, SNID, SUPI, ARPF, AUSF),
     SndS(<'seaf_ausf','SEAF','AUSF'>,SNID,AUSF, <'air', msg>)]
```

Figure 4.8: A pair of example rules from the 5G-AKA models.

The facts `St_1_SEAF(…)` and `St_2_SEAF(…)` contain the state of the SEAF at each stage of the protocol. The Action Facts (`StartSeafSession(SNID)`, `SEAF_SUPI(SNID, SUPI)`, and `Send_AIR_to(AUSF, HN)`) do not ordinarily affect the execution of the protocol, and are used for analysis of traces.

The complete models are attached electronically, or can be found at [69].

4.6.2 Modelling choices

Standards documents are very often underspecified and open to interpretation. Protocols are perhaps unsurprisingly rarely designed with formal modelling and analysis in mind, and as such this presents various challenges and decisions to the modeller. We describe some of the decisions we made for modelling 5G-AKA within our tools, and their associated rationale in the following section.

4.6.2.1 Exclusive OR

The main release of TAMARIN at the time of modelling and analysis (v1.3.1) did not have support for analysis of the Exclusive OR (**XOR**, ⊕) operation on terms. It is possible to encode XOR's equational reasoning into TAMARIN models naïvely, but doing so quickly causes non-termination in many cases. More mature support for *analysis* of models which use XOR's full equational reasoning is under active development in TAMARIN, and was released in v1.4.0 on May 7th 2018, but as this was not available in released or stable form during our analysis phase, we chose not to take advantage of it for this work. Instead of modelling XOR's properties completely and accurately, for the sake of termination we over- or under-approximated XOR dependent on the stage of analysis; we under-approximate the adversary's knowledge from the XOR function for attack finding, and then over-approximate the adversary's knowledge from XOR for fix verification.

4.6.2.2 Counters, 'SQN'

The 5G-AKA protocol makes use of a counter or sequence number, SQN. The document defining 5G-AKA (TS 33.501 [8]) refers to TS 33.102 [6] for the definition and behaviour of this term; we cite this directly in Figure 4.9. CK and IK are the symmetric Confidentiality

6.3.2 Distribution of authentication data from HE to SN (from TS 33.102 p.22)

The HE [ARPF] has some flexibility in the management of sequence numbers, but some requirements need to be fulfilled by the mechanism used:

a) The generation mechanism shall allow a re-synchronisation procedure in the HE described in section 6.3.5.
b) In case the SQN exposes the identity and location of the user, the AK may be used as an anonymity key to conceal it.
c) The generation mechanism shall allow protection against wrap around of the counter SQN in the USIM. A method how to achieve this is given in informative Annex C.2.

NOTE: A wrap around of the counter SQN could lead to a repeated use of a key pair (CK, IK). This repeated key use could potentially be exploited by an attacker to compromise encryption or forge message authentication codes applied to data sent over the 3GPP-defined air interfaces.

Figure 4.9: Distribution of authentication data from HE to SN (from [6] p.22)

and Integrity Keys respectively, both generated from K_{SEAF}, by both the UE and SEAF. A repeated SQN would not lead to repeated CK/IK values *directly* as they are derived solely from K and RAND, but if a protocol run can be replayed with a previously seen RAND value (and the ARPF and UE will accept it), then the resultant CK/IK will be the same. The standard explicitly acknowledges that counters wrapping around could lead to repetition of a CK/IK key-pair, and gives a method for how to protect against this ("in informative Annex C.2" of [6], i.e., "Handling of sequence numbers in the USIM"). In light of this counter-measure, we model counters as strictly monotonically increasing, and with no possibility of wrapping around.

We do not consider deltas, or allowed increases between maximum previously seen counter values, we simply permit all SQN values which are strictly greater than the current maximum value. In this way, we are slightly more permissive than many implementations of the standard may be; we do not believe this affects our results. We acknowlegde that UEs and network operators are not required to implement the given counter-measure; we discuss the implications (or lack thereof) counters have overall on our discovered attacks and further analysis in Section 4.10.1.2.

4.6.3 Separation of components

5G-AKA is based closely upon the EPS-AKA protocol from LTE/4G [11]. There are a few notable differences between the protocols (such as the inclusion of concealed identities or

SUCIs, and the addition of the `5G-AC` and `5G-ACA` messages), but one important difference which we discuss here is the components formally described by the protocol standard.

In LTE/4G, the 'Authentication and key-agreement' protocol section of TS 33.401 [11, §6.1.1] describes just *three* components: the UE (joint with the USIM; for the purpose of network transmissions, the two are considered part of the same entity), the "MME" (the Mobile Management Entity), and the "HSS", or Home Subscriber Server. In 5G, (as earlier described) we have four named components involved in the 5G-AKA protocol, i.e., the UE, SEAF, AUSF, and ARPF. The SEAF and MME are broadly analagous in functionality between protocols; the equivalent HSS functionality in 5G-AKA is split between the AUSF and ARPF.

We model the AUSF and ARPF separately, where previous models of 5G-AKA continued to conjoin these two functions. This component-based separation of the AUSF and ARPF and their associated roles within the 5G-AKA protocol is key to our modelling and analysis, and the discovered vulnerabilities. When the AUSF and ARPF are modelled as a single component, the race condition (resulting in session confusion) described in Section 4.10.1 is not observable as a trace of the protocol model.

4.6.4 Modelling limitations

While we have attempted to model what we consider to be the most important elements of the 5G ecosystem (specifically in relation to the 5G-AKA protocol), there are many aspects which we do not model.

In addition to the limitations on e.g., XOR and counters described above, we also do not model the 'resync' mode of 5G-AKA: this is addressed by Basin et al. in [40], although including this in our models would be interesting future work. We do not model any parties or components apart from the four named components within 5G-AKA, and we do not consider the later derivation of keys within the key hierarchy after 5G-AKA has finished. We do not consider distinctions between e.g., the User Plane, Control Plane, Radio Resource Control, Access Stratum, and Non-Access Stratum, except where these make a difference to the 5G-AKA protocol's behaviour. We do not model the EAP-AKA′ protocol (described in [8, §6.1.3.1] and RFC 5448 [29]) as it and the very closely related EAP-AKA protocol have been studied in depth elsewhere [21,57,143]. That said, integrating a model of this protocol into our models of 5G-AKA would also be useful future work: analysing their composition would be interesting and non-trivial, as EAP-AKA′ makes use of the same long-term key K.

We do not model secrecy of the SUPI, whether concealed by SUCI or 5G-GUTI; we allude to the use (and honest de-concealment) of the SUCI in the protocol description, and we discuss how the use of SUCIs vs SUPIs does not affect the discovered vulnerabilities in Section 4.9. We performed initial exploratory modelling of the SUPI concealment, but did not find any issues directly, and so do not present these results formally. It is worth noting that the involved parties know significantly more about the identity of the involved UE in the un-concealed case by knowing its SUPI, than when it is concealed and it just knows an ephemeral SUCI. The vulnerabilities which we discuss later concern identity mis-binding, so we reason that if these mis-bindings can occur when all parties know the un-concealed SUPIs, they will most likely also occur when the honest core network parties know even less about the identities of the UEs with whom they are communicating. As such, we consider subscriber identity privacy out of scope for this research.

4.7 Formalisation of security properties

Having modelled the four-party version of 5G-AKA in Tamarin's multiset rewriting rules, we now model the range of desired security properties as (temporal) first-order logic formulae. These formulae are then evaluated over traces generated by the protocol model.

In the informal descriptions of Sections 4.3 and 4.4 we considered the basic threat model $\mathcal{A}_{Basic}$ first, and the required security properties second, as the threat modelling informs the desired security properties. Here, bearing the informal threat model descriptions in mind, we give formal definitions of the security properties first, as these remain broadly constant; the range of threat models is the varying factor in the coming analysis, so we consider that afterwards, starting in Section 4.8.

As detailed in Section 4.4, and as cited in Figures 4.5 and 4.7, TS 33.501 requires the following informally described security properties:

Key Secrecy

S1. The adversary must not be able to learn the long-term secret key K of an honest subscriber (stored within the UE/USIM).

S2. The adversary must not be able to learn an honest subscriber's "anchor key" derived by 5G-AKA, K_{SEAF}, or its cryptographic parent, K_{AUSF}.

Agreement

A1. The serving network and UE (subscriber) must agree on the identity of the UE.

A2. The UE and serving network must agree on the identity of the serving network.

A3. The home network and serving network must agree on the identity of the UE (and upon agreement, the home network confirms that the UE is a legitimate subscriber).

A4. The UE and home network must agree on the identity of the home network.

A5. The UE and home network must agree on the identity of the serving network (and this agreement implies that the serving network is authorised by the home network).

A6. The UE, serving network, and home network must agree on K_{SEAF}.

A7. The anchor key K_{SEAF} must not be replayable.

This leads us to believe that 5G-AKA should uphold the more traditionally defined properties of session key secrecy (which in this instance implies long-term key secrecy), non-injective agreement on the parties involved, and injective agreement on the session key, K_{SEAF}.

In this research we therefore consider session key secrecy, long-term key secrecy, non-injective agreement and injective agreement on a variety of terms. We believe that non-injective agreement on certain component identities in combination with session key-agreement encompasses unknown key-share attack resistance: an unknown key-share attack is one where an adversary convinces two honest parties into sharing a key, but crucially where at least one of them doesn't correctly know the identity of the other party [48]. We can preclude this category of attack if a protocol achieves mutual agreement on each of the components' identities, at the same time as achieving mutual agreement on the shared session key.

We now describe how we model these security properties. We start by giving formal descriptions of the secrecy and authentication properties in Sections 4.7.1 and 4.7.2 respectively. Then, in Section 4.8 we formalise the threat model $\mathcal{A}_{Basic}$, we consider the protocol under this threat model, evaluating which desired properties the protocol actually meets, giving results in Section 4.9.

4.7.1 Secrecy properties

We consider both session key secrecy (of K_{SEAF} or K_{AUSF}) and long-term key secrecy (of K) for 5G-AKA.

The session key secrecy lemmas state that for all traces where there was not a long-term key reveal action by the adversary for the UE/SUPI in question, the adversary never learns (or derives) the resultant session key, K_{SEAF} (or in the case of the ARPF, K_{AUSF}). As defined in TS 33.501 Annex A.6, $K_{SEAF} = \mathrm{KDFA}(K_{AUSF}, \mathrm{SNID})$; this is calculated at the AUSF. If the adversary is in possession of K_{AUSF}, they can derive K_{SEAF}, but not the other way round. The ARPF can clearly derive K_{SEAF}, but from the point of view of the ARPF we consider the secrecy of the K_{AUSF} to pin-point any compromise precisely: adversarial knowledge of K_{AUSF} implies knowledge of K_{SEAF}, but the reverse is not the case.

We consider session key secrecy from the point of view of each of the four parties involved, as this captures a broader range of properties than just secrecy for the UE. This means that we do not just consider the adversary being able to violate the secrecy of what is ostensibly the correct session key, but also the situation where an adversary can trick a party into accepting an incorrect session key which the adversary knows or can derive. This latter property also involves a violation of authentication, but we discuss those properties in the next section.

The session key secrecy properties in our models are of the following form:

```
lemma secrecy_UE:
"All UE t #i . Secret(<'UE', UE>, t) @ #i
    & not(Ex SUPI HN #r . RevealKforSUPI(SUPI) @ #r
          & Honest(<SUPI,HN>) @ #i )
    ==> not (Ex #j . K(t) @ #j )"
```

This roughly says: for all traces such that a `Secret` Action Fact is declared at the UE at time point `#i` for term `t` (the session key), and there is not an adversary key-reveal action for the same SUPI as in use at point `#i`, then there does not exist a time point `#j` such that the adversary learns or can derive that same term `t`. We discuss use of the '`RevealKforSUPI(…)`' action fact in more depth in Section 4.8. We consider session key secrecy properties of this form for all four of the parties UE, SEAF, AUSF, and ARPF.

The long-term key secrecy lemma simply says: for all long-term keys (each bound to a specific `SUPI`), where that key `Ki` (= K) was not revealed directly by the adversary, there is no time point such that the adversary learns the long-term key. This is modelled as follows:

```
lemma secrecy_Ki:
    " All SUPI Ki #i . LongTermKey(SUPI,Ki) @ #i
      & not(Ex #r . RevealKforSUPI(SUPI) @ #r)
        ==> not (Ex #j . K(Ki) @ #j)"
```

We believe these lemmas model the informal properties S1 and S2. We provide full results for session key secrecy and long-term key secrecy in Section 4.9.

4.7.2 Authentication properties

As 5G-AKA is a protocol considering four parties (each with different roles), it is not sufficient to consider the traditional two-party authentication properties just defined between the components in possession of the long-term secret key, i.e., the UE and ARPF.

The authentication properties described below systematically cover the range of pairwise authentication properties which we believe the 5G-AKA protocol could be expected to provide. For properties within the serving and home networks, we do not expect 5G-AKA itself to create confidentiality, integrity, or authentication guarantees between parties within the 5G core network. The standard explicitly describes the properties which must already be guaranteed by the underlying network connections, and we explore these guarantees in more detail in Sections 4.3.1 and 4.8.2.

To analyse authentication properties, we place Action Facts within protocol rules. The convention we use for ordering of terms within `Commit(…)` and `Running(…)` Action Facts is

```
ActionFactName(actor, <a,b,c,d>, t, <'nameOfActor', 'nameOfTerm'>)
```

where `ActionFactName` is either ‘`Commit`’ or ‘`Running`’, `actor` is the unique identity of the component in which the Action Fact occurs (UE, SEAF, AUSF, or ARPF), `<a,b,c,d>` is a vector containing all of the identities of the components the `actor` believes are involved (whether agreement over these is required in the claim or not), `t` is the ‘data’ term over which the parties seek to agree (e.g., the session key, K_{SEAF}), `'nameOfActor'` is a string containing the name of the actor (e.g., `'UE'`), and `'nameOfTerm'` is a string containing the name of the data term (e.g., `'K_SEAF'`).

We consider pairwise agreement properties from the points of view of each of the UE, serving network, and home network (‘`Commit`’ Action Facts). These properties are then considered in relation to each one of the four parties (UE, SEAF, AUSF, and ARPF), who generate the relevant ‘`Running`’ Action Facts. We do not consider agreement properties *from* the point of view of the ARPF on its own, because: firstly, the ARPF’s only role is to initiate the cryptographic section of the protocol (generating RAND, AUTN, XRES*, and K_{AUSF}); none of the messages **before** the `Auth-Info Request` message involve any keys, randomness, state, or other cryptographic elements; secondly, no messages are returned to the ARPF.

The agreement lemma *names* in `5G-AKA.m4` [69] follow the convention

```
lemma agreement_Commit_Running_Term
```

where ‘`Commit`’ is the name of the party who performs the ‘`Commit(…)`’ Action Fact (i.e., the claim is from their point of view), ‘`Running`’ is the name of the party who performs the ‘`Running(…)`’ Action Fact, and ‘`Term`’ is the term over which the lemma claims the parties should agree. For example, the lemma ‘`agreement_UE_SEAF_ARPF`’ indicates the lemma which tests whether the UE and SEAF agree on the identity of the ARPF. Each ‘`Commit(…)`’ Action Fact is placed in each party’s final rule, i.e., the point at which the protocol is finished for that party.

We can combine the results from these pairwise properties (agreeing on single terms) together to achieve the more traditional properties of e.g., non-injective or injective agreement over the identities involved in the protocol run *and* a term such as the session key, as described in [131]. Performing the analysis systematically in this manner (on a term by term basis) helps us to pin-point precisely which terms (if any) cause any violations of agreement.

For example, if all three of the individual properties ‘`agreement_UE_ARPF_ARPF`’ (UE and ARPF agree on the identity of the ARPF), ‘`agreement_UE_ARPF_UE`’ (UE and ARPF agree on the identity of the UE), and ‘`agreement_UE_ARPF_K_SEAF`’ (UE and ARPF agree on the term K_{SEAF}) were to hold true, this would imply the stronger, more traditional property from the point of view of the UE of non-injective agreement between the UE and ARPF on the term K_{SEAF}.

4.7.2.1 UE

From the informal authentication properties defined in Section 4.4, we believe the properties directly concerning the point of view of the UE are as follows:

A1. The serving network and UE must agree on the identity of the UE.
A2. The UE and serving network must agree on the identity of the serving network.
A4. The UE and home network must agree on the identity of the home network.
A5. The UE and home network must agree on the identity of the serving network
A6. The UE, serving network, and home network must agree on K_{SEAF}.
A7. The anchor key K_{SEAF} must not be replayable.

Achieving all of these properties would be similar to achieving the traditional property of injective agreement (as described in [131]) on the identities of the UE, serving network, and home network, in combination with the term K_{SEAF}.

In our analysis we explore the full range of properties from the point of view of the UE, considering agreement with the SEAF, AUSF, and ARPF on the identities of the parties involved, and the 'data' term, e.g., the session key K_{SEAF}. Modelled formally, these properties follow the pattern illustrated in the following example lemma:

```
lemma agreement_UE_SEAF_ARPF:
    "All a b c d t #i . (Commit(a,<a,b,c,d>,t,<'UE','K_SEAF'>) @ #i
    & not(Ex #r . RevealKforSUPI(a) @ #r & Honest(<a,d>) @ #i))
    ==>
        (Ex a2 b2 c2 t2 #j .
            Running(b2,<a2,b2,c2,d>,t2,<'SEAF','K_SEAF'>) @ #j )"
```

This example models agreement from the UE's point of view with a SEAF on the identity of the ARPF.

In more detail, this says: For all traces such that there was a '`Commit(...)`' Action Fact at the UE, where the UE believes the parties involved in the protocol are `a`, `b`, `c`, and `d` (instantiating as the unique IDs of the UE, SEAF, AUSF, and ARPF respectively), and the UE believes that the session key K_{SEAF} is term `t`, and there was not an adversary key reveal against the UE's long-term key K, then there **must** exist at least one '`Running(...)`' Action Fact from a SEAF which agrees with the UE on the identity of the ARPF. Note that this specific lemma does not require agreement on *any other terms:* e.g., the UE and SEAF involved in the specific `Commit` and `Running` Action Facts could completely disagree on the identity of the AUSF, or even on the identities of the two directly involved parties, the UE and SEAF. Proving this property true demonstrates **non-injective agreement** on just the named term, (in this case, the ARPF).

We then also consider **injectivity:** achieving injective agreement requires agreement on the same terms (and/or parties) as before, but now also requires that there must be precisely one `Commit(...)` Action Fact with the specified term. As all of the identities of the parties involved may reasonably be used in repeated protocols, the only terms over which we might expect to achieve injectivity are the 'data' terms, e.g., the session key, K_{SEAF}.

Injective agreement lemmas are modelled in the following way:

```
lemma agreement_UE_ARPF_K_SEAF_injective:
    "All a b c d t #i . (Commit(a,<a,b,c,d>,t,<'SUPI','K_SEAF'>) @ #i
    & not(Ex #r . RevealKforSUPI(a) @ #r & Honest(<a,d>) @ #i))
    ==>
    (Ex a2 b2 c2 d2 #j . Running(d2,<a2,b2,c2,d2>,t,<'ARPF','K_SEAF'>) @ #j
    & not (Ex a3 b3 c3 d3 #k .
      Commit(a3,<a3,b3,c3,d3>,t,<'SUPI','K_SEAF'>) @ #k
    & not (#k = #i)
    )
)"
```

This is identical to the format of the previous property, (this time requiring agreement between the UE and ARPF on the K_{SEAF}), but it now additionally requires that there must not exist another `Commit(…)` Action Fact agreeing on the same term `t` at a different time point `#k` (i.e., such that `#i` and `#k` are not the same event). (Events in traces are strictly ordered in Tamarin, therefore if two Action Facts occur at the same time point in a trace, they were from the same event or rule.)

We analyse this range of authentication properties for each party communicating (directly or indirectly) with the UE, and for each party, the terms over which they might meaningfully agree. We provide full results for this in Section 4.9.

4.7.2.2 Serving Network (SEAF)

From the informal authentication properties defined in Section 4.4, we believe the properties directly concerning the point of view of the serving network are as follows:

A1. The serving network and UE must agree on the identity of the UE.
A2. The UE and serving network must agree on the identity of the serving network.
A3. The home network and serving network must agree on the identity of the UE
A6. The UE, serving network, and home network must agree on K_{SEAF}.
A7. The anchor key K_{SEAF} must not be replayable.

The serving network shares privileged, authentic, and non-replayable access to the 5G core network through which it can communicate with the home network. Before the protocol run, the SEAF does not share any specific secrets directly relevant to the 5G-AKA protocol with any other parties, nor does it generate any randomness, or maintain any state beyond each run of the protocol. We might ordinarily expect that the strongest authentication property which could be achieved with a confidential channel would be non-injective agreement; however, as the secure channel between the AUSF and SEAF is explicitly non-replayable, we can potentially leverage this fact to achieve injective agreement on the parties involved and the session key.

This example lemma requires non-injective agreement from the point of view of the SEAF with an AUSF on just the value of the term K_{SEAF}:

```
lemma agreement_SEAF_AUSF_K_SEAF:
    "All a b c d t #i . (Commit(b,<a,b,c,d>,t,<'SEAF','K_SEAF'>) @ #i
    & not(Ex #r . RevealKforSUPI(a) @ #r & Honest(<a,d>) @ #i))
        ==>(Ex a2 b2 c2 d2 #j .
        Running(c2,<a2,b2,c2,d2>,t,<'AUSF','K_SEAF'>) @ #j )"
```

As with the UE, we analyse the full range of pairwise properties (non-injective and injective) from the point of view of the serving network to all other parties in the protocol, over the range of identities and terms. We present the results in Section 4.9.

4.7.2.3 Home Network (AUSF and ARPF)

From the informal authentication properties defined in Section 4.4, we believe the properties directly concerning the point of view of the home network are as follows:

A3. The home network and serving network must agree on the identity of the UE.
A4. The UE and home network must agree on the identity of the home network.
A5. The UE and home network must agree on the identity of the serving network
A6. The UE, serving network, and home network must agree on K_{SEAF}.
A7. The anchor key K_{SEAF} must not be replayable.

Achieving all of these properties would be similar to achieving the traditional property of injective agreement on the identities of the UE, serving network, and home network, in combination with the term K_{SEAF}, as described in [131].

While we separate the components of the home network into the AUSF and ARPF for the sake of modelling the protocol, we consider them to be much more closely related than e.g., the relationship between the AUSF and SEAF. The ARPF receives and sends only one pair of messages, and the contents of the received `Auth-Info Request` message only indicate that a party wants to start a protocol run; it does not contain any cryptographically generated or signed terms, or any randomness generated by the initiator.

As the ARPF does not participate in the protocol after sending the `Auth-Info Response` message, it cannot know (within the defined protocol) whether the UE responded to its challenge correctly or not. As the AUSF has sufficient information to determine the correctness of the response from the UE, and as the AUSF is part of the home network, we consider the AUSF and ARPF as a pair for the high level properties regarding authentication. This is why we consider the final group of authentication properties from the point of view of the "home network", rather than either one of the AUSF or ARPF.

This example lemma requires non-injective agreement from the point of view of the AUSF (within the home network) with a UE on just the identity of the SEAF:

```
lemma agreement_AUSF_UE_SEAF:
    "All a b c d t #i . (Commit(c,<a,b,c,d>,t,<'AUSF','K_SEAF'>) @ #i
    & not(Ex #r . RevealKforSUPI(a) @ #r & Honest(<a,d>) @ #i))
        ==>(Ex a2 c2 d2 t2 #j .
            Running(a2,<a2,b,c2,d2>,t2,<'SUPI','K_SEAF'>) @ #j )"
```

As with the UE and serving network, we analyse the full range of pairwise properties (non-injective and injective) from the point of view of the home network to all other parties in the protocol, over the range of identities and terms. We present the results in Section 4.9.

4.8 Formalisation of threat model: $\mathcal{A}_{Basic}$

We now consider how we formalised and modelled the adversary powers associated with the threat model $\mathcal{A}_{Basic}$, as described in Section 4.3. As this threat model does not allow compromise of components apart from *other* UEs (i.e., no SEAF, AUSF, or ARPF compromise is allowed), we only need to consider adversary key reveal (of other UEs' long term keys) and secure channel modelling.

4.8.1 Adversary key reveal

Distinct from any other adversary actions, network channel and component compromises described in later sections, we say that the adversary can compromise the long-term key K of UEs other than the 'honest' UE we are considering directly in a particular trace. The rule for this compromise is as follows:

```
rule reveal_LTKSym:
    [ !LTKSym(SUPI, ARPF, K) ]
  --[ RevealKforSUPI(SUPI) ]->
    [ Out(K) ]
```

When considering the security properties of protocols, we use lemmas written as guarded first-order logic formulae. We have already described these in detail in Section 4.7. Within these lemmas, we include a restriction on the allowed Action Facts within a trace, e.g.,

```
...
& not(Ex SUPI HN #r. RevealKforSUPI(SUPI) @ #r
      & Honest(<SUPI,HN>) @ #i )
...
```

This clause states that we do not allow the adversary to compromise the long-term key K for the specific SUPI named within the '`Honest(…) @ #i`' Action Fact. Time point `#i` is the same time point as the main Action Fact we consider in the specific lemma; normally '`Commit(...) @ #i`' – see Section 4.7.2. Note that the compromise of *any* other long-term K is allowed (i.e., those of other UEs).

4.8.2 Secure channel modelling

Within our 5G-AKA TAMARIN models we model the secure channels within the 5G Core Network in a number of ways. For the threat model $\mathcal{A}_{Basic}$ as described informally in Section 4.3 we use the standard secure channel abstraction, as used and described in [41]. This construction replaces TAMARIN's adversary-controlled Dolev-Yao-style channels with secure channels. This takes the `In(msg)` and `Out(msg)` facts, and replaces them with secure channels. These facts are similar to the form `SndS(A,B,msg)` and `RcvS(A,B,msg)`, sending the term `msg` from A to B, who are explicitly named parties. The adversary cannot learn or modify any terms in these facts directly from this channel. We augment the standard construction very simply by including a 'channel name', 'SendType', 'ReceiveType', and associated Action Fact for ease of later channel and component selection and analysis. In practice, the `channelname` term describes which of the two secure channels the instantiation of the rule considers, and looks like `'seaf_ausf'` or `'ausf_arpf'`, and the `SendType` and `ReceiveType` terms just contain one of the strings `'SUPI'`, `'SEAF'`, `'AUSF'`, or `'ARPF'`. The rules defining this secure channel construction are as follows:

```
rule send_secure:
    [SndS(<channelname,SendType,ReceiveType>,A,B,msg)]
  --[SendSecure(channelname,A,B,msg)]->
    [Sec(<channelname,SendType,ReceiveType>,A,B,msg)]

rule receive_secure:
    [Sec(<channelname,SendType,ReceiveType>,A,B,msg)]
  --[ReceiveSecure(channelname,A,B,m)]->
    [RcvS(<channelname,SendType,ReceiveType>,A,B,msg)]
```

This construction ensures that the adversary cannot read or modify the contents of a message (`msg`) sent over this secure channel; likewise, the sender (`A`) and recipient (`B`) of each message cannot be modified or spoofed by the adversary.

By TAMARIN's semantics, only rules with this fact in their conclusion can produce a fact `SndS(<channelname,SendType,ReceiveType>,A,B,msg)`, and only rules with the fact `RcvS(<channelname,SendType,ReceiveType>,A,B,msg)` in their premise can consume it (i.e., the Adversary cannot construct this itself). Assuming all of the protocol's rules are modelled and constructed correctly, i.e., all rules of the protocol honestly identify the sender and intended recipient, this will also guarantee the authenticity of sender and recipient. N.B. This construction itself says nothing about which session of the protocol the message was intended for, nor the order in which messages are delivered.

Each of the facts '`SndS(…)`', '`Sec(…)`', and '`RcvS(…)`' are linear (as against persistent) meaning they can only be consumed as a premise to a rule once. A message transmitted through this channel therefore cannot be replayed by the adversary; the adversary can only attempt to trigger the original rule which invoked `SndS(…)` again, but this rule's premises will have to be satisfied again before this can occur.

We believe this construction very closely matches the "e2e core network interconnection" channel requirements precisely as described in [8, §5.7.4] (cited in Section 4.3.1 of this document), and hence we use it to model channels 2 and 3 under $\mathcal{A}_{Basic}$, i.e., the channels between the SEAF and AUSF, and the AUSF and ARPF respectively. We consider what happens under different threat models when secure channels and components are compromised (and how we model this) separately, in Section 4.13.

4.9 Analysis and results: $\mathcal{A}_{Basic}$

We have described 5G-AKA's desired security properties informally and formally in Sections 4.4 and 4.7, and the threat model under which we evaluate these properties, $\mathcal{A}_{Basic}$. Our systematic analysis has allowed us to reach conclusions about which of these properties are upheld. We present our findings for secrecy in Table 4.2, and authentication properties for each of the UE, serving network, and home network below in Tables 4.3, 4.4, and 4.5 respectively. Rather than just presenting results for the informal properties S1–2, A1–7 (as per Section 4.4), we present the more systematic range of secrecy and agreement results on each actor and term.

Each authentication table considers security properties from the point of view of one party, i.e., the UE, the serving network, or the home network respectively. Each row then considers agreement with another named party in the protocol, and then the columns indicate on which identity or term this pair of parties agree (or not).

Where the result is a tick (✓), this indicates that the property was verified. Where the result is a cross (✗), this indicates that a counter-example trace was found, falsifying the property. 'SC' next to a cross indicates that the violation of the property was caused by 'Session Confusion', and 'SNID' next to a cross indicates that the violation of the property was caused by the fact that the SNID term is unauthenticated. (We discuss these different violations in Sections 4.10.1 and 4.10.2.) Where the result is a T (in yellow), this indicates that we have not yet been able to obtain a result, as Tamarin did not terminate within a reasonable period of time (at least 1 hour; most terminating results were achieved in at most a few minutes). These results can be automatically verified in Tamarin from the model and properties in `5G-AKA.m4` [69].

For example, the red cross (✗) in the first row and first column of Table 4.3 indicates that the UE and SEAF do not necessarily agree on the identity of the UE: there exists a trace where the UE completes the protocol, but in which there was never a `Running(…)` Action Fact at *any* SEAF agreeing with the original UE on that UE's unique ID, or SUPI.

Likewise, the green tick (✓) in the first row and fourth column of Table 4.3 indicates that the UE and SEAF do always agree on the identity of the ARPF: for all traces where the

Party	Term	Result
UE	K_{SEAF}	✓
SEAF	K_{SEAF}	✗ (SC)
AUSF	K_{SEAF}	✗ (SC)
ARPF	K_{AUSF}	✓
	K	✓

Table 4.2: Secrecy properties of 5G-AKA under threat model $\mathcal{A}_{Basic}$

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗ (SC)	✗ (SNID)	N/A	✓	✗ (SNID)	✗ (SNID)
AUSF	✗ (SC)	✗ (SC)	N/A	✓	✗ (SNID)	✗ (SNID)
ARPF	✓	✗ (SNID)	N/A	✓	✗ (SNID)	✗ (SNID)

Table 4.3: Authentication properties of 5G-AKA from the **UE's** point of view under threat model $\mathcal{A}_{Basic}$

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗ (SC)	✗ (SC)	N/A	✗ (SC)	✗ (SC)	✗ (SC)
AUSF	✓	✓	✓	✓	✓	T
ARPF	✗ (SC)	✗ (SC)	✓	✓	✗ (SC)	✗ (SC)

Table 4.4: Authentication properties of 5G-AKA from the **serving network's** point of view under threat model $\mathcal{A}_{Basic}$

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗ (SC)	✗ (SC)	N/A	✗ (SC)	✗ (SC)	✗ (SC)
SEAF	✓	✓	✓	✓	✓	T
ARPF	✗ (SC)	✗ (SC)	✓	✓	✗ (SC)	✗ (SC)

Table 4.5: Authentication properties of 5G-AKA from the **home network's** point of view under threat model $\mathcal{A}_{Basic}$

UE completes the protocol, there must have been a `Running(…)` Action Fact agreeing with the UE on (at minimum) the identity of the ARPF.

With regards to the AUSF column (especially in relation to the UE, and the results in Table 4.3): at no point is the UE ever necessarily aware of the distinction between the parties within the UE's home network, i.e., the split between the AUSF and ARPF. We claim that the UE only cares that it is talking to the correct home network overall (made up of the AUSF and ARPF), and should not mind which specific servers it talks to (indirectly) within the home network. We chose to model the ARPF's ID as being representative of what the UE considers to be the home network, as the ARPF is the party which shares a long-term secret (K) with the UE. As such, it is trivial to consider a run of the 5G-AKA protocol where one (valid, honest) AUSF partakes in the first half of the protocol (i.e., before the `Auth-Resp` message is sent by the UE), and a different (valid, honest) AUSF within the same home network attempts to complete the protocol, receiving (but not necessarily accepting) the `5G-AC` message from the SEAF. The protocol would have therefore finished from the point of view of the UE, but would not complete successfully from the point of view of the new AUSF (as it would not have access to the same HXRES* and XRES* values). Strictly, from the UE's point of view, the result is that these properties are violated. This is therefore an academic violation of agreement properties as encoded, but not a property which we consider 5G-AKA should uphold. As such, we mark this column's results *in relation to the UE* as "Not Applicable". We note that the other parties within the 5G core network could reasonably care about the precise identity of the AUSF, so we include the property and results for these other parties.

While we give the full results in Tables 4.2–4.5, we refer back to the informally described properties from Section 4.4. Here we stated secrecy properties S1 and S2, and the authentication properties between UE, serving network (SN), and home network (HN) in A1–A7; we consider which of these are upheld by 5G-AKA under $\mathcal{A}_{Basic}$:

Secrecy properties under $\mathcal{A}_{Basic}$:

S1.	Secrecy of an honest subscriber's long term key K:	✓
S2.	Secrecy of anchor keys K_{SEAF} and K_{AUSF}:	✗

Authentication properties under $\mathcal{A}_{Basic}$:

A1.	The SN and UE agree on the identity of the UE:	✗
A2.	The UE and SN agree on the identity of the SN:	✗
A3.	The HN and SN agree on the identity of the UE:	✓
A4.	The UE and HN agree on the identity of the HN:	✗
A5.	The UE and HN agree on the identity of the SN:	✗
A6.	The UE, SN, and HN agree on K_{SEAF}:	✗
A7.	The anchor key K_{SEAF} must not be replayable:	✗

4.10 Protocol vulnerabilities

As the results from Section 4.9 show, we encountered various secrecy and agreement violations against 5G-AKA. The attack which violates the secrecy of the term K_{SEAF} (Table 4.2) is of particular interest, and we discuss this in some detail in this section. We first give an informal overview of this violation before giving an in-depth description. After describing the secrecy violation in Section 4.10.1, we consider the authentication violations in Section 4.10.2.

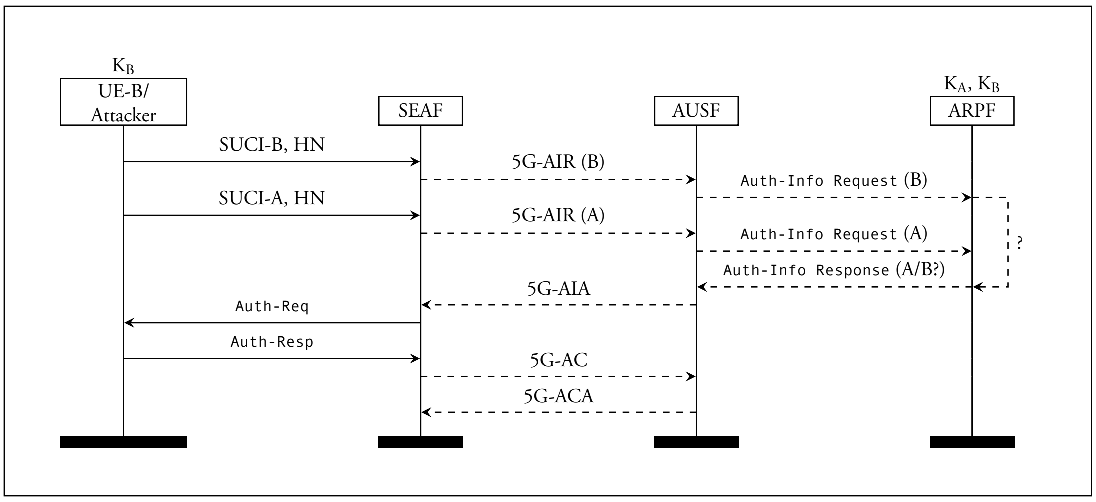


Figure 4.10: The **attack** flows of the 5G-AKA Protocol, secrecy violation caused by session confusion.

The attack proceeds in a similar manner to that presented by Tsay and Mjølsnes's attack against the three-party UMTS-AKA in [177], which is between the serving network and home network. In our four-party attack, we take advantage of a race condition over the AUSF ↔ ARPF channel entirely within the home network (neither additional channel or component is defined in UMTS-AKA or LTE-AKA), rather than the SEAF ↔ AUSF channel at the interface of the serving and home networks.

4.10.1 Secrecy violation

A malicious actor 'B' starts two 5G-AKA sessions with a local serving network at roughly the same time. One session is initiated by replaying an overheard SUCI (of the target, user 'A'), and the other session is with the malicious actor's own USIM and SUCI (for user 'B'). The sessions run in parallel, and result in a race condition; if this occurs, the AUSF will be unable to distinguish between the two responses containing the Authentication Vectors from the credential store (ARPF), and is liable to associate the wrong response (and resultant keys) with the wrong user. In the case that this occurs, the AUSF and SEAF will incorrectly believe that a set of Authentication Vectors and 'anchor key' were intended for user A (and derived from user A's long-term key K_A), when they were in fact derived from user B's long-term key K_B. As a result, the malicious user B will now be able to derive the anchor key, and use it to impersonate user A to the network. See Figure 4.10 for the message sequence chart of the attack.

4.10.1.1 What does the secrecy violation break?

We now give a more in-depth description of the secrecy violation.

The specific violated property which we consider now is the secrecy of the 'anchor key' K_{SEAF} (and its cryptographic parent, K_{AUSF}), *from the points of view of the SEAF and AUSF*. That is, at the end of the 5G-AKA protocol run:

- the SEAF, AUSF, and a user will have agreed and be in possession of a cryptographic anchor key, K_{SEAF},
- the SEAF and AUSF believe this key is for an honest and un-compromised user (in our example, user 'A' with 'SUPI-A' and 'SUCI-A'), and,
- both the SEAF and AUSF believe this key is secret from the attacker.

Thus, the protocol draft lacks a crucial containment property: an attacker that can compromise the long-term key of a user (e.g., 'B') will be able to impersonate *any* user (e.g., 'A') to the SEAF and the AUSF, because it knows the K_{SEAF} for a session that the SEAF and AUSF believe to be for 'A'.

4.10.1.2 Detailed attack scenario

The attack takes place in two (possibly temporally and even geographically) separate phases. In the first phase, the attacker eavesdrops and records a legitimate 'encrypted SUPI' (or 'Concealed SUPI'), also known as a SUCI. In the second phase, the main body of the attack takes place. Full message definitions can be found primarily in TS 33.501 [8] (some are in TS 33.102, [6]). N.B. This can attack occur even *more* easily if the SUPI is transmitted unconcealed, i.e., not encrypted into SUCI form.

Setup to the attack

1. A legitimate user 'A' with ID 'SUPI-A' is registered with its home network (HN). We are not interested in its long-term key K_A, as the attack does not require access to it. This honest user initialises the 5G-AKA protocol, sending the SUCI-A (user A's ephemerally encrypted SUPI) and 'HN' to a SEAF. The user might then complete the protocol as normal.

2. The attacker eavesdrops on the public radio transmissions from the previous step, and records the message containing SUCI-A and HN. (See Section 4.10.1.3 for a minor comment on the use of replayed SUCIs.)

3. The attacker purchases a legitimate USIM from the same home network as his intended victim; this has ID 'SUPI-B'. The attacker physically attacks and compromises the USIM, and extracts the long-term key K_B of this USIM in its possession.[1]

[1]After discussion with Vodafone, we believe this physical extraction of K_B may not even be necessary, although if this step is completed, from a practical point of view this gives the attacker even greater control over the timing and flow of messages.

Main phase of the attack The message sequence chart of the main phase of the attack can be found in Figure 4.10.

1. After the setup phase, the attacker (perhaps within the physical realm of a visiting network) initiates the 5G-AKA protocol by replaying to a SEAF the pre-recorded SUCI-A. This is the concealed ID of SUPI-A, but the attacker does not need to know whose ID it is, or its decrypted form. The attacker sends a message containing 'SUCI-A' and the name of the user's home network (HN) to a SEAF in a serving network (of name SNID).

2. The protocol proceeds as normal: the SEAF communicates with an AUSF in the specified home network by sending the '`5G-AIR`' message. This contains 'SUCI-A', and SNID (i.e., the ID of the serving network being used).

3. In parallel with start of the session for SUCI-A, the attacker starts a 5G-AKA session for the USIM it owns (SUPI-B) with the same home network, via the same serving network (and SEAF). The attacker is already in possession of SUPI-B's long-term key (K_B), as it has compromised the USIM in the setup phase. As before, it starts the 5G-AKA session by sending its own concealed ID ('SUCI-B') and the name of the home network (HN), to the same SEAF as in the other, parallel session. The SEAF clearly and correctly treats this as a separate session.

4. As before, the SEAF communicates with the AUSF in the home network by sending the '`5G-AIR`' message, containing 'SUCI-B' and SNID. The AUSF then sends the '`Auth-Info Request`' message to the home network's ARPF, as per the protocol.

5. The SIDF (co-located with the ARPF) de-conceals SUCI-B into SUPI-B, and the ARPF then responds by sending the '`Auth-Info Response`' message to the AUSF. This message contains terms derived from (the compromised) K_B, and the terms RAND, SQN, and SNID, but notably contains no reference to either the SUPI or the SUCI.

6. The '`Auth-Info Response`' message is received by the AUSF, but as this message does not have a SUPI or SUCI attached to it, the AUSF *does not know whether this message was for the session with 'SUCI-A/SUPI-A', or whether it was for the session with 'SUCI-B/SUPI-B'*. The AUSF can legitimately continue its session intended for 'A' with the '`Auth-Info Response`' message that was actually intended for the session with 'B'.

7. The AUSF then proceeds with the protocol, sending the `5G-AIA` message for 'SUPI-A' to the SEAF; this contains the anchor key K_{SEAF} that the ARPF generated for 'SUPI-B', but now the AUSF associates it with 'SUPI-A' (and as a result, so does the SEAF). As the attacker has compromised SUPI-B's long-term key K_B (and RAND and SQN are publicly transmitted during the protocol), the attacker can now construct the anchor key K_{SEAF} that the AUSF and SEAF now believe is the anchor key for 'SUPI-A'. That is, the attacker can derive the K_{SEAF} that the AUSF and SEAF believe to be for the (honest) 'SUPI-A' (and *not* 'SUPI-B' which the attacker has compromised). Hence, we have an attack.

It is worth noting that **Counters** or SQN values do not have any bearing on this attack, as only the ARPF and UE store what the 'correct' value of SQN is. The AUSF and SEAF do not use SQN directly in any calculations or derivations, and hence do not check whether it matches (or is greater than) stored values for a particular user. The attacker can accept Authentication Vectors (leading to an anchor key) generated with *any* SQN value, and is able to deduce directly which SQN value was used. In other words, while counters are used in the protocol to prevent some forms of replay, they are used in exactly the other direction as the one in which the attack proceeds.

4.10.1.3 Replaying ephemerally encrypted SUCIs

Our vulnerability (in the concealed setting) relies on the SIDF accepting a previously replayed SUCI value. We now discuss why we believe that adversary-replayed values will be accepted in this context.

The purpose of the UE sending an ephemerally encrypted SUCI is to conceal the SUPI, maintaining the privacy and unlinkability of the USIM's global ID. This attack does not violate that property. The UE uses an ephemeral public key (rather than static) to maintain its own unlinkability; hence the onus is on the UE to use a new ephemeral key with each run of the protocol to maintain this property. While the ARPF or SIDF maintaining a complete list of previously used ephemeral public keys is touched upon as a possible suggestion in the (formally withdrawn) Technical Report TR 33.899 [5], we find no evidence within TS 33.501 that this is required or even formally proposed, and therefore no evidence that an ARPF or SIDF will not accept a re-used ephemeral key or SUCI. TR 33.899 is a large study collecting various possible proposals from multiple authors across 3GPP, some of which were considered for inclusion within TS 33.501. This document has now been formally withdrawn as of August 2017, but gives good insight into different suggested proposals for 5G security measures from member companies of 3GPP.

In terms of fixing the protocol (and preventing the discovered attack), including a counter within the ECIES-encrypted SUCI (whose value is then checked against the previously highest seen value by the SIDF) happens to be sufficient for the concealed setting, but isn't sufficient overall: in the case that the null-scheme is used for SUPI/SUCI 'encryption', the attack then holds again. In the non-concealed case adding a counter isn't sufficient: there are no cryptographic protections at this stage, so the attacker can simply increment the last counter it observed. TS 33.501 (notably [8, §6.12.2]) does not mention a counter or any similar values within the SUCI.

If a counter is added to the SUCI, while this prevents the direct attack described in this document from succeeding (because replaying an overheard SUCI will no longer be sufficient), it means that an attacker merely has to learn a target user's un-concealed SUPI by some means to be able to impersonate them. It is our belief that the purpose of concealed IDs (and introduction of SUCIs rather than just SUPIs) is intended to maintain the privacy of the user's ID, and is not intended as a means to guarantee the overall authentication and key-agreement properties of the protocol.

4.10.2 Authentication violations

In addition to the secrecy violations described in Section 4.10.1, there are also multiple authentication violations, which are listed in Tables 4.3, 4.4, and 4.5. We detail the violation of these agreement properties here.

These property violations listed fall into three distinct categories: AUSF ID, authentication of serving network ID ('SNID'), and session confusion ('SC').

Firstly, as discussed in Section 4.9, we do not consider agreement on the AUSF ID to be a concern: we treat this as a purely academic attack. There is no direct reason within 5G-AKA for a UE to know precisely with which AUSF it is communicating, as long as it is communicating with the correct home network, and therefore we do not consider violations of this type any further.

Secondly, authentication of the serving network ID, or SNID. Within 5G-AKA, this is a genuine violation of agreement over the identity of the SNID, as the UE never learns this term in an authenticated message. This then causes agreement to fail on the term K_{SEAF}, as this term is derived by the UE from a series of terms including SNID; the ARPF, AUSF, and SEAF will derive an anchor key K_{SEAF} on which the UE will not agree if the adversary has injected an arbitrary SNID into the UE's (unauthenticated) serving network discovery phase.

We do not claim to be the original discoverers of this particular violation: this was originally discovered in concurrent work by Basin et al. [40]. These models now additionally consider sections and properties which we do not, such as 'resync' and more in-depth modelling of XOR; see Section 4.6.4. Our more fine-grained models and systematic analysis further confirms the existence of this violation of agreement on the SNID, and that the

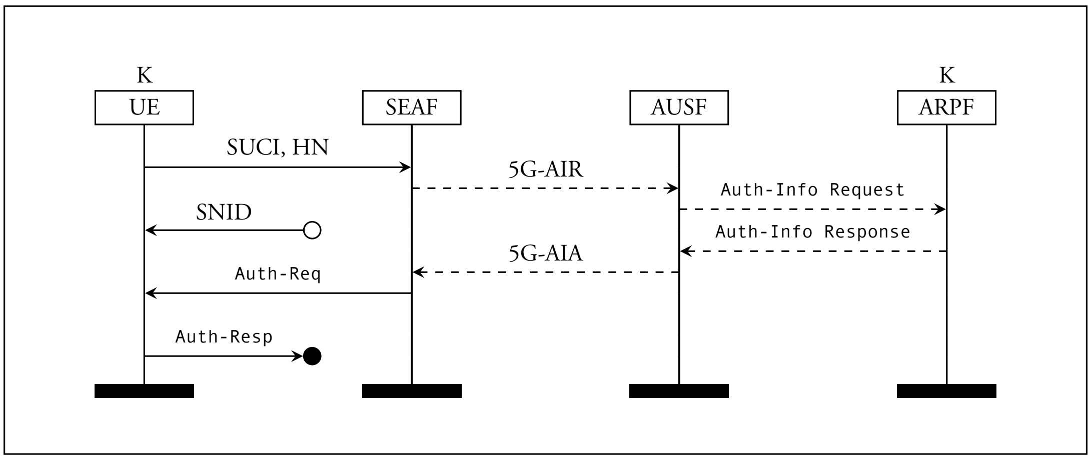


Figure 4.11: SNID injection on the insecure UE ↔ SEAF channel

5.10 Authentication and Authorization	(from TS 33.501 p.23)
Serving network authentication: The UE shall authenticate the serving network identifier through implicit key authentication. The meaning of 'implicit key authentication' here is that authentication is provided through the successful use of keys resulting from authentication and key-agreement in subsequent procedures.	

Figure 4.12: Implicit serving network authentication (from [8] p.23)

agreement violation for the K_{SEAF} is specifically due to the lack of agreement on the term SNID, and not anything else.

Figure 4.11 illustrates the adversary injecting an arbitrary SNID onto the UE ↔ SEAF channel: the UE has no way of validating the authenticity of the received serving network's ID, so assumes it is valid. The adversary must then block the UE's `Auth-Resp` response message, as otherwise a genuine SEAF would quickly discover that the UE generated HRES* and AUSF generated HXRES* do not match. As the UE then does not receive any rejection messages from the serving network, it assumes the authentication was successful, and so from the UE's point of view, the protocol is finished. The UE will then attempt to start normal communications with a nearby base station and SEAF.

It is worth noting that outside of the definition of 5G-AKA, TS 33.501 seemingly excludes this attack: see Section 5.10, cited in Figure 4.12. This attempted communication and use of the K_{SEAF} with the wrong SNID *after the protocol has finished* will fail, but we agree that this attack still violates serving network authentication in the explicit sense. We leave the question of whether this violation would allow the adversary any separate, meaningful benefit for future work; we suspect that a fake base-station (or gNB) might be able to de-conceal the SUCI in this scenario. We propose and formally verify a fix for this authentication violation in Section 4.12.2.1.

Thirdly, session confusion. As with the secrecy attack, allowing different sessions for different users to be accidentally confused is a violation of correctness, and also causes the violation of secrecy and authentication properties. We have described the secrecy violation caused by session confusion in some detail in Section 4.10.1; the principle and message flow behind many of the authentication violations is the same. In these violations marked 'SC', session confusion occurs at the same point as with the secrecy violation, i.e., two sets of authentication vectors sent by the ARPF are received by the AUSF at roughly the same time resulting in a race condition, and the intended session for each is not reliably identified.

We discuss broader implications of the discovered vulnerabilities in Section 4.14.

4.11 Weaker threat model, stronger channel properties: $\mathcal{A}_{Weaker}$

As described in Section 4.8.2, in the process of finding this series of attacks, we have deliberately and explicitly modelled the network channels in $\mathcal{A}_{Basic}$ between the SEAF, AUSF, and ARPF precisely as TS 33.501 v0.7.0 describes, and with no greater or lesser protections or security capabilities.

We recognise that standards often make implicit assumptions about the reality of engineering solutions, so in this section we explore some possible additional properties that the secure network channels may observe in any real-world implementation: the properties which we now explore are **not** specified or required by TS 33.501 [8].

We emphasise that the security properties of any authentication and key-agreement protocol *must not depend on implicit engineering solutions* – instead, such implicit requirements should be made explicit.

We note that the reason the secrecy violation works is a race condition and allowing identity mis-binding: there is nothing to ensure that the '`Auth-Info Response`' message sent by the ARPF isn't accidentally fed into the wrong session; potentially a session initiated by an attacker. This can be considered either as a stronger channel property, or equally, a weaker threat model: such a threat model would ideally prevent the adversary from confusing messages between parties within the 5G core network.

In the following section, we explore the range of threat models $\mathcal{A}_{Weaker}$: we deny the adversary the ability to take advantage of race conditions or session confusion by binding sessions together more tightly with specific Channel Session IDs. These IDs are unique, random nonces inserted into each message to ensure that 'response' messages are paired up correctly with their original 'initiator' message's session. We first list four options for the channels and then discuss their resultant properties and implications.

N.B. The threat model $\mathcal{A}_{Weaker}$ only modifies the secure channel properties of $\mathcal{A}_{Basic}$; the other adversary capabilities such as D-Y channel behaviours and adversary key reveal remain the same.

4.11.1 Four secure network models

There are several ways in which underlying channels might maintain session state IDs (or not) which have different consequences in the context of our attack. We define four different models of channel properties, with new session state IDs variously included in all messages transmitted over certain channels; this is to ensure pairing of specific session states before and after sending and receiving a message to another party.

All four models listed below maintain confidentiality, integrity, replay protection over the channels, and authenticity of the involved parties (i.e., the authenticity of the parties at either end of the secure channels, namely the SEAF, AUSF, and ARPF) as a minimum: this means that all of the below models meet or exceed the properties required by TS 33.501. The listed properties are in addition to the requirements of the standard.

1. In the first model, we do not include any channel-session IDs in transmitted messages. We believe this is equivalent to the required channel properties specified by TS 33.501. This model corresponds, for example, to using long-lived TLS sessions between two parties.

2. In the second model, we augment one channel's security properties by adding channel-session IDs just to messages sent over the AUSF $\leftrightarrow$ ARPF channel. This means that all pairs of `Auth-Info Request` and `Auth-Info Response` messages now contain unique IDs to ensure that the `Auth-Info Response` message's contents are definitely transmitted directly to the same session that sent the original `Auth-Info Request` message (and that it therefore shouldn't end up in the AUSF state for the wrong user). This model effectively approximates a new TLS session per protocol run between these two actors.

3. In the third model, we do broadly the same thing as in the second, but just between the SEAF and AUSF. Note that the AUSF does not forward on any information about the SEAF $\leftrightarrow$ AUSF channel-session ID to the ARPF. This means that a single, unique ID is added (per session) to each of the `5G-AIR`, `5G-AIA`, and `5G-AC` messages sent between the SEAF and AUSF. This models channel-session IDs across the boundary of the serving and home networks.

4. In the fourth model, we include *both* channel-session IDs in their respective messages, i.e., the combination of models 2 and 3.

We believe that all four models are possible candidates for actual network implementation.

Model vs. Property	Secrecy of K_{SEAF}
Model 1: No channel-session IDs	✗
Model 2: IDs for AUSF ↔ ARPF	✓
Model 3: IDs for SEAF ↔ AUSF	✗
Model 4: Channel-session IDs for **both** channels	✓

Table 4.6: How do the underlying channel properties affect the attack? (i.e., with or without channel-session IDs)

4.11.2 Analysis and results: $\mathcal{A}_{Weaker}$

We formally analysed each of these weaker threat models using the Tamarin Prover. We detail the secrecy results of different channel properties and channel-session IDs on the attack in Table 4.6. We consider the secrecy of the anchor key from the points of view of both the AUSF and the SEAF, where secrecy was not previously maintained.

The first result (Model 1) is as stated in the main secrecy violation: it shows that when no channel-session IDs are included, there exist attacks such that both the AUSF and SEAF are fooled into thinking the anchor key K_{SEAF} for an honest user is secret, when in reality it is not.

Adding a channel-session ID *per protocol run* between the AUSF and ARPF, i.e., Model 2, correctly prevents the violation of the secrecy of the anchor key. It is worth noting that the connection between AUSF and ARPF is completely internal to a specific "home network": connections between these two actors are internal and do not necessarily cross any network boundaries. As such we believe there is an argument to be made that requiring a brand new channel-session ID per protocol run might not be seen as an engineering / implementation necessity or priority above e.g., the same requirement across the serving network ↔ home network divide.

Adding a channel-session ID per protocol run between the AUSF and SEAF, i.e., Model 3, does **not** prevent the attack. This is the boundary between the serving and home networks, so would seem to be a likely and realistic priority candidate for a requiring a new channel-session ID per protocol run.

Adding channel-session IDs to both network channels i.e., Model 4 prevents the attack; this is a result of having correctly fixed the identity binding issue across the AUSF ↔ ARPF channel. This is a rough approximation of our proposed fix from Section 4.12.

4.11.3 Implications of alternative channel properties

Our analysis indicates that:

1. Confidentiality, integrity, and replay protection on channels, and authenticity of involved parties are necessary but not sufficient protections on secure network channels,
2. Channel ID binding to messages are not always sufficient, and that stronger, protocol-level solutions are often required to guarantee a protocol's desired security properties, and
3. Protocols cannot and must not rely on implicit assumptions about the properties of underlying network implementations; these assumptions cannot be left as implementation details, especially when these details have tangible security consequences.

It might be acceptable for identity binding or channel-session IDs to be left as implementation details when they only affect the reliability of a protocol, but when these details have security impact, they must not be left to the implementers of a specific system. This is even more important than usual when concealed and ephemeral identities are involved in a protocol.

While some implementations might effectively use e.g., a new TLS or Diameter session per protocol run to ensure the correct pairing of messages to original sessions for the sake of efficiency and not un-necessarily confusing packets, this is not specified in the standard. Our analysis shows that such a mechanism would not just prevent message confusion for reliability, but would in fact be security-critical.

In any case, to ensure that *all* implementations of the standard provide strong security guarantees, it is clear that the standard is currently at least underspecified, and needs to include one of our suggested fixes from the following section.

4.12 Proposed fixes

The essence of the session confusion-based attacks is identity mis-binding that can occur when the sessions in the channel between AUSF and ARPF are not bound tightly enough. This leads to two possible ways to prevent these attacks: one can either bind the identities of the intended parties to each message all the way through the 5G-AKA protocol, or one can ensure a one-to-one mapping between the high-level 5G-AKA sessions and its internal AUSF $\leftrightarrow$ ARPF sessions. We propose the latter solution as a fix to the 5G-AKA protocol, as while both prevent the secrecy violation, we believe this approach also upholds strictly stronger authentication properties than using the user's identity as the session ID.

Using the Tamarin Prover, we have formally verified that this solution prevents the secrecy violation and various authentication violations. We provide full formal verification results for this in Section 4.12.2.

4.12.1 Proposed fix: tighter session binding

The session confusion attack causing e.g., the secrecy violation is dependent on the ability of messages between the AUSF and ARPF from one 5G-AKA session to end up in that channel for another 5G-AKA session. Currently, there is nothing in the protocol specification that prevents this. This binding can be fixed in several ways:

1. **The protocol should include a fresh (unique, random) value in `Auth-Info Request`. The ARPF should include this in `Auth-Info Response`, and the AUSF should check that they match.**
    - To ensure similar session binding across the SN / HN boundary, i.e., between the SEAF and AUSF, the SEAF should also include a *different* fresh value in `5G-AIR`; the AUSF should include the same value in `5G-AIA`; the SEAF should then check that they match.
2. Emulating individual sessions within a long-lived TLS or Diameter session between both pairs of the AUSF and ARPF, and the SEAF and AUSF by an intermediate layer. In practice, this boils down to turning the previous solution into a separate emulation layer that creates new session-identifiers for each `Auth-Info Request` and expects the `Auth-Info Response` to be bound to this session identifier.
3. Initiating entirely new TLS or Diameter sessions between 5G Core Network parties *per 5G-AKA session* (i.e., between both AUSF $\leftrightarrow$ ARPF and SEAF $\leftrightarrow$ AUSF.

Each of these minor changes would successfully bind the correct sessions to the messages at each stage of the protocol, preventing this (and other) identity and session mis-binding attacks from occurring. We believe that the small proposed modifications have negligible impact on the computational and transmission efficiency of the protocol. We propose method 1 to prevent session confusion.

4.12.2 Verification results for proposed fix

We have formally analysed the properties of the 5G-AKA protocol with the proposed fix from Section 4.12.1. Specifically, we 'implement' version 1 of the proposed fix, by including fresh values in the messages between the AUSF $\leftrightarrow$ ARPF and SEAF $\leftrightarrow$ AUSF. Table 4.7 details the new secrecy properties, and Tables 4.8, 4.9 and 4.10 detail the resultant authentication properties from the points of view of the UE, serving network, and home network respectively. We refer the reader back to Section 4.9 for a reminder of how to interpret these results tables.

Party	Term	Result
UE	K_{SEAF}	✓
SEAF	K_{SEAF}	✓
AUSF	K_{SEAF}	✓
ARPF	K_{AUSF}	✓
	K	✓

Table 4.7: Secrecy properties of 5G-AKA: with proposed fix

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✓	✗ (SNID)	N/A	✓	✗ (SNID)	✗ (SNID)
AUSF	✓	✗ (SNID)	N/A	✓	✗ (SNID)	✗ (SNID)
ARPF	✓	✗ (SNID)	N/A	✓	✗ (SNID)	✗ (SNID)

Table 4.8: Authentication properties of 5G-AKA from the **UE's** point of view: **with proposed session ID fix**

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	✓	T
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table 4.9: Authentication properties of 5G-AKA from the **serving network's** point of view: **with proposed session ID fix**

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	✓	T
SEAF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table 4.10: Authentication properties of 5G-AKA from the **home network's** point of view: **with proposed session ID fix**

It is worth noting that while our session-ID binding solution correctly fixes the secrecy violations and many of the previous agreement violations, some of the properties, particularly from the point of the UE, are still violated.

We discuss the AUSF column first: as described in the original results tables in Section 4.9, at no point is the UE ever aware of the distinction between the parties within the UE's home network, i.e., the split between the AUSF and ARPF. The UE only cares that it is talking to the correct home network overall (made up of the AUSF and ARPF), and should not mind which specific servers it talks to indirectly within the home network.

We chose to model the ARPF's ID as being representative of what the UE considers to be the home network. As such, we mark the AUSF column's results *in relation to the UE* as "Not Applicable".

Secondly, agreement on the ID of the SEAF, or the term SNID. As discussed in Section 4.10.2, this is a violation of agreement over the identity of the SNID, as the UE never learns this term in an authenticated message. This then causes agreement to fail on the term K_{SEAF}, as this term is derived by the UE from a series of terms including SNID; the ARPF, AUSF, and SEAF will derive an anchor key K_{SEAF} on which the UE will not agree if the adversary has injected an arbitrary SNID into the UE's unauthenticated serving network discovery phase.

Once more, we give results compared to the informally stated security properties from Section 4.4. We consider which of these are upheld by 5G-AKA under $\mathcal{A}_{Weaker}$:

Secrecy properties under $\mathcal{A}_{Weaker}$:

S1.	Secrecy of an honest subscriber's long term key K:	✓
S2.	Secrecy of anchor keys K_{SEAF} and K_{AUSF}:	✓

Authentication properties under $\mathcal{A}_{Weaker}$:

A1.	The SN and UE agree on the identity of the UE:	✓
A2.	The UE and SN agree on the identity of the SN:	✗
A3.	The HN and SN agree on the identity of the UE:	✓
A4.	The UE and HN agree on the identity of the HN:	✓
A5.	The UE and HN agree on the identity of the SN:	✗
A6.	The UE, SN, and HN agree on K_{SEAF}:	✗
A7.	The anchor key K_{SEAF} must not be replayable:	✗

This is a distinct improvement on the results from Section 4.9, but due to the lack of agreement on the SNID, various agreement properties are still not achieved.

4.12.2.1 Fix for SNID agreement violation, and verification

To correct this violation of agreement, we propose that the SNID be added to the definition of the MAC (defined in TS 33.102 [6]), as this is keyed by the long-term secret key, K. This would redefine the MAC from

```
MAC = f1(K, <SQN, RAND, AMF>)
```

to the following:

```
MAC = f1(K, <SQN, RAND, SNID, AMF>)
```

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✓	✓	N/A	✓	✓	✓
AUSF	✓	✓	N/A	✓	✓	✓
ARPF	✓	✓	N/A	✓	✓	✓

Table 4.11: Authentication properties of 5G-AKA from the **UE's** point of view: with proposed session ID **and SNID** fixes.

We have formally verified that this minor change now allows the 5G-AKA protocol to gain non-injective agreement on the SEAF's identity from the UE's point of view, and both non-injective and injective agreement on the K_{SEAF} from the UE's point of view. We re-state the table of results from the UE's point of view with this final fix in Table 4.11.
For the final time, we give results for the informally stated security properties from Section 4.4, but now with the SNID fix included:

Secrecy properties under $\mathcal{A}_{Weaker}$ (with SNID fix):

S1. Secrecy of an honest subscriber's long term key K: ✓

S2. Secrecy of anchor keys K_{SEAF} and K_{AUSF}: ✓

Authentication properties under $\mathcal{A}_{Weaker}$ (with SNID fix):

A1. The SN and UE agree on the identity of the UE: ✓

A2. The UE and SN agree on the identity of the SN: ✓

A3. The HN and SN agree on the identity of the UE: ✓

A4. The UE and HN agree on the identity of the HN: ✓

A5. The UE and HN agree on the identity of the SN: ✓

A6. The UE, SN, and HN agree on K_{SEAF}: ✓

A7. The anchor key K_{SEAF} must not be replayable: ✓

With these fixes, we believe 5G-AKA now explicitly achieves all of its desired security properties within the symbolic model.

4.12.3 Re-statement of proposed fixes

We have gone through multiple stages of analysis and modification followed by proposed changes before arriving at the final model which meets its stated security properties. As such, we gather our preferred proposed changes together:

1. The AUSF should include a fresh (unique, random) value in `Auth-Info Request`. The ARPF should include this in `Auth-Info Response`, and the AUSF should check that they match.
2. The SEAF should include a *different* fresh (unique, random) value in `5G-AIR`; the AUSF should include the same value in `5G-AIA`; the SEAF should then check that they match.

3. The SNID should be added to the MAC, i.e., the MAC should be defined as: `MAC = f1(K, <SQN, RAND, SNID, AMF>).`

4.12.4 Alternative fixes

4.12.4.1 Explicit identity binding

An alternative way to fix the *secrecy* attack is to create explicit identity binding rather than session binding throughout the 5G-AKA protocol, as follows:

- In the `Auth-Info Response`[2] message sent from the ARPF to the AUSF, the SUPI *and SUCI* of the intended UE should be added to the message.
- In the `5G-AIA`[3] message sent from the AUSF to the SEAF, the SUCI should be added to the message. The SUPI is already included in the concealed setting; this must additionally be included in the non-concealed setting.

Adding the relevant identities to each of the `Auth-Info Response` and `5G-AIA` messages now successfully binds the correct parties to the messages throughout the full flow of the protocol, preventing this identity mis-binding attack from working. If the addition of the full SUCI value to these two messages is deemed to require too much bandwidth, using a cryptographically secure hash of the SUCI can also suffice.

The disadvantage of this proposed fix is that while it prevents the secrecy attack, it does not prevent other agreement and authentication violations. While the SUPI is globally unique, and hence plausibly suitable as a session ID value, each session would use the same ID, i.e., the SUPI. This does not prevent two sessions *from the same SUPI* from becoming confused, the confusion of which would violate agreement on the resultant session key.

We include this here for completeness as it is a plausible modification to the 5G-AKA protocol to prevent what could be considered the highest priority attack to mitigate, i.e., the secrecy attack.

4.12.4.2 Other options

We have considered several alternative fixes, but they all seem either more complex or insufficient. For example, one might consider putting unique nonces in other ways in the channels to solve this attack, especially since this is likely to be implemented at a lower level due to engineering concerns. However, in Section 4.11 we have already given four separate techniques for including such nonces and evaluate their effectiveness.

[2]As specified in 6.1.3.2-2 of TS 33.501 v0.7.0.
[3]As specified in 6.1.3.2-5 of TS 33.501 v0.7.0.

4.12.5 Session binding does not always imply security

It may be tempting to conclude that any solution to the race condition (in the honest case) prevents the attack. It turns out this is not the case. To prove this, we give an example of a solution that succeeds in preventing honest session confusion, and hence the race condition in the honest case, but is still vulnerable to a variant of the described attack. The underlying idea is that it is possible to prevent honest session confusion in a way that can be subverted by an adversary.

Let us assume the standard required the UE to choose a fresh nonce and append it to the first message sent to the SEAF; this message otherwise contains the SUCI, SUPI, or 5G-GUTI. If this UE-chosen nonce was then appended by all other parties (SEAF, AUSF, and ARPF) to all messages within this 5G-AKA protocol run, this would be sufficient to prevent honest session confusion. If the first message contained a SUCI or 5G-GUTI rather than a SUPI it would also not reveal any information about the user's SUPI or identity.

However, this is not sufficient to prevent the secrecy attack. If the 'unique nonce' is chosen by the UE, the attack still holds: the adversary can simply repeat the nonce sent by an honest UE, or if it is initiating both protocol runs required for the attack itself, it can simply insert the same nonce into both runs of the protocol. This further improves the probability of session confusion (and therefore the attack) succeeding, as no other honest sessions should accidentally get confused with either of these two.

Each sub-case of our proposed fix in Section 4.12.1 relies upon tighter session binding, but avoids the above error by ensuring that the adversary cannot control the identifying term or method used to bind sessions; each option allows trusted parties to choose the session binding method or identifier, and the adversary is not able to influence it as the choice is made within the 5G core network.

The alternative fix proposed in Section 4.12.4.1 does in some way allow the session identifier to be influenced or chosen by the adversary, but the SUPI or SUCI are specific to an individual user where a fresh nonce is not, and hence are implicitly bound back to the original intended user, preventing the secrecy attack. As discussed, this does not necessarily uphold other authentication properties.

While there are many other engineering solutions which would *coincidentally* prevent our attack, we believe this demonstrates that not all solutions to the problem of session confusion necessarily prevent it either. Similarly, we believe this further demonstrates that any solution required to prevent this attack must be mandated by the standard, and that security critical details must not be left as an implementation decision.

4.13 Compromised {channels, components}: $\mathcal{A}_{Stronger}$

In Section 4.11, we modelled and analysed the threat model $\mathcal{A}_{Weaker}$, considering stronger channel properties between the 5G core network parties. In addition to the already required properties of confidentiality, integrity, and authenticity, we considered making the channels strictly stronger, i.e., by introducing various session-binding methods across each or both of the channels. This implies a weaker threat model, as these modified channel properties mean the adversary is no longer able to take advantage of race conditions. We considered these stronger bindings as a response to and proposed fix for the secrecy and authentication violations which we have discovered.

We now consider the implications of strictly weaker channel and component properties, by considering *stronger* adversary capabilities against the previously secure channels and components. This is the threat model $\mathcal{A}_{Stronger}$.

Informally, this section explores what could happen if an adversary were able to gain illegitimate access to a secure channel or component within the 5G Core Network. In effect, we explore whether this protocol relies upon all of the components and secure channels, and whether they all have a role to perform, or whether any serve little or no purpose within 5G-AKA. We note that each component has many roles to fulfil outside of its functionality within the protocol and what we model, and that conclusions drawn here are only for the protocol specifically, not wider functionality within 5G. We additionally note the fact that the ARPF's functionality is usually performed in a Hardware Security Module, requiring a greater level of security than other components. Even though compromise of core network components is explicitly disallowed, it is plausible that this could suggest the overall threat model from TS 33.501 may implicitly consider compromise of the AUSF.

We re-emphasise that all of these compromised variants of any of the involved channels or components are *explicitly disallowed* by the threat model within TS 33.501 (i.e., $\mathcal{A}_{Basic}$); these variants are used to explore further the range of security properties achieved by 5G-AKA in various compromise scenarios. All previously stated vulnerabilities including the secrecy violation presented to 3GPP, violating the secrecy of K_{SEAF}, were found through fine-grained component modelling under $\mathcal{A}_{Basic}$, with adversarial capabilities matching the threat model requirements of TS 33.501 as precisely as possible.

We now describe our modelling for compromised channels and components, followed by analysis and results for each.

4.13.1 Compromised channel modelling

We consider the situation where one or both of the 'secure' channels (SEAF ↔ AUSF or AUSF ↔ ARPF within the 5G Core Network) have been compromised. We consider two distinct forms of channel compromise on a per-channel basis: 'Dolev-Yao'-style compromise, and 'read-only' compromise. The first gives the adversary full control of the network channel, as if it were a standard Dolev-Yao channel, as used for the UE ↔ SEAF channel, and as described in Section 2.3 and [86].

We create this change in modelling automatically by adding to the channel communication rules: the definitions of `send_secure` and `receive_secure` are still in place, but they are also joined by definitions for new rules, `send_insecure` and `receive_insecure`.

For the **Dolev-Yao-style** adversary-controlled channels, we define the rules `send_insecure` and `receive_insecure` as follows:

```
rule send_insecure:
    [SndS(<channelname,SendType,ReceiveType>,A,B,m)]
    --[SendInsecure(channelname,A,B,m)]->
    [Out(<<channelname,SendType,ReceiveType>,A,B,m>)]

rule receive_insecure:
    [In(<<channelname,SendType,ReceiveType>,A,B,m>)]
    --[ReceiveInsecure(channelname,A,B,m)]->
    [RcvS(<channelname,SendType,ReceiveType>,A,B,m)]
```

These rules consume and produce `SndS(…)` and `RcvS(…)` facts in the same way as the premise and conclusion from `send_secure` and `receive_secure`, but now everything that passes through them is also known to and modifiable by the adversary due to use of the public network channel `In(…)` and `Out(…)` facts.

The terms `channelname`, `SendType`, and `ReceiveType` are bookkeeping terms, only ever instantiated with strings such as `'ausf_arpf'` or `'seaf_ausf'` (for `channelname`) and any one of `'SUPI'`, `'SEAF'`, `'AUSF'`, or `'ARPF'` for the `SendType` or `ReceiveType` terms. These strings are then used in lemmas and restrictions to work out which channels (and later, components) have been compromised, and therefore if the trace we consider is a valid trace in the specified threat model.

For the '**Read-Only**' compromise of channels, we do not need a `receive_insecure` rule as the adversary cannot inject or modify terms on the channel in question; we only need to ensure the adversary learns the transmitted terms. We therefore add only the following rule:

```
rule send_insecure:
    [SndS(<channelname,SendType,ReceiveType>,A,B,m)]
    --[SendInsecure(channelname,A,B,m)]->
    [Sec(<channelname,SendType,ReceiveType>,A,B,m),
     Out(<<channelname,SendType,ReceiveType>,A,B,m>)]
```

Note that the conclusion of this rule contains *both* `Sec(…)` and `Out(…)` facts. This allows the protocol to proceed as normal on an authentic (and un-modifiable) but now non-confidential channel, as the corresponding 'Receive' rule will be `receive_secure`, consuming a `Sec(…)` fact, rather than `In(…)` as its premise.

We select channel properties within the protocol models through use of a series of macros. These options within macros introduce the correct rules, restrictions, and required sources lemmas to the resultant Tamarin `.spthy` file, creating a model with the specified threat model. We modify these varying adversarial capabilities on a per-channel basis through the use of macro-introduced first-order restrictions on allowed traces: we can allow Dolev-Yao-style compromise on either one (specifically named) or both of the two 'secure' channels, and likewise we can allow read-only compromise on either one or both of the channels. We can then limit the number of times the insecure channel rules can be used (if so desired) in a similar manner.

The Tamarin rules as stated above allow the adversary unrestricted ability to access (and in the D-Y case, inject/delete terms into/from) any arbitrary channel: we limit the adversary's behaviour to accessing (and potentially injecting terms into) only a particular type of channel through the use of restrictions, matching against the `channelname`. These restrictions limit which traces are valid. For example, to allow compromise of the AUSF ↔ ARPF channel only, we use macros to introduce the following restrictions when the relevant compromise model is enabled:

```
restriction only_ausf_arpf_chan_compromised_send:
    "All channelname A B m #i . SendInsecure(channelname,A,B,m) @ #i
        ==> not(channelname = 'seaf_ausf')"

restriction only_ausf_arpf_chan_compromised_receive:
    "All channelname A B m #i . ReceiveInsecure(channelname,A,B,m) @ #i
        ==> not(channelname = 'seaf_ausf')"
```

During Tamarin's protocol analysis, any traces which violate a restriction are not considered valid, and are dropped from further consideration. For example, a trace which contains the Action Fact:

```
SendInsecure('seaf_ausf', AUSF, SEAF,
                    <'air', <SUPI, <'5G', $VPLMNID>, '3gpp_creds'>>)
```

would violate the property described in the '`only_ausf_arpf_chan_compromised_send`' restriction, and hence construction of any trace requiring this Action Fact would be dropped by Tamarin.

Once these new adversarial powers have been selectively introduced to the model (by means of these additional rules and restrictions), we can then analyse the resultant security properties of the protocol under each new threat model. We provide analysis and results of our channel compromise experiments in Section 4.13.4.

4.13.2 Compromised component modelling

We now separately consider the situation where one or more components within the protocol have been compromised. Here, instead of compromising the (otherwise secure) network channels, we compromise individual components.

4.13.2.1 Compromise of the SEAF and/or AUSF.

We start by considering the **SEAF** and **AUSF**: before the start of a protocol run, neither of these components have any shared secret with each other, the UE, or the ARPF within the 5G-AKA protocol, but instead leverage their secure and authentic channel access: this access is what prevents the adversary from impersonating them to other actors within the 5G core network, and allows other actors within the core network to be confident of the identity of the component to whom they are talking.

To ensure secure and authentic 5G core network access, the SEAF, AUSF, and ARPF will necessarily have some shared secret(s) (whether directly or indirectly), but this is explicitly performed at a lower layer of the 5G core network compared to the 5G-AKA protocol. For our models we therefore continue to use the secure channel abstraction as described by Basin, Radomirovic, and Schmid in [41], additionally requiring channels where sessions are correctly bound together; we have already detailed our usage of secure channel modelling in Section 4.8.2.

The reason for discussing secure *network* access in a section considering component compromise is that without other shared secrets (as is the case here), taking over a component's secure network channel access is sufficient to fully impersonate said component. The SEAF and AUSF components do not maintain state beyond a run of the 5G-AKA protocol, so if the adversary is able to send and receive the correct messages for a particular protocol run in lieu of the honest component, no other party should be able to detect any difference, and the progression of protocol runs will not be limited or restricted by the adversary's involvement. Where the adversary can control or impersonate a component, we would, however, expect the adversary to be able to learn all (secret and public) terms normally available to this component in the course of the 5G-AKA protocol.

We therefore allow the adversary to impersonate components by giving the adversary access to the secure network channels. Recalling that the adversary learns and introduces terms through use of `Out` and `In` facts respectively, this is achieved through the addition of the following rules in Tamarin:

```
// Compromised Component Send rule.
// The compromised component is A, sending it on the channel A-B.
// B is (potentially) honest.
rule component_compromised_send:
    [In(<<channelname,SendType,ReceiveType>,A,B,m>)]
  --[CompCompromisedSend(A,<<channelname,SendType,ReceiveType>,A,B,m>)]->
    [RcvS(<channelname,SendType,ReceiveType>,A,B,m)]

// Compromised Component Receive rule.
// Honest component A sends a message to compromised component B
//  (controlled by the Adversary)
rule component_compromised_receive:
    [SndS(<channelname,SendType,ReceiveType>,A,B,m)]
  --[CompCompromisedReceive(B,<<channelname,SendType,ReceiveType>,A,B,m>)]->
    [Out(<<channelname,SendType,ReceiveType>,A,B,m>)]
```

For *channel* compromise rules, we gave the adversary control of both ends of a network channel; for *component* compromise, we only give the adversary access to one end of the relevant secure channel, but give it access to all the secure channels the component would have (e.g., a compromised AUSF will be able to read and inject terms into both the SEAF ↔ AUSF and AUSF ↔ ARPF channels).

The TAMARIN rules as stated above allow the adversary unrestricted access to impersonate any arbitrary component: as with channel compromises, we use macros to introduce limitations on the adversary's behaviour (namely learning terms from and injecting terms into only a particular component's endpoints) through the use of restrictions, requiring a match on the `SendType` (e.g., `'SEAF'`, `'AUSF'`, or `'ARPF'`) for the `component_compromised_send` rule, and a match on the `ReceiveType` for the `component_compromised_receive` rule. For example, to allow compromise of SEAFs only, we introduce the following restrictions through the use of macros when the relevant compromise model is enabled:

```
restriction CompromiseAllSEAFs_send:
    "All A SendType B ReceiveType cn m #i.
    CompCompromisedSend(A,<<cn,SendType,ReceiveType>,A,B,m>) @ #i
        ==> SendType = 'SEAF' "

restriction CompromiseAllSEAFs_receive:
    "All A SendType B ReceiveType cn m #i.
    CompCompromisedReceive(B,<<cn,SendType,ReceiveType>,A,B,m>) @ #i
        ==> ReceiveType = 'SEAF' "
```

Note that this allows compromise of all SEAFs, including the SEAF with which we as honest components may be communicating. In many threat models we often consider the scenario where compromise of all actors or components *other than the ones with whom we are communicating* is allowed, but not our communications partner(s). It is therefore important to distinguish between threat models which allow compromise of other components (but not the ones in *my* run of a protocol) and models which allow compromise of all types of a particular component. This specific threat model allows compromise of *all* components of type SEAF; we also create the similar threat model allowing compromise of all AUSFs in the same manner.

4.13.2.2 Compromise of the UE and/or ARPF.

Having considered the two components which do not have explicit shared secrets with each other within 5G-AKA, we now consider those which do, i.e., the UE and ARPF. As the **UE** does not have any privileged or secure network access, it is sufficient to give the adversary the UE's long-term key K.

```
rule reveal_LTKSym_ue_arpf_compromise:
    [!LTKSym(SUPI, ARPF, K)]
  --[UE_ARPF_Compromise_K(SUPI,K)]->
    [Out(K)]
```

For the **ARPF**, it is sufficient to compromise the initiating UE's long-term key K with the above rule, in combination with giving the adversary access to the secure network channels in the same manner as with the SEAF and AUSF above.

While we can model and analyse compromise of both the UE and ARPF, it is worth considering what these compromises actually mean for the protocol. From the ARPF's point of view, the home network fundamentally needs to authenticate the UE's identity (e.g., for access permissions and billing purposes), so the adversary compromising the UE seems to be a plausible candidate for a compromise from which a protocol cannot reasonably be expected to recover Post-Compromise Security [62] notwithstanding. As described in Section 4.7, we do not consider modern notions of security such as post-compromise security here, as the 5G-AKA protocol does not even attempt to achieve forward secrecy, never mind properties such as post-compromise security.

The UE has no secure network access, and the only mechanism by which the UE attempts to gain a secure connection to its home network is through its secret knowledge of its long-term shared symmetric key, K. We therefore observe that without needing any secure network access, any party who also has K can imitate the user's ARPF to the UE, and by extension, the intermediate AUSF and SEAF.

4.13.2.3 Actor Key Compromise.

More generally, and irrespective of 5G, it is clear that learning e.g., Alice's long term secret key allows the adversary to impersonate Alice to other parties. Actor Key Compromise (AKC) is the secondary property whereby upon compromise of Alice's (and only Alice's) long term secret key, the adversary can successfully impersonate *other parties to Alice*. Basin et al. [36] show the impossibility of achieving a large class of authentication properties when only using symmetric cryptography and hashing, including all of those properties described by Lowe in [131].

5G-AKA only uses hashing, and upon completion, symmetric key cryptography: from results in [36], we can be certain that any authentication properties as a result of this protocol are not resilient to AKC. We therefore know that if the adversary has compromised the UE we cannot achieve any meaningful authentication properties from its point of view, and hence do not consider compromise of the UE any further.

We are not necessarily able to make the same claim for the ARPF, as the ARPF has secure network access as well as the secret key K: while the 5G-AKA protocol itself cannot create AKC resistance in its present form, the choice of protocol to create the underlying secure channels might well provide this resistance, albeit only for parties within the 5G core network.

In terms of choices for 5G secure channel protocols: TLS 1.2 achieves AKC resilience in the mutually authenticated DHE RSA mode (among others) although does not achieve it in all modes [36]. Mutually authenticated TLS 1.3 achieves AKC resilience automatically: the only available mutual authentication mode uses Elliptic-Curve Diffie-Hellman (ECDH). While more generally the use of asymmetric-key cryptography is not sufficient to guarantee AKC resilience, all AKC susceptible modes have been removed from TLS 1.3. The other major protocol used within the 5G core networks is Diameter [93], but we cannot find any literature discussing or formally analysing this specific property in this protocol, so without modelling the protocol ourselves, we are unable to determine whether this achieves AKC resilience, or remains vulnerable.

The way we have chosen to model component compromise within the 5G core network does not allow actor key compromise attacks. An AKC attack against (for example) a compromised SEAF would proceed as follows in terms of Tamarin execution: the adversary sends a message to the honest SEAF, pretending to be the AUSF. This will necessarily use the '`component_compromised_send`' rule as that is the only one of the two component compromise rules that takes terms `In(...)` from the adversary. Now the SEAF must consume a '`RcvS(...)`' fact, and the only one it can consume is a '`RcvS(...)`' from the AUSF, with `SendType = 'AUSF'`, as the AUSF is the only other component with which the SEAF communicates on the secure channels. However, the restriction in place for this threat model says that the adversary can't send a message as anything apart from the SEAF (because it requires `SendType = 'SEAF'`), ensuring that AKC attacks are not possible in this abstraction. N.B. This modelling decision only affects component compromise, and not any other results based upon channel or component properties.

4.13.2.4 Which components can the adversary compromise?

Within compromised component modelling we additionally consider two strands: firstly, if we compromise a component of type X, we say the adversary can compromise *all* components of type X; e.g., allowing SEAF compromise implies that all SEAFs can be compromised by the adversary, whether these are the ones in use by a particular execution of the protocol which we consider or not. We refer to this as “All-X”.

Secondly, we consider the case where the adversary may compromise all components of type X *apart from “mine”*, e.g., if an honest component such as the UE thinks it is talking to a SEAF with identity ‘SNID’, then it really is, and the adversary has not compromised this specific SEAF; the adversary can still compromise any other SEAF. We refer to this as “Not-My-X”.

To achieve analysis of the second scenario, we include the following additional restrictions to the left-hand side of each lemma within Tamarin, automatically included by macro:

```
[...]
& not(Ex cn rt y m #j. CompCompromisedSend(b,<<cn,'SEAF',rt>,b,y,m>) @#j)
& not(Ex cn rt y m #j. CompCompromisedSend(c,<<cn,'AUSF',rt>,c,y,m>) @#j)
& not(Ex cn rt y m #j. CompCompromisedSend(d,<<cn,'ARPF',rt>,d,y,m>) @#j)
& not(Ex cn st x m #j. CompCompromisedReceive(b,<<cn,st,'SEAF'>,x,b,m>) @#j)
& not(Ex cn st x m #j. CompCompromisedReceive(c,<<cn,st,'AUSF'>,x,c,m>) @#j)
& not(Ex cn st x m #j. CompCompromisedReceive(d,<<cn,st,'ARPF'>,x,d,m>) @#j)
[...]
```

The free variables `cn`, `rt`, and `st` represent Channel Name, Receive Type, and Send Type respectively. The string describing the type of component will then potentially match and restrict the Send Type for `CompCompromisedSend`, and Receive Type for `CompCompromisedReceive`, and the variable ‘`b`’, ‘`c`’, or ‘`d`’ will match the SEAF, AUSF, or ARPF with whom the point-of-view component believes he is communicating, as indicated by the vector of IDs within the `Commit` Action Fact, e.g., `Commit(a,<a,b,c,d>,t,<'SUPI','K_SEAF'>) @i`. These restrictions then limit which component compromises can occur, preventing compromise of components with which the viewpoint component believes they are communicating.

In the Not-My-X scenario, for compromise of the ARPF we remove what was previously available to the ARPF in All-X, i.e., the `reveal_LTKSym_arpf_compromise` rule, as the normal K compromise rule and associated restrictions within each lemma is sufficient.

The first scenario (All-X) considers the importance of the components involved directly in the protocol; the second (Not-My-X) considers whether an honest actor needs to trust all components within the 5G core network, or just the ones to whom it believes it is talking.

We note that we model the single ARPF with whom the UE is in indirect communication as the only holder of the UE’s long-term key K. We do not know if this is accurate or if e.g., there are multiple ARPFs per home network, and therefore whether compromise of another ARPF would in reality violate the secrecy properties of K and thus the protocol. However, we also note that compromise of any ARPF within a user's home network seems a reasonable candidate for out of scope behaviour, where compromise of ARPFs from other networks seems plausibly within scope of a reasonable threat model.

4.13.3 Composition

In this thesis we do not directly consider results for scenarios with both compromised channels *and* components. In terms of composition of models and results, it is generally apparent that negative results (i.e., those where a property is not upheld) compose: additional compromise of a new element of the protocol will not remove or impede the adversary's ability to perform an 'attack' or violation of a property first exhibited in the simpler scenario, i.e., when the adversary has less power. However, the same cannot necessarily be said for results where a security property is upheld in the individual setting.

Consider three different scenarios:

1. *Component* X is compromised, but security property Φ is upheld.
2. *Channel* Y is compromised, but security property Φ is upheld.
3. Both component X and channel Y are compromised: is Φ upheld?

As the combination of compromise abilities may allow an adversary to learn new terms which were not accessible or derivable in either of scenarios 1 or 2 alone, we cannot deduce with any certainty that the results from scenarios 1 and 2 imply that Φ will be upheld in scenario 3. Automatically deducing properties when different protocols are composed is notoriously difficult, as any such analysis has to consider possibility of attacks exploiting of cross-protocol interactions [46,75,101,118,134]. Not specific to 5G, we consider the compromise of both components and channels further in Chapter 5.

4.13.4 Compromised channels: analysis and results

We now detail the analysis and results of the various channel compromise models.

4.13.4.1 Analysis

Each new channel-compromise threat model introduces new challenges. The channel-compromise threat models which we consider are:

1. Secure channels as defined by TS 33.501 (see Section 4.8)
2. Readable channels:
   (a) SEAF $\leftrightarrow$ AUSF channel is readable by the adversary
   (b) AUSF $\leftrightarrow$ ARPF channel is readable by the adversary
   (c) Both channels are readable by the adversary

3. Dolev-Yao channels:
   (a) SEAF ↔ AUSF channel is D-Y compromised
   (b) AUSF ↔ ARPF channel is D-Y compromised
   (c) Both channels are D-Y compromised

These channel-compromise threat models (i.e., the strength of the adversary) form a partial ordering, which we show in Figure 4.13. Using the implications of this partial ordering, we are able to reduce computation time spent generating results. For example, if a property is falsified in the 'secure channels' threat model, then it will necessarily also be false in every other threat model, so we can pre-populate the other results tables. Likewise, if a property is verified in e.g., the 'SEAF ↔ AUSF D-Y' threat model, then we can be confident that it is also true in the 'SEAF ↔ AUSF readable' threat model, and the 'secure channels' threat model, but we cannot make any claims about results under other threat models based upon this single data point.

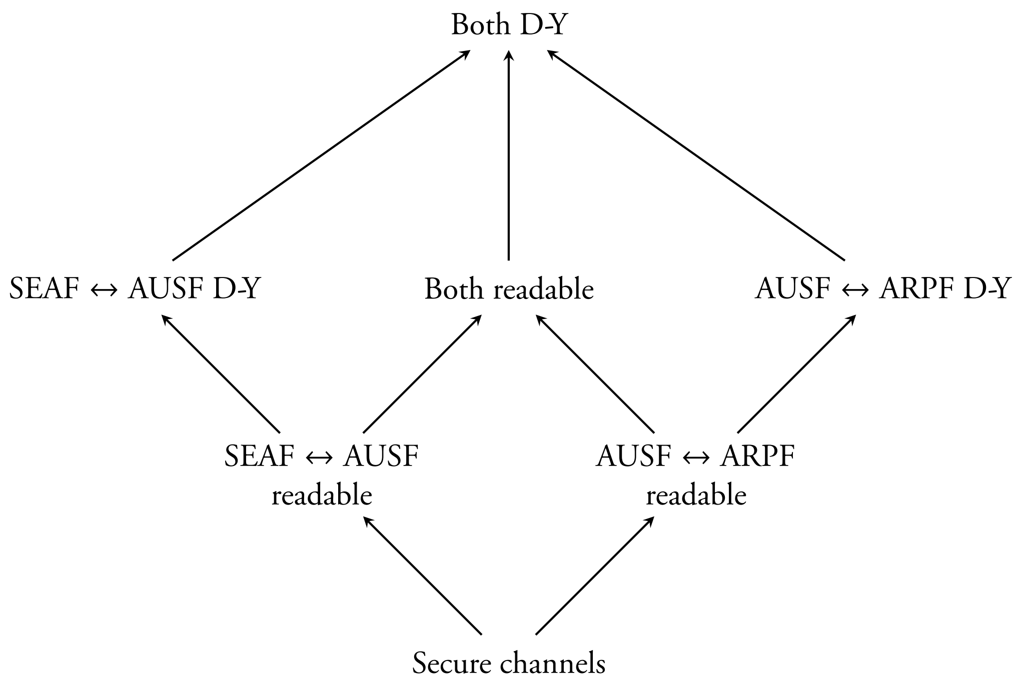


Figure 4.13: Partial ordering of network channel compromises, i.e., the ordering on the strength of the adversary.

We analyse the original, unfixed 5G-AKA protocol strictly as per the specification when under this new range of threat models against the set of considered security properties as discussed in Section 4.7. Then, we consider the same 5G-AKA protocol, but with many of the previously described issues fixed by the introduction of channel-session IDs creating TLS-like channels, as described in Section 4.12.1, and how this new protocol fares against this above range of threat models.

To aid modelling, analysis, and reproducibility of results for both compromised channels and compromised components, we place threat-model specific changes, rules and restrictions for compromised channel behaviour, described in Section 4.13.2, and the necessary sources lemmas inside `m4` macro definitions. This ensures that the modeller, analyst, or reader can easily toggle definitions in the master `m4` file (`5G-AKA.m4` [69]), whose processing then generates and outputs the relevant `.spthy` file for direct analysis of security properties in TAMARIN. For instructions on how to enable each threat model and versions of the 5G-AKA protocol, please see the `5G-AKA.m4` [69] file directly.

The introduction of new adversarial powers (in this case the ability to read terms from channels, or even stronger, the ability to read and delete from, and inject terms into channels) gives many new sources for terms that the adversary may learn and then use within a protocol. This is especially true in comparison to a protocol where otherwise the adversary could only learn terms from one of the three channels.

These new sources of terms are two-fold: firstly, the adversary may learn new terms from a read-only channel; secondly, the adversary may invent a brand new term, and inject it into a D-Y channel. Each of these situations requires new `sources` lemmas to be written and proven for each specific situation and threat model before any analysis of the protocol's security properties can occur.

As in our original analysis of 5G-AKA under $\mathcal{A}_{Basic}$ (see Section 4.7), we consider the security properties of secrecy of the session key from the points of view of each of the four components, and agreement each of the components' identities, non-injective agreement on the session key, and injective agreement on the session key, all from the points of view of all of the four components.

4.13.4.2 Compromised channels: results

As including a full set of results for each individual threat model would be quite cumbersome, we give an overview of the differences between normal channels and the compromised channels for each of the broken and fixed protocols; we include the full results in Appendix A.1, but we illustrate the upheld properties vs. the partial ordering of channel threat models in Figures 4.14 and 4.15.

Secrecy results Compromising secure network channels has broadly the effect on key secrecy that one would expect:

- Giving the adversary either read-only or D-Y capabilities over both of the secure channels violates the secrecy of the **session keys** (K_{SEAF} and K_{AUSF}), both when the channels are precisely as the specification, and when they include channel-session IDs as per our proposed fix in Section 4.12.1. This is because the key K_{AUSF} is transmitted directly from the ARPF to the AUSF, and the key K_{SEAF} is calculated by the AUSF and transmitted directly to the SEAF. Compromising these channels gives the adversary access to these terms.

- Where only the SEAF $\leftrightarrow$ AUSF channel is compromised (read-only or D-Y), this maintains the secrecy of the key K_{AUSF}, but does not maintain the secrecy of K_{SEAF}.
- Where only the AUSF $\leftrightarrow$ ARPF channel is compromised, the adversary learns K_{AUSF}, and can therefore also derive K_{SEAF}.
- The secrecy of the **long-term key K** is unaffected by channel compromise.

For full results, please see Appendix A.1.

N.B. K_{SEAF} is calculated using a KDF, from K_{AUSF} and SNID; see Annexes A.1, A.2, and A.6 of TS 33.501 [8] for the use of the KDF to create K_{AUSF} and K_{SEAF}, and TS 33.220 [10] for the definition of the KDF itself.

Authentication results

- Giving the adversary D-Y capabilities over both of the 'secure' network channels violates almost all of the agreement properties, in both the naïve channel scenario, and when the channels are bound with channel-session IDs. The only properties which are maintained in these compromise scenarios are agreement from the point of view of the UE, with the ARPF, on the IDs of the UE and the ARPF. This is due to the fact that the UE and ARPF are the only two parties who share the long-term key K, and unless the adversary (separately) compromises this secret key, cannot impersonate the ARPF to the UE.
- D-Y compromise of the SEAF $\leftrightarrow$ AUSF channel only is a similar picture: the majority of agreement properties are violated, but now in addition, from the point of view of the home network, we gain agreement with the ARPF on the IDs of the AUSF and ARPF.
- More properties are upheld when the AUSF $\leftrightarrow$ ARPF channel is D-Y compromised than when just the SEAF $\leftrightarrow$ AUSF channel is similarly compromised: we gain agreement between the parties on either end of the non-compromised channel for the majority of terms. The serving network gains agreement with the AUSF over the majority of terms (although still not injective agreement on the K_{SEAF}), and vice versa, the home network gains agreement with the SEAF on most terms, excepting the K_{SEAF} due to a lack of authentication on the SNID term.
- Restricting the adversary to read-only compromise of the channels improves the number of properties which are upheld, although this does not materially change the picture for perhaps the most important party, the UE. This is because the adversary can learn terms from core network channels, and then inject any modified or derived

term it desires into the inherently D-Y channel between the UE and SEAF, violating most of the agreement properties. Termination notwithstanding, the picture here is identical across all three non-TLS-like readable-only channel compromise models. The 5G core network parties broadly gain agreement between each other on their own identities.

- In terms of agreement properties, models with TLS-like channel-session IDs are at least as strong as or stronger than the models without. This difference is especially noticeable when we require channel-session IDs within channels where the adversary has read-only capabilities: agreement on component IDs and the K_{SEAF} within the 5G core network are upheld internal to the 5G core network, but due to the adversary's extra knowledge, this agreement is not extended across the UE $\leftrightarrow$ SEAF channel to the UE.

We conclude that full D-Y or read-only adversarial compromise of secure channels in 5G-AKA is devastating for both properties of secrecy and agreement. We note that this is true even with strong channel-session binding, and that the UE is worst affected in terms of violation of properties.

Security properties vs. channel threat models partial ordering Figures 4.14 and 4.15 illustrate 5G-AKA's upheld security properties vs. the partial ordering of channel threat models (originally shown in Figure 4.13).[4] These two figures show the naïve model specified precisely as per $\mathcal{A}_{Basic}$ (i.e., no channel session binding) and the correctly bound TLS-like model of Section 4.12.1 (i.e., with our proposed channel session binding, Fix 1) respectively. The power of the adversary, or the strength of the threat model increases with the direction of the edges. The presence of a property at a node means that the property is upheld in that threat model, *but none higher;* said property is therefore implicitly upheld in all lower threat models. We omit any nodes where no property changes, e.g., we omit the nodes 'SEAF $\leftrightarrow$ AUSF readable' and 'AUSF $\leftrightarrow$ ARPF readable' as all properties upheld at these nodes are also upheld in higher nodes (e.g., the 'Both channels readable' node). Figure 4.14 recreates Tables A.1–A.24, and Figure 4.15 recreates Tables A.25–A.48. Labels of properties marked 'Secrecy of X' indicate that the Secrecy of term 'X' is upheld in that threat model, from the stated actor's point of view, but no higher. The labels of all other properties (separated by commas) are agreement properties, and should be read as 'Component 1 non-injectively agrees with Component 2 on the term at position 3 (potentially one from a set)', except where the final term is *Inj*–K_{SEAF}, in which case it identifies Injective agreement on the term K_{SEAF}. Any properties not listed on the graph are not upheld in any of the given threat models.

[4]These figures are very similar in style to the protocol security hierarchies in [34]; please see this paper for more applications of such graphs.

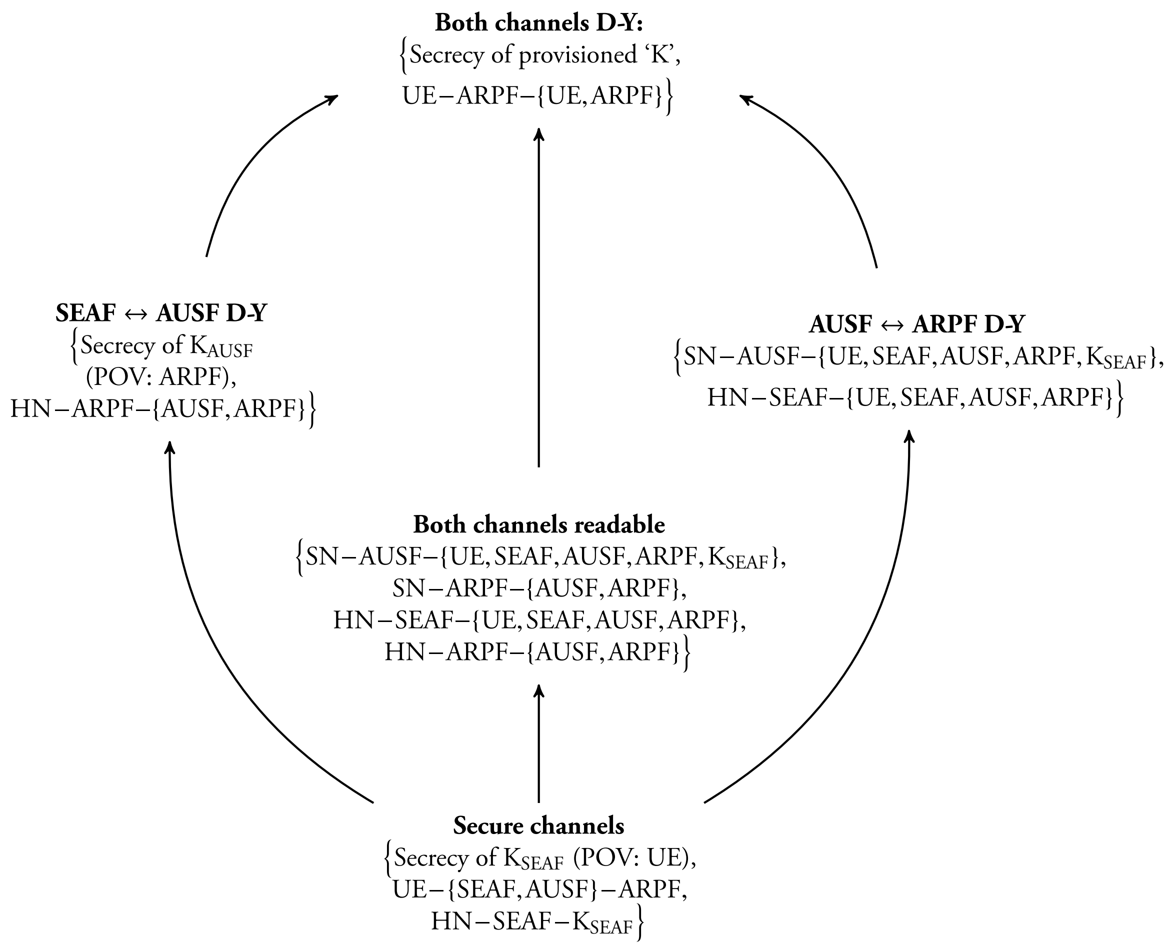


Figure 4.14: Naïve model: security properties vs. network channel threat models. This represents the results from Tables A.1–A.24. Please see Section 4.13.4.2 "Security properties vs. channel threat models partial ordering" for more details.

We note that more properties are included in Figure 4.15 than in Figure 4.14, and likewise more properties are maintained against stronger threat models: the session binding preventing race condition ensures that more properties are upheld to start with, and then even under partial channel compromise, properties which were previously upheld fare slightly better overall. While the overall number of security properties maintained under partial channel compromise is low, we believe this further demonstrates the importance of guaranteeing correct session binding.

4.13.5 Compromised components: analysis and results

As with the channel compromise results from the previous section, we do not include the full component compromise results here, instead including them in Appendix A.2.

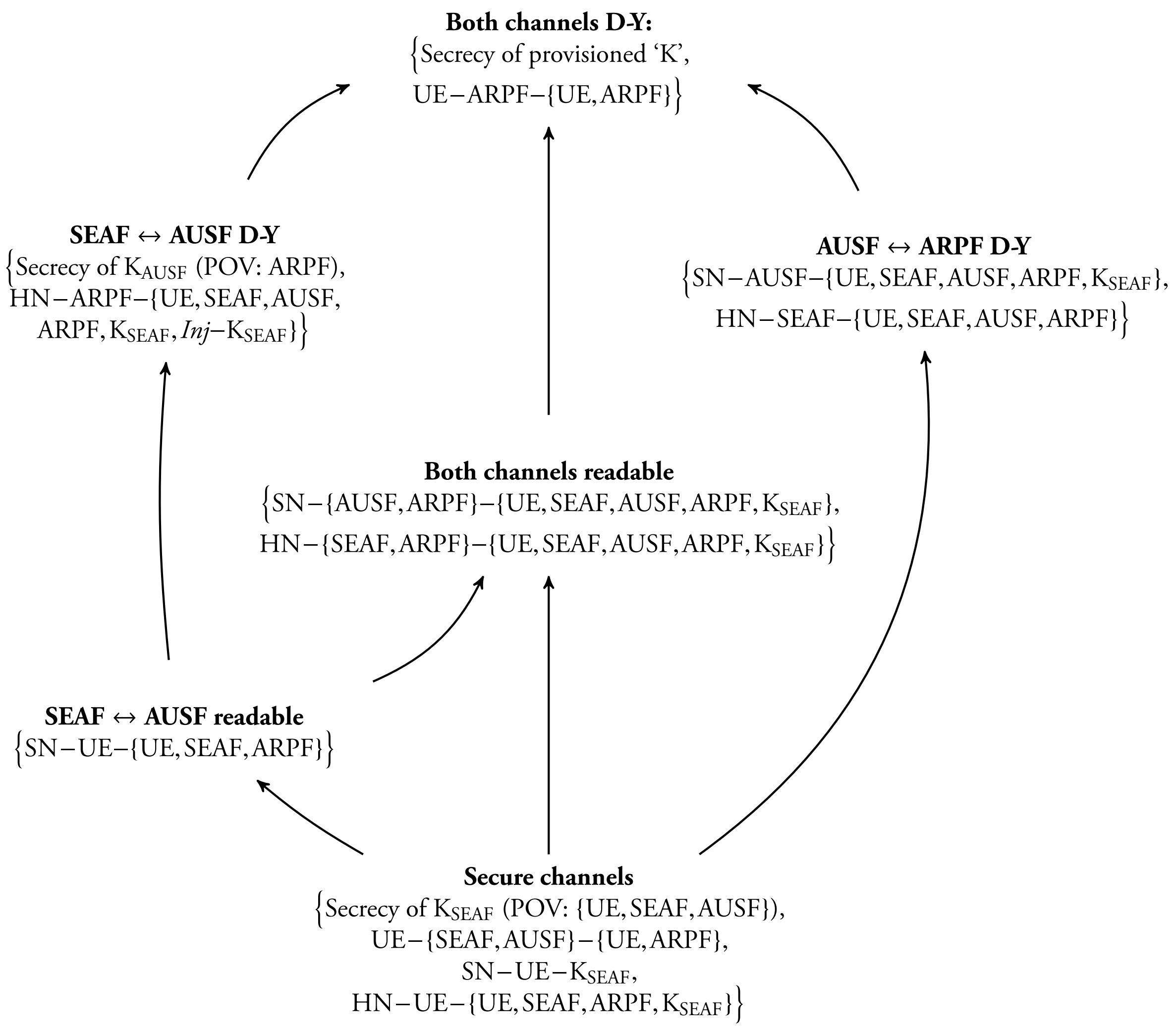

Both channels D-Y:
{Secrecy of provisioned 'K',
UE−ARPF−{UE, ARPF}}
SEAF ↔ AUSF D-Y
{Secrecy of K_AUSF (POV: ARPF),
HN−ARPF−{UE, SEAF, AUSF, ARPF, K_SEAF, *Inj*−K_SEAF}}
AUSF ↔ ARPF D-Y
{SN−AUSF−{UE, SEAF, AUSF, ARPF, K_SEAF},
HN−SEAF−{UE, SEAF, AUSF, ARPF}}
Both channels readable
{SN−{AUSF, ARPF}−{UE, SEAF, AUSF, ARPF, K_SEAF},
HN−{SEAF, ARPF}−{UE, SEAF, AUSF, ARPF, K_SEAF}}
SEAF ↔ AUSF readable
{SN−UE−{UE, SEAF, ARPF}}
Secure channels
{Secrecy of K_SEAF (POV: {UE, SEAF, AUSF}),
UE−{SEAF, AUSF}−{UE, ARPF},
SN−UE−K_SEAF,
HN−UE−{UE, SEAF, ARPF, K_SEAF}}

Figure 4.15: TLS-like model: security properties vs. network channel threat models. This represents the results from Tables A.25–A.48. Please see Section 4.13.4.2 "Security properties vs. channel threat models partial ordering" for more details.

Achieving results in Tamarin for compromised components has proved more difficult than the channel compromise results, leading to fewer results terminating automatically with Tamarin's built-in heuristics. As a result, the majority of the component compromise results were achieved through manual direction of Tamarin's interactive mode, and we provide descriptions of how to repeat these manual proofs in the README associated with `5G-AKA.m4` [69]. We remind the reader that choice of heuristic or manual intervention in proof steps does not affect the outcome, i.e., whether Tamarin concludes a property is violated or upheld. The choice of heuristic or manual proof-steps only affects the duration of computation, and in many cases, whether Tamarin terminates or not.

As we detail in Section 4.13.2.4, we consider two main scenarios: firstly, we consider “All-X”, which lets the adversary compromise all components of a particular type, including the specific components with whom actors believe they are communicating. Secondly, we consider “Not-My-X”, i.e., executions of the protocol where the adversary is allowed to compromise all components apart from the ones with whom parties believe they are communicating. E.g., if a UE believes it is talking to ‘SEAF.1’ in a particular run of the protocol (as indicated in its vector of component IDs), then the adversary can compromise all other SEAFs, but not that one.

The first scenario (All-X) considers the importance of the components involved directly in the protocol; the second (Not-My-X) considers whether an honest actor needs to trust all components within the 5G core network, or just the ones with whom it believes it is communicating.

As discussed in Section 4.13.2.3, due to the impossibility of Actor Key Compromise resilience across the UE $\leftrightarrow$ SEAF insecure channel, we do not consider compromise of the UE.

Secrecy results: All-X

- Allowing the adversary to compromise any of the components violates the secrecy of the session key K_{SEAF}.
- If the compromised component type is SEAF, this only violates the secrecy of K_{SEAF}. The adversary does not learn the term K_{AUSF}.
- If the compromised component type is AUSF or ARPF, this violates the secrecy of both the K_{AUSF} and K_{SEAF}.
- The secrecy of K is only violated if the UE or ARPF is compromised.

Authentication results: All-X

- Compromising the **SEAF** causes most agreement properties from the point of view of the UE to be violated. From the point of view of the serving network, due to AKC resilience, the adversary cannot violate agreement the UE, AUSF, or ARPF. From the point of view of the home network (i.e., upon completion of the protocol), the home network gains agreement with the UE and ARPF on most identities and terms. It does not gain agreement with the SEAF on any terms.
- Compromising the **AUSF** causes similar violation of most agreement properties from the point of view of the UE. All agreement properties are violated from the point of view of the serving network. Due to AKC resilience, from the point of view of the home network, many of the internal properties are upheld between home network, and serving network. Based upon the extra knowledge gained by the adversary combined with the D-Y UE $\leftrightarrow$ SEAF channel, compromise of the AUSF causes all the explored agreement properties with the UE to be violated.

- Compromising the **ARPF** unsurprisingly causes most agreement properties to be violated between all parties, except between the SEAF and AUSF.

From this, we conclude that compromise of any 5G core network component used within an execution of 5G-AKA has a severely detrimental effect on the protocol's secrecy and authentication properties.

We now consider the results of component compromise excluding the components with whom an actor believes they are communicating, i.e., Not-My-X.

Secrecy results: Not-My-X

- Compromising components other than the ones with whom we believe we are communicating broadly does not impact the secrecy of the session keys or K. This result is not to be taken for granted, and it is a positive outcome that secrecy is maintained.

- Compromising other AUSFs allows for academic compromise of the K_{SEAF} and K_{AUSF} from the points of view of the UE and ARPF respectively; however, this violation of secrecy is due to modelling choices, and not necessarily indicative of a real vulnerability.

  At a high level: the UE knows with which home network it is communicating overall, but has no method within the protocol to know specifically with which AUSF it is (indirectly) communicating. We have made the modelling choice to couple the ARPF's identity to that of the home network, and these results might be different if the home network's identity had instead been bound to a specific AUSF. We believe that if "AUSFs for home network Z" are not allowed to be compromised, but that all other AUSFs across other networks are, and the home network will not allow one of their customers' UEs to perform 5G-AKA through a different network's AUSF (which seems a reasonable assumption), then this problem may well not occur.

Authentication results: Not-My-X

- Allowing compromise of all **SEAFs** apart from 'mine' violates almost all of the authentication properties from the point of view of the UE: this is because the SNID is not explicitly authenticated. When re-modelled and analysed to include our proposed SNID authentication fix as described in Section 4.12.2.1, the protocol broadly achieves its desired properties. We were unfortunately not able to gain termination on the injectivity of K_{SEAF}.

  The security properties from the points of view of the SEAF and AUSF in this situation are broadly upheld regardless of the inclusion (or not) of the SNID fix.

- Allowing compromise of all **AUSFs** apart from 'mine' similarly violates almost all of the authentication properties from the point of view of the UE. In this instance, this violation of these properties is not just due to the lack of SNID authentication, but additionally due to the fact that the UE does not know to which specific AUSF it is talking; it only knows with which home network it is communicating. This would allow a malicious AUSF to violate the UE's agreement properties with both the SEAF and AUSF. While our models indicate that these properties are not upheld, we perhaps consider this lack of agreement to be relatively academic: compromise of another AUSF within the UE's home network could reasonably be considered out of scope, and a compromised AUSF from another network might be able to fool a UE of its identity, but not another party within the 5G core network, such as the UE's genuine home network or ARPF component.

  The security properties from the points of view of the SEAF and AUSF in this situation are broadly upheld regardless of the inclusion (or not) of the SNID fix.

- Allowing compromise of all **ARPFs** apart from 'mine' shows violation of many of the agreement properties from the point of view of the UE; however, these violated properties are no more than when *no components* are compromised but the SNID authentication fix is not included. Including the SNID authentication fix broadly allows the protocol to achieve its desired properties. We were unfortunately not able to gain termination on the injectivity of K_{SEAF}.

  The security properties from the points of view of the SEAF and AUSF in this situation are broadly upheld regardless of the inclusion (or not) of the SNID fix.

As stated before, full channel and component compromise results can be found in Appendix A.

4.13.6 Conclusions from results: $\mathcal{A}_{Stronger}$

Having performed this large, systematic analysis of 5G-AKA under the range of threat models encompassed by $\mathcal{A}_{Stronger}$, what are the overall conclusions?

We initially asked whether this protocol relies upon all of the components and secure channels, whether they all have a role to perform, or whether any serve little or no purpose within 5G-AKA. We believe this analysis shows that regardless of whether each component within the 5G core network performs a major and important role in terms of key derivation or not, the security of all components is vital to the overall security of the protocol. This analysis has shown that preventing compromise of each core component is essential for both secrecy and authentication properties, as all components have significant influence over whether the protocol achieves the secrecy and authentication properties it seeks to uphold.

In many cases compromise of one of the core components (SEAF, AUSF, or ARPF) does not allow the adversary to 'fool' the other core components in terms of violating agreement properties; however, the same broadly cannot be said for the UE, which, our analysis shows, becomes vulnerable to violations of both secrecy and agreement in most component compromise scenarios, especially those in the "All-X" scenario. For the "Not-My-X" scenario, we believe our analysis demonstrates the importance of explicitly including the SNID in an authenticated part of the protocol's messages.

Adversarial compromise of secure channels within the 5G core network has a similarly devastating effect. Unsurprisingly, Dolev-Yao-like (read, inject, delete, etc.) access allows an adversary to violate most secrecy and authentication properties. As we would broadly expect, read-only access to secure channels does not allow the adversary to break agreement properties between 5G core components; more interestingly, this simple read-only access to either of the SEAF $\leftrightarrow$ AUSF or AUSF $\leftrightarrow$ ARPF channels does allow the adversary to violate almost all agreement properties from the UE's point of view, as this additional information allows the adversary to impersonate a serving and/or home network successfully to the UE over the insecure UE $\leftrightarrow$ SEAF channel.

In summary, while the 5G-AKA protocol meets its desired security properties after inclusion of our proposed fixes, it is still a very fragile protocol. 5G-AKA loses the ability to uphold most of its desired properties very quickly upon compromise of almost anything outside of its explicitly allowed, relatively weak threat model compared to the majority of modern key-exchange protocols. We recognise that many design decisions will have been strongly influenced by legacy considerations, but nevertheless we urge 3GPP to pursue stronger properties for future generations of AKA, hopefully meeting Perfect Forward Secrecy [139], and even properties such as Post-Compromise Security [62]; we suppose that such a future protocol could then fall back to merely achieving session-key secrecy and (injective) agreement on actors and terms in the case of component or secure-channel compromise.

4.14 Secrecy violation: statement of implications

Having described $\mathcal{A}_{Basic}$, $\mathcal{A}_{Weaker}$, and $\mathcal{A}_{Stronger}$, and discovered the properties which are and are not upheld under each group of threat models, we now consider specifically how the **secrecy violation** under $\mathcal{A}_{Basic}$ (described in Section 4.10.1) might be instantiated in the real world, and the implications of such an attack. The secrecy violation allows a malicious actor, or an attacker that compromised the long-term key of an actor, to impersonate *another* user to a serving network.

Strictly from the point of view of the 5G-AKA protocol, this allows the attacker to agree upon an anchor key (and thereby gain access to a serving network) dishonestly, under the newly generated false credentials of a legitimate user. This is a substantial containment problem.

We note that this attack holds whether the protocol uses encrypted/concealed SUPIs or not, i.e., the attack holds both when the protocol uses the null-scheme for 'SUPI (non-)concealment', *and* when SUPIs are concealed. We make this distinction because the SEAF and AUSF know strictly *more* about the claimed identities of session owners in the non-concealed situation than they do when identities are concealed. This, in turn, implies that there is a deeper identity mis-binding issue that is not caused directly by SUPI/SUCI encryption. We first discovered this attack in the non-concealed setting, and then saw that it also applied to the concealed protocol.

This attack relies on a race condition between two sessions of the protocol. This means the attack is probabilistic, and an attacker would not be able to guarantee success on every run; however, in any secure protocol, there ought not exist *any* run of the protocol under normal circumstances which violates the required security properties.

N.B. This does **not** allow an attacker to decrypt any honest radio traffic, past or present, originally generated by the impersonated, legitimate user.

4.14.1 Potential practical implications

In the real world, we conjecture that this attack might allow an attacker to access a serving network (and its services) in the name of a legitimate user other than itself. This attacker could then bill services, air-time, or access charges to another user account, rather than its own; this is clearly not the intended behaviour or level of security required within 5G networks.

We are not confident of the range of further authentication and authorisation procedures which may or may not be in place distinct from the 5G-AKA protocol, or any billing–authentication procedures: e.g., whether specific billing actions sent back to a user's home network are tied to and verified against a named anchor key or not. We note that an ARPF *would* be able to establish that the anchor key was not for the correct user; but that an AUSF or any other party without direct access to the honest user's long-term key K within the 5G Core Network would not. We do, however, believe it is plausible that once access is granted in the form of an anchor key, this key is sufficient to allow a user to perform the normal range of actions within a network. Access in the form of an anchor key gives the resultant ability to derive the other keys, such as K_{gNB}, K_{AMF}, CK, IK, CK′, and IK′ associated with it; see the 5G Key Hierarchy in [8, Figure 6.2.1-1].

It is also plausible that the same attack works when the SEAF is within the home network, in addition to the attack we describe, with the SEAF in the serving network – and thus the attacker would not have to be in the physical realm of a serving network. We have not modelled any distinctions between the two situations, so only make the weaker claim.

We acknowledge that there may be other technical measures within 5G that make full implementation of this attack impossible in the real world. Regardless, we believe that any authentication and key-agreement protocol must meet its own required security properties as defined.

The strongest statement we can definitively make is that the 5G-AKA protocol on its own does not meet its security requirements. However, as the primary method for authentication and key-agreement within 5G, we believe that 5G networks should not rely solely upon secondary mechanisms for security; we believe this is sufficient reason on its own to fix the protocol to prevent this and similar attacks.

4.15 Disclosure, and changes to TS 33.501

4.15.1 Responsible disclosure

With respect to the discovered vulnerabilities, we opted not to follow any responsible disclosure path, as at the time of writing, 5G (and 5G-AKA) is not yet an implemented, used, or complete standard; our publication of this work was intended to highlight the stated attack *during* the standardisation phase, rather than after it has been finalised.

4.15.2 Liaison with 3GPP SA3 and CT4

After discovering the secrecy violation described in Section 4.10.1, we prepared a document for public release describing the vulnerability, its potential implications, and our proposed fix. We then contacted members of the 3GPP SA3 (Security) working group to inform them of our research, and to request that they read and comment upon our vulnerability report (available at https://www.cs.ox.ac.uk/5G-analysis/).

Following the document's release and distribution, we received mostly supportive feedback from members of SA3 and other researchers. Some SA3 members did not view the vulnerability as an issue with TS 33.501 specifically, suggesting instead that it was not necessarily the standard's responsibility to ensure security, and that individual vendors could introduce further measures to guarantee security if they desired. We respectfully but strongly disagree.

Steve Babbage and Tim Evans from Vodafone Group kindly helped us to prepare a Change Request to TS 33.501, with the aim of fixing the documented issues. They then submitted this document on our behalf to 3GPP SA3. This submitted document was "S3-180727 - pCR to TS33.501 - Session binding to prevent potential 5G-AKA vulnerability" [181], and was considered at "3GPP SA WG3 Meeting #90Bis" in San Diego, 26 Feb –

2 March 2018. While this specific change request itself was not immediately accepted at this meeting of 3GPP SA3, the working group noted the document for further consideration.

As a result of our publicly released document and associated change request, SA3 have since stated that they believe our described race condition could arise, and that they need to ensure its mitigation. This has resulted in a formal liaison document being sent to 3GPP CT4, the Core Network and Terminals working group, which (to paraphrase) "standardizes aspects within the Core Network focusing on services including Mobility Management within the Core Network. CT WG4 is also responsible as a protocol steward for some IP related protocols."

The liaison document from SA3 to CT4 "S3-181468-v3 - LS to CT4 on avoiding race condition in 5G AKA" [76] submitted by Sander de Kievit (TNO) to 3GPP SA WG3 Meeting #91 in April 2018 states:

> In a research paper, it has been pointed out to SA3 that a 5G AKA race condition could be possible if a UE initiates multiple Registration Requests. SA3 thinks that such a race condition could arise because, based on the current authentication service description, the `Nausf_ UEAuthentication_Authenticate` Request sent by the SEAF includes the SUCI while the `Nausf_UEAuthentication _Authenticate` Response received by the SEAF does not. This means that the request and response cannot be bound together using the SUCI.
>
> If across the whole procedure, there is nothing that binds the first request and the final response, SA3 believes that an attacker could exploit this in the scenario where the SEAF sends multiple `Nausf_ UEAuthentication_Authenticate` Request(s) for the same UE and is unable to correlate the different responses to the respective requests.

The document further requests of 3GPP CT4 whether they agree that this lack of binding is present, whether it has been taken into account, and if it has, how the race condition is avoided (requesting reference to the relevant specification). Finally, the document attaches a formal change request [84] (prepared and submitted by NTT Docomo), which is proposed as a potential solution to the race condition. This proposed solution is a generic adaptation of our proposed fix from Section 4.12.1.

At the time of writing, CT4 have not yet provided an answer to SA3, and no solution has yet been formally adopted. That said, we believe that SA3 have shown strong intent to require one of two outcomes: either CT4 will demonstrate that it has been fully mitigated elsewhere (and hopefully this will then be highlighted within TS 33.501), or if it is not yet mitigated, they will adopt our proposed (or similar) solution, thereby preventing the race condition and resultant vulnerability.

4.16 Conclusions

In this chapter we have demonstrated issues within the draft 5G-AKA protocol, particularly one which, if unmitigated, could potentially allow a malicious actor to impersonate an honest user to a network. We propose a range of possible fixes, and we have verified the correctness of our preferred solution using the TAMARIN Prover. We have worked with 3GPP SA3, encouraging them strongly to adopt one of our proposed fixes immediately, and are pleased with the significant progress made in this respect.

We believe that this result demonstrates the importance of fine-grained component-based modelling, as well as answering our second research question: without this level of detail in modelling, we would not have discovered the presented race condition. Correct but incomplete and even defensibly simplistic modelling of protocols is very valuable, but we believe our results demonstrate the further value of spending the extra time and effort modelling protocols in as much detail as is reasonably possible with the tools available.

We have additionally provided a systematic and in-depth analysis of 5G-AKA, considering vulnerabilities under a broad range of secure channel- and component-compromising threat models. This has demonstrated the fragility of the 5G-AKA protocol; to improve the resilience of future mobile telephony authentication and key-agreement protocols, we urge 3GPP to minimise the impact that legacy decisions have on future protocol design in the way it has on 5G-AKA, and to require much stronger, more resilient protocol properties, principles, and primitives.

We recognise that standards often make implicit assumptions about the reality of engineering solutions, and that there may be other mechanisms in place to mitigate the real-world impact of this protocol attack. In particular, we conjecture that method 2 of our proposed fix (Section 4.12.1) might be proposed as an implementation choice in practice, accidentally mitigating many attacks. Even if that were the case, our analysis demonstrates that such a mechanism would in fact be security-critical.

It might be tempting to think that our attack would be prevented if underlying layers (accidentally) provide session binding, but we showed in Section 4.12.5 that this is not necessarily the case. Regardless of whether or not implementations or mechanisms at lower layers accidentally prevent the described attack, our analysis demonstrates that such mechanisms would in fact be security-critical.

We emphasise strongly that security critical properties of any security protocol *must not depend on implicit engineering solutions*. In other words, the specification of the standard should be such that any implementation provides the desired security properties. This is not the case for either the version of TS 33.501 which we modelled (v0.7.0, [8]) nor the latest version at the time of writing, v15.0.0.

Many of the encountered issues under the whole range of threat models are exacerbated by continued reliance on symmetric cryptography. We recognise that legacy considerations restrict the choices available to protocol and system designers, but continuing to rely solely on symmetric cryptography in this context seems wholly inadequate in 2018. Future mobile telephony standards could achieve much stronger security properties (in both authentication and secrecy) with the introduction of almost any modern, asymmetric key-exchange such as SIGMA or HMQV [122, 123] at the heart of any new 'AKA' protocol.

In other ways, this reticence to adopt public-key cryptography has proven (most likely accidentally) to be partially beneficial: symmetric-key cryptography is widely believed to be far less vulnerable to attacks by quantum computers than asymmetric-key cryptography, and the best known quantum attacks on symmetric-key cryptography can be mitigated by doubling key sizes [150, 170]. While many 'post-quantum secure' algorithms have been proposed, the leading schemes very often require much larger key-sizes and more computation power to perform the required cryptographic operations than might be available on a SIM card [152]. These schemes, while currently undergoing much scrutiny, have not yet been widely accepted.

In terms of confidentiality, use of non-quantum-secure asymmetric-key cryptography could be a concern: an adversary today (without a quantum computer) would need only to record all asymmetrically encrypted traffic, and wait until they possess a quantum computer powerful enough to decrypt it. Estimates vary on how long this wait would need to be, but this time scale and level of required resources are widely believed to be within the reach and patience of nation-state actors. However, this is not such a concern for authentication or agreement properties, as to violate these an adversary normally has to be able to inject 'correct' messages into the communications channel in the same time-span as the protocol run; this seems to preclude waiting until the adversary is in possession of this sufficiently powerful quantum computer.

Identity binding and protocol design is tough, especially within complex, multi-party protocols with subtle assumptions. We believe that the discovery of these issues further demonstrates the importance of systematic automated verification for security-critical functionality and protocols.

Finally, we want to emphasise the importance of communication between academia and industry: as we described earlier (and elaborate on in Section 6.5), Tsay and Mjølsnes [177] discovered a very similar style attack across a different network boundary in 2012, but when designing 5G-AKA, 3GPP do not appear to have taken this into account. If they had, we believe our presented vulnerability could also have been avoided. When we contacted 3GPP with our report, not a single reply mentioned this previous research.

While the implications of some of the issues we discuss can be subtle and tricky to convey, we need to ensure these can still be communicated to industry clearly and accurately. Some parties may not yet be used to receiving and acting upon results and feedback from academia. In spite of this, in our view, it is important to continue to analyse systems, providing feedback to the relevant stakeholders, ensuring that our research is received, read, and most importantly, acted upon.

Future work

While this analysis has covered many aspects of the 5G-AKA protocol, there is much work we did not pursue. Of interest is further analysis which would meaningfully improve the breadth and depth of our results, and further understanding of the very large 5G authentication ecosystem. Firstly, with the new release of much improved XOR modelling, equational reasoning, and analysis in Tamarin 1.4.0 (May 2018) and higher, it would hopefully be possible to gain stronger confidence in the verification results that we have achieved. Secondly, 5G-AKA is not the only authentication protocol in 5G: authentication of a UE through a serving network to its home network and vice versa can be achieved either through the flagship 5G-AKA, or the the older protocol EAP-AKA′ [29], as described in [8, §6.1.3.1]. Both protocols use the long-term shared, symmetric key K, and thus it is not necessarily true that verifying each protocol individually implies the security of both protocols in composition. The next step would be to model all of the authentication protocols available to a UE in the same overall models, thus providing composition results.

As we discuss in Chapter 6, we believe it is still possible to downgrade 3G, 4G, and 5G to 2G, which would then allow interception by fake base-station, as no mutual authentication is performed in 2G. While this attack construction would be 're-discovered' in formal analysis, modelling all the protocols together would more easily allow for verification of any future proposed mitigations against downgrade attacks, such as an extension field for all older protocols including a signature over the complete transcript.

We note that the only source of randomness in the protocol is the ARPF: no other parties contribute new randomness each run to the values from which the anchor key is derived. It would be interesting to study the consequences of the compromise of this single RNG, modelling it as a separate component from the ARPF. Regardless, we believe that where multiple components are involved in a key-agreement protocol, many of these parties should contribute to the entropy used in a protocol if possible, to lessen the consequences of faulty [162] or predictable RNG libraries [45], or a compromised initial seed.

"It's not good enough to have a system where everyone using the system must be trusted."

— Robert Morris

5 The Operational Semantics of Partial Compromise

5.1 Introduction

Having modelled and analysed two major real-world systems in depth, considering their security properties both strictly under their standards' threat models, and separately under partial compromise, we observe that the modelling and analysis methods used to achieve partial compromise were system-specific. This does not detract from the results we have achieved in Chapters 3 and 4, but we might have been able to gain these results more easily, and could potentially gain even stronger assurance of the completeness of any partial compromise results more generally if the modelling and analysis techniques we used had partial compromise 'built in' from the outset.

In this chapter, we therefore build upon the experience from previous chapters to create an initial symbolic mathematical framework for modelling and analysing systems as collections of abstract, communicating components. This framework builds upon many of the lessons and conclusions drawn from the previous chapters. It has been designed such that it can be used to assess the correctness of a system's security properties under a much wider and more fine-grained range of threat models than is usual, most importantly allowing for modelling of whole systems under partial compromise.

These semantics only serve as a starting point. Further work in modelling real-world, multi-component systems is needed to assess the utility of this approach more fully. Almost all real-world systems we encountered (which might otherwise be of modelling and analysis interest) take advantage of both large amounts of state and unbounded looping;

these semantics would need to be extended to provide better support for these before such real-world systems could be analysed fully.

For our purposes, a component is anything that can be modelled as discrete, communicating subsections of a system. These might include hardware components such as a processor, keyboard, monitor, or hard drive; they might include radio base-stations, hardware security modules, control centres, or power grid sub-stations; these could equally include software components such as databases, random number generators, cryptographic implementations, or software libraries.

Instead of considering systems as a potentially unbounded number of communicating agents within large network security protocols (which have been extensively studied, e.g., in the area of key-exchange [79]), we focus on systems (single-site or distributed) with a fixed number of components and channels in a static configuration, such as industrial control systems, personal computers, mobile phones, and even cash machines.

We model these systems as (finite) collections of abstract, fixed-role components and channels performing protocols, and, in contrast to most symbolic protocol modelling to date, we do not necessarily assume a Dolev-Yao adversary [86], i.e., we allow as much or as little of the system's 'network' to be adversary controlled as needed. The most notable distinction between this framework and those starting with a Dolev-Yao attacker is that *the adversary does not necessarily control the network;* messages are *not* automatically passed to the adversary for delivery. We believe this is a more realistic representation of physical systems where an adversary might be able to intercept, probe, or control only a limited subset of communication channels, but is just as likely to have placed a backdoor in an individual component at the point of that component's design [15], manufacture [186], or transportation [178].

As such, we allow both communication channels *and* components to be compromised at any point during the execution of a system, and we describe threat models as a series of restrictions on adversarial capability, rather than explicit behaviours. This very expressive approach to both threat models and security properties allows us to move away from the stricter notion of binary compromise, both in terms of constituent components and a system's security goals.

This operational semantics allows us to model and analyse the partial compromise of systems. We start by presenting our partial compromise model, describing terms, events, systems, components, protocols, our execution model, and an example protocol execution. We then introduce the adversary, how we model its knowledge and behaviour, and how we restrict the possible actions it can take in different situations, considering both network and component compromise. We then give a worked example of a protocol under attack from an adversary, and show how this specific adversary can violate the protocol's desired security property (secrecy of a specific term). We elaborate further on the range of threat models available to modellers within this framework, and how the tools offered here give fine-grained control over what constitute valid traces of protocols under specific threat models. We give a formal description and examples of security properties and claim events, before giving a manual proof of the correctness of an example protocol under attack by a specific adversary. We finally discuss future work and implementation of tool support.

This framework uses a combination of the (network-compromise-focussed) semantics from both [34] and [74] as a generic starting point, before significantly building upon and expanding the scope and fine-grained control available to the modeller over compromise of specific components, threat models, and adversarial behaviour.

5.2 Component model

5.2.1 Terms and events

Protocols are executed by a system's components, resulting in threads. Each thread is a sequence of events that can either send or receive messages. We start by defining the lowest level of the execution of a system, that is the messages transmitted by the components. We have the pairwise disjoint infinite sets *Component*, *Fresh*, *Var*, *Func*, and TID, that is the sets of component names, freshly generated terms (random numbers, nonces, session keys etc.), variables, functions, and thread identifiers. To identify a term t bound locally to a specific thread with thread identifier *tid*, we write t^{tid}. Components transmit messages via buffers, which are defined by their end-points, i.e., a pair of components.

For a component $a \in Component$, we denote its long-term public key by $pk(a)$, and its long-term private (asymmetric) key by $sk(a)$. If the given a does not have a public or private key associated with it, we say $pk(a) = sk(a) = \perp$. We denote the long-term *symmetric* key between two components $a, b \in Component$ by $k(a, b)$. Note that $k(a, b)$ is not necessarily equal to $k(b, a)$.

The asymmetric encryption of term t_1 with key t_2 is represented by $\{\!|\ t_1\ |\!\}^a_{t_2}$, and similarly the symmetric encryption of t_1 with key t_2 is represented by $\{\!|\ t_1\ |\!\}^s_{t_2}$. *Func* models other cryptographic functions such as hash functions.

To establish the relationship between public and private keys, we assume the existence of an inverse function on terms, where t^{-1} is the inverse key of term t. We have that $pk(a)^{-1} = sk(a)$, and $sk(a)^{-1} = pk(a)$ for all $a \in Component$. $t^{-1} = t$ for all other terms.

Definition 5.2.1.1 (Terms)**.**

$$\begin{aligned} Term ::= {} & Component \mid Fresh \mid Var \mid Fresh^{\mathrm{TID}} \mid Var^{\mathrm{TID}} \\ & \mid (Term, Term) \mid pk(Term) \mid sk(Term) \mid k(Term, Term) \\ & \mid \{\!|\ Term\ |\!\}^a_{Term} \mid \{\!|\ Term\ |\!\}^s_{Term} \mid Func(Term^*) \end{aligned}$$

Definition 5.2.1.2 (Inference)**.** We define the binary relation $\vdash$, where $M \vdash t$ denotes that the term t can be inferred from the set of terms M. Let $t_0, \ldots, t_n \in \mathit{Term}$ and let $f \in \mathit{Func}$. We define $\vdash$ as the smallest relation satisfying:

$$\begin{array}{rl}
t_1 \in M \Rightarrow M \vdash t_1 & M \vdash t_1 \wedge M \vdash t_2 \Leftrightarrow M \vdash (t_1, t_2) \\
M \vdash \{\!|\, t_1 \,|\!\}^s_{t_2} \wedge M \vdash t_2 \Rightarrow M \vdash t_1 & M \vdash t_1 \wedge M \vdash t_2 \Rightarrow M \vdash \{\!|\, t_1 \,|\!\}^s_{t_2} \\
M \vdash \{\!|\, t_1 \,|\!\}^a_{t_2} \wedge M \vdash (t_2)^{-1} \Rightarrow M \vdash t_1 & M \vdash t_1 \wedge M \vdash t_2 \Rightarrow M \vdash \{\!|\, t_1 \,|\!\}^a_{t_2} \\
 & \bigwedge\limits_{0 \le i \le n} M \vdash t_i \Rightarrow M \vdash f(t_0, \ldots, t_n)
\end{array}$$

We recall that subterms t' of a term t are written $t' \sqsubseteq t$, e.g., $t_1 \sqsubseteq \langle t_1, t_2 \rangle$, and $t_2 \sqsubseteq \langle t_1, \langle t_2, t_3 \rangle \rangle$. Accessible subterms are ones which might potentially be derived by the inference rules. E.g., within the term $\{\!|\, t_1 \,|\!\}^s_{t_2}$, t_1 is accessible because $M \vdash \{\!|\, t_1 \,|\!\}^s_{t_2} \wedge M \vdash t_2 \Rightarrow M \vdash t_1$, but t_2 is *not*, because $M \vdash \{\!|\, t_1 \,|\!\}^s_{t_2} \wedge M \vdash t_1 \not\Rightarrow M \vdash t_2$. We denote the free variables of t by $FV(t)$, where:

$$FV(t) = \left\{ t' \;\middle|\; t' \subseteq t \wedge t' \in \mathit{Var} \cup \{ v^{tid} \mid v \in \mathit{Var} \wedge tid \in \mathrm{TID} \} \right\}$$

We define ground terms to be terms containing no free variables as subterms.

Definition 5.2.1.3 (Events)**.** We now define the events available to components and an adversary, in *CompEvent* and *AdvEvent* respectively. Together these form the set *Event*:

$$\begin{array}{rl}
\mathit{CompEvent} ::= & \mathrm{create}_P(\mathit{Component}) \\
 & \mid \mathrm{send}((\mathit{Component}, \mathit{Component}), \mathit{Term}) \\
 & \mid \mathrm{recv}((\mathit{Component}, \mathit{Component}), \mathit{Term}) \\
 & \mid \mathrm{del}(\mathit{Component}, \mathit{Term}) \\
 & \mid \mathsf{running}_\ell(\mathit{Component}, \mathit{Term}) \\
 & \mid \mathsf{claim}_\ell(\mathit{Component}, \mathsf{claim\text{-}type}, \mathit{Term}) \\
 & \mid \mathsf{claim}_\ell(\mathit{Component}, \mathsf{claim\text{-}type}, \mathit{Component}, \mathit{Term}) \\
\mathit{AdvEvent} ::= & \mathit{PassiveAdvEvent} \mid \mathit{ActiveAdvEvent} \\
\mathit{PassiveAdvEvent} ::= & \mathrm{BR}(\mathit{Component}, \mathit{Component}) \mid \mathrm{CMR}(\mathit{Component}) \\
\mathit{ActiveAdvEvent} ::= & \mathrm{BI}((\mathit{Component}, \mathit{Component}), \mathit{Term}) \\
 & \mid \mathrm{BD}((\mathit{Component}, \mathit{Component}), \mathit{Term}) \\
 & \mid \mathrm{CMI}(\mathit{Component}, \mathit{Term}) \\
 & \mid \mathrm{adv\text{-}send}((\mathit{Component}, \mathit{Component}), \mathit{Term}) \\
\mathit{Event} ::= & \mathit{CompEvent} \mid \mathit{AdvEvent}
\end{array}$$

These events and their precise semantics are described later, but we give a brief overview now. From the set *CompEvent*: 'create$_P(a)$' creates or starts a thread for component a of protocol P; send$((a,b),t)$ sends the term t from component a to the buffer (a,b) such that component b can then perform the recv$((a,b),X)$ action, where X is a variable. Claim and running events are bookkeeping events used to aid description and verification of the desired security properties of a protocol.

From *AdvEvent*, intuitively, *PassiveAdvEvent* are those adversary events which do not affect the state of components or buffers, and *ActiveAdvEvent* are those adversary events which do. BR(a,b) reveals the contents of the buffer (a,b) to the adversary, BI$((a,b),t)$ allows the adversary to inject a term t into the buffer (a,b), and BD$((a,b),t)$ allows the adversary to delete the term t from the buffer (a,b) before it is received. CMR(a) allows the adversary to learn all terms known by component a, and CMI(a,t) allows the adversary to inject the term t into component a's memory. adv-send$((a,b),t)$ allows the adversary to send the term t to the buffer (a,b) by controlling the *component*; the distinction between this and BI$((a,b),t)$ is which element (either buffer or component) of a system the adversary has compromised.

Unlike in a Dolev-Yao network, messages are not passed automatically to the adversary for delivery; as such, we require that 'send' and 'receive' events include an explicit declaration of the buffer for which the message is intended, as given by the ordered tuple (*Component*, *Component*) preceding the message term.

5.2.2 Systems

A system's component and buffer configuration can be depicted by a directed graph, containing vertices (components) and edges (buffers); that is $G = (V,E)$. This graph is derived from the system's protocol description; see Section 5.2.3 for more detail.

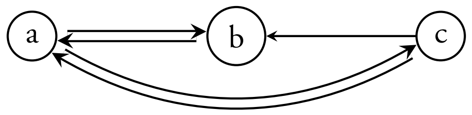


Figure 5.1: Example system with components and buffers between them

Let V be a set of vertices or components ($V \subseteq$ *Component*), and E be the set of directed edges or connections between them. For example in the diagram above, we have the vertices and edges:

$$\begin{aligned} V &= \{a,b,c\}, \qquad \text{and} \\ E &= \{(a,b),(a,c),(b,a),(c,a),(c,b)\} \end{aligned}$$

Definition 5.2.2.1 (The Buffer Function, *Net*)**.** We define a buffer for each edge, by defining the function

$$Net : (Component \times Component) \to {}^{\sharp}Term$$

that maps pairs of components to multisets of ground terms, where no subterm is an element of Var. $Net(a,b)$ is the multiset for the edge (a,b); *Net* then describes the full complement of multiset buffers for all edges in E. Intuitively, a Dolev-Yao adversary would have full read and write access to all multisets within *Net*. We require the use of multisets because the same message between two components can be sent multiple times before being received by the recipient component.

Where there is *no* edge between two components in V, we say the buffer does not exist, that is:

$$\forall x,y \in V,\ (x,y) \notin E \implies Net(x,y) = {}^{\sharp}\{\bot\}$$

where $\bot$ denotes 'non-existence', and hence cannot be written to or read from. Note this is not the same as the empty set (${}^{\sharp}\varnothing$) as this could just indicate that the buffer does not currently contain any terms.

Definition 5.2.2.2 (Component Memory, *CM*)**.** We say that the terms a component knows are its memory. We define the function

$$CM : Component \to \mathcal{P}(Term)$$

that maps components to sets of ground terms, where no subterm is an element of Var. That is, $CM(a)$ contains the terms that a either knows as part of its initial memory, or has received. Then *CM* contains the full complement of components' memories for all components in V.

We define *AK* to be the set containing all of the adversary's Knowledge. This is further described in Section 5.3.1.

5.2.3 Components and protocols

Within a system, only the components with the correct and designated functionality can perform certain tasks, or certain roles; for example, a random number generator cannot perform the functionality of a keyboard, nor vice-versa. It is however, perfectly possible that a system has multiple or redundant components, e.g., two random number generators, in case one fails, or perhaps because each are connected to different components. We say that within a given system running a protocol, a component and its role are synonymous; that is, each component has its own pre-defined sequence of Component Events (*CompEvent*) within its protocol script which it follows. A protocol is therefore a mapping of components to behaviours or event sequences.

For example, a protocol might state that the Keypad component transmits a value v (e.g., a PIN), which is received as a variable by the CPU; on receipt of this variable, the CPU might then encrypt it with the Bank's public key, and transmit it on to the Bank. Note that this does not allow an arbitrary component to take the role of e.g., the Keypad, CPU, or Bank; this functionality (and consequently the component's role) is determined by the system's 'software', or protocol. Most importantly, an honest component cannot take one protocol role's functionality in one thread, and another's functionality in a different (potentially concurrent) thread.

The adversary can imitate any role or component's behaviour that it wants, but we do not discuss this until Section 5.3.

5.2.3.1 Protocols

A protocol P is a partial function mapping a system's components to both their initial memory and sequences of component events, i.e.,

$$\mathit{Protocol} : \mathit{Component} \nrightarrow \mathcal{P}(\mathit{Term}) \times \mathit{CompEvent}^*$$

The first section of a protocol (an element of $\mathcal{P}(\mathit{Term})$) is a component c's 'initial memory', denoted by $\pi_0(P(c))$ (i.e., $\mathit{Protocol} \times \mathit{Component} \rightarrow \mathcal{P}(\mathit{Term})$). The second section of a protocol, the 'sequence of component events' ($\mathit{CompEvent}^*$), is denoted by $\pi_1(P(c))$ (i.e., $\mathit{Protocol} \times \mathit{Component} \rightarrow \mathit{CompEvent}^*$).

We require that no thread identifiers occur as subterms of events in a protocol definition, or variables within a component's 'initial memory'. As honest components only follow protocol specifications, we do not require that each sequence of component events in the protocol specification is well-formed (i.e., all variables are initialised in an accessible position in a 'recv' event before being used in another event).

The graph G of a system is implicitly defined by the 'recv' operations contained within the protocol. $V (\subseteq \mathit{Component})$ is the set of components named within the protocol, and E is the set of all pairs named in the protocol's recv events. If no recv operation occurs by component B from component A, we can infer that there is no edge between them in the direction $A \rightarrow B$ (or at least not one that is used, nor, importantly, one which an adversary can use).

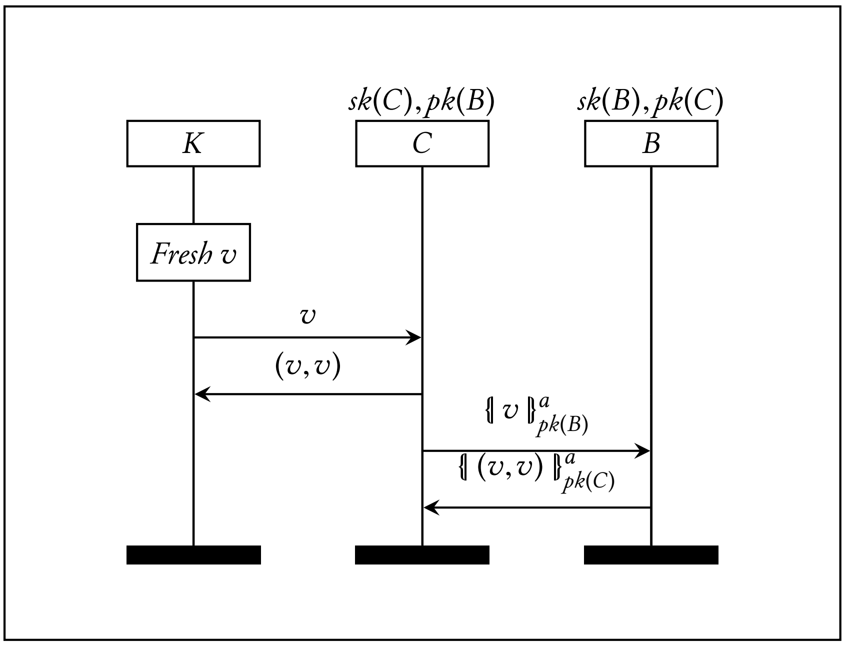


Figure 5.2: Message Sequence Chart of the Example Protocol

Example Protocol: We now give an example of a simple protocol *EP*, which executes across a system of three components. We define the components of the system by the ordered tuple (K, C, B) s.t.

$$V = \{K, C, B\},\ Fresh = \{v\},\ \text{and}\ Var = \{X, Y\}$$

Intuitively, if this system were a simplistic ATM, the component K becomes the Keypad, C the CPU, and B the Bank. Then each component's role within the protocol proceeds as follows, starting with the statement of the component's initial memory, $(\pi_0(EP(a)),\ a \in \{K, C, B\})$ succeeded by the sequence of component events it follows:

$$\begin{aligned}
EP(K) &= \{K, C, B, v\},\\
&\quad \langle \text{send}((K,C), v), \text{recv}((C,K), (v,v))\rangle\\
EP(C) &= \{K, C, B, sk(C), pk(C), pk(B)\},\\
&\quad \langle \text{recv}((K,C), X), \text{send}((C,K), (X,X)),\\
&\quad\ \text{send}((C,B), \{\!| X |\!\}^a_{pk(B)}), \text{recv}((B,C), \{\!| X, X |\!\}^a_{pk(C)})\rangle\\
EP(B) &= \{K, C, B, sk(B), pk(C), pk(B)\},\\
&\quad \langle \text{recv}((C,B), \{\!| Y |\!\}^a_{pk(B)}), \text{send}((B,C), \{\!| Y, Y |\!\}^a_{pk(C)})\rangle
\end{aligned}$$

See Figure 5.2 for the message sequence chart of this protocol. As we know the graph of the system $G = (V, E)$ is defined by the 'recv' messages contained within the protocol, we can deduce that the graph of the system defined by the Example Protocol is as depicted in Figure 5.3.

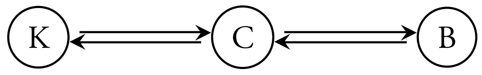


Figure 5.3: Example ATM system implicitly defined by Example Protocol.

Example 5.2.3.1 (Extraction of initial memory and event sequences from a protocol)**.** Within 'Example Protocol', component K's initial memory is:

$$\pi_0(EP(K)) = \{K, C, B, v\}$$

(where K, C, and B are the components' names), and K's event sequence section is:

$$\pi_1(EP(K)) = \langle \mathrm{send}((K,C),v), \mathrm{recv}((C,K),(v,v))\rangle$$

5.2.3.2 Threads

Each instance of a component executing a series of events with other components is called a thread. We distinguish between the fresh terms and variables of each thread by assigning them unique names, using the function *localise* : TID → $\mathcal{S}ub$.

Definition 5.2.3.2 (Localise)**.** Let *tid* ∈ TID. Then

$$localise(tid) = \bigcup_{cv \in Fresh \cup Var} [cv^{tid}/cv]$$

This renames local fresh terms and variables in each thread to universally distinguishable fresh terms and variables.

Using *localise* we define a function *thread* : $(CompEvent^* \times \mathrm{TID}) \rightarrow CompEvent^*$. Applying the *localise* substitution in this function then gives the sequence of events which may happen in a thread, where all fresh terms and variables in that sequence are now explicitly bound to the named thread.

Definition 5.2.3.3 (Thread)**.** Let l be a sequence of events, *tid* ∈ TID. Then:

$$thread(l, tid) = localise(tid)(l)$$

We define *th* to be a partial function mapping thread identifiers of initiated threads to sequences of component events, that is *th* : $(\mathrm{TID} \nrightarrow CompEvent^*)$. This can be thought of as the system's program counter. A thread is not dissimilar to the notion of a 'strand' from Strand Spaces [175], inasmuch as a thread models the actions of an individual actor (or adversary) in a single execution of a protocol: this is distinct from the overall protocol 'trace' (cf. a 'bundle'; we define traces in Section 5.3.3), as each trace consider a whole series of interactions between multiple parties and the adversary.

5.2.4 Execution model

We now describe the execution model, the initial and dynamic states of a system and its components, and the different actions available to a system within this model. We define the execution of a system by a labelled transition system,

$$(State, TraceEvent, \rightarrow, IS(P))$$

We now define each of these elements in turn.

Definition 5.2.4.1 (State)**.** The state of a system at any point is described by the four-tuple (Net, CM, AK, th), i.e.,

$$\begin{aligned} State = {} & ((Component \times Component) \rightarrow {}^{\sharp}Term) \times (Component \rightarrow \mathcal{P}(Term)) \\ & \times \mathcal{P}(Term) \times (\mathrm{TID} \nrightarrow CompEvent^{*}) \end{aligned}$$

whose constituent elements are the buffers *Net* and components' memory *CM* (both defined in Section 5.2.2), the adversary's knowledge *AK* (to be defined in Section 5.3), and the partial function *th* acting as the 'program counter', defined in Section 5.2.3.2.

Definition 5.2.4.2 (Trace Events)**.** A *TraceEvent* is an *Event* paired with a specific thread ID (i.e., that of the thread that executed it). That is:

$$TraceEvent = (\mathrm{TID} \times Event)$$

Each finite sequence of *TraceEvent*s captures an execution history. We will see in Section 5.2.4.1 that transitions between states (e.g., S and S') are labelled by *TraceEvent*s, e.g.,

$$S \xrightarrow{(1,\, \mathrm{send}((a,b),m))} S'$$

Definition 5.2.4.3 (Transition Relation)**.** $\rightarrow$: $State \times TraceEvent \times State$ is a ternary transition relation, where $S \xrightarrow{TrEv} S'$ is syntactic sugar for $(S, TrEv, S') \in \rightarrow$.

The rules in Section 5.2.4.1 more fully define the transition relation; each of these rules have a series of premises which must be met before the conclusion of the rule proceeds. The conclusion of each of these rules is of the form $S \xrightarrow{TrEv} S'$. This new state, S', will contain potentially updated buffers (*Net*), components' memory (*CM*), adversary knowledge (*AK*), and 'program counter' (*th*), depending on the rule that was executed.

Definition 5.2.4.4 (Initial System State)**.** For a protocol P, the initial system state $IS(P)$ is defined to be:

$$IS(P) = (Net_0, CM_0, AK_0(P), \varnothing)$$

All buffers in *Net* are initially empty, that is, we simply map all of the graph's edges to the empty multiset. We denote this initial state of the buffers by Net_0:

$$Net_0 = \{(a,b) \mapsto {}^{\sharp}\varnothing \mid (a,b) \in E\}$$

The overall collection of component memory *CM* is also initially empty, thus mapping each component to the empty set. We denote this initial state of component memory by CM_0:

$$CM_0 = \{a \mapsto \varnothing \mid a \in V\}$$

Each component's memory is initially empty. As soon as a thread is initialised (using the 'create' rule) for a particular component and protocol (e.g., $\text{create}_P(a)$), that component will gain knowledge of the terms specified in the first part of its protocol script, e.g., $\pi_0(P(a))$, including insantiated fresh terms. We do not add the terms automatically at the instantiation of the overall system, as the creation of each new thread may contain fresh terms.

The adversary's initial knowledge (on a protocol P) is $AK_0(P)$, and is defined in Section 5.3.1. In brief, it is defined to be all public keys and names of components. The 'program counter' *th* starts with empty domain, $\varnothing$.

We define Traces in Section 5.3.3, after we have defined the full set of actions available to both honest components and the adversary.

5.2.4.1 Component operational semantics rules

We now describe the operational semantics rules which allow the system to progress from one state to another.

Definition 5.2.4.5 (Create)**.** We create a new session or thread for component c in the protocol P as follows, requiring that the chosen thread ID *tid* is not currently in use:

$$\frac{c \in dom(P) \qquad tid \notin dom(th) \qquad l = thread(\pi_1(P(c)), tid)}{(Net, CM, AK, th) \xrightarrow{(tid,\ \text{create}_P(c))} (Net, CM[c \mapsto CM(c) \cup \pi_0(P(c))], AK, th[tid \mapsto l])}\ [\text{create}_P]$$

The conclusion of this rule is that a new thread is created, the component c gains knowledge of the terms specified in the 'initial memory' of the protocol, or $\pi_0(P(c))$, and the component's 'program counter' now maps to the sequence of component events defined by $thread(\pi_1(P(c)), tid)$. This event has the *TraceEvent* $(tid,\ \text{create}_P(c))$.

After a thread has been created, if the next *CompEvent* within the sequence of events (l) in the protocol script requires it, a component will send a message to another, using the $\text{send}((a,b),m)$ operation. This adds the term m to the buffer on edge (a,b), i.e., it updates the function *Net* as follows: $Net[(a,b) \mapsto Net(a,b) \ {}^\sharp\cup \{m\}]$. We do not care about the ordering of messages within a buffer, as we will use pattern matching to determine which messages are received at which point.

Definition 5.2.4.6 (Send)**.** The send rule is therefore:

$$\frac{th(tid) = \langle \text{send}((a,b),m)\rangle \hat{}\, l}{(Net, CM, AK, th) \xrightarrow{(tid,\ \text{send}((a,b),m))} (Net[(a,b) \mapsto Net(a,b) \ {}^\sharp\cup \{m\}], CM, AK, th[tid \mapsto l])}\,[\text{send}]$$

We might require that a component is able to infer a term m before it can send it as a pre-condition, but as any honest component should only ever be following a protocol script, this should be implicit. Verification that this condition holds can occur later as a check for well-formedness.

Before a component can perform a receive ('recv') action, there must be a term m in the edge's buffer. We use substitutions to allow for the transmission of variables and fresh values: for the recv rule we define $m = \sigma(pt)$, where the pattern pt may contain free variables. Then the $\text{recv}((a,b),pt)$ operation proceeds only with messages that match the pattern pt. On success, the recv operation removes the term m from the buffer on edge (a,b); that is, it updates the function *Net* as follows: $Net[(a,b) \mapsto Net(a,b) \ {}^\sharp\setminus \{m\}]$.

Definition 5.2.4.7 (Receive)**.** The receive rule is therefore:

$$\frac{dom(\sigma) = FV(pt) \qquad m = \sigma(pt) \qquad m \in Net(a,b) \qquad th(tid) = \langle \text{recv}((a,b),pt)\rangle \hat{}\, l}{\begin{array}{l}(Net, CM, AK, th) \xrightarrow{(tid,\ \text{recv}((a,b),m))} (Net[(a,b) \mapsto Net(a,b) \ {}^\sharp\setminus \{m\}], \\ \qquad\qquad CM[CM(b) \cup \{m\}], AK, th[tid \mapsto \sigma(l)])\end{array}}\,[\text{recv}]$$

We use 'claim' and 'running' events to mark and describe the desired security properties of protocols. The purpose and use of these events are described in Section 5.5.

Definition 5.2.4.8 (Claim)**.** The premises of the 'claim' rule requires that a thread has a claim event as its next step within its protocol definition. The conclusion does not affect the state of the system, except to progress the thread. The form that claim events take, and the different claim-types, are both described in Section 5.5.

$$\frac{\begin{array}{c} e = \mathsf{claim}_\ell(c, \mathsf{claim\text{-}type}, t) \lor \\ e = \mathsf{claim}_\ell(c, \mathsf{claim\text{-}type}, c', t)\end{array} \qquad th(tid) = \langle e\rangle \hat{}\, l}{(Net, CM, AK, th) \xrightarrow{(tid,\ e)} (Net, CM, AK, th[tid \mapsto l])}\,[\text{claim}]$$

Intuitively, a claim event is from a single component's point of view. It makes a statement either about the component's view of the system, or of the component's view of a specific term.

Definition 5.2.4.9 (Running)**.** Similar to the 'claim' rule, the premises of the 'running' rule requires that a thread has a running event as its next step within its protocol definition. This is merely a bookkeeping event to help with evaluating the correctness of agreement 'claim' rule instances. The conclusion does not affect the state of the system, except to progress the thread. The form that running events take are also described in Section 5.5.

$$\frac{e = \mathsf{running}_\ell(c,t) \qquad th(tid) = \langle e\rangle \hat{}\, l}{(Net, CM, AK, th) \xrightarrow{(tid,\, e)} (Net, CM, AK, th[tid \mapsto l])}\,[\text{running}]$$

An event $\mathsf{running}_\ell(c,t)$ simply indicates that at the point of execution, component c is running the protocol, and stakes term t into the trace, potentially indicating belief about another component's identity, or a term's value. This term t (mirrored in the claim event) could of course be a tuple, e.g., $t = <\ 'SessKey', x\ >$, perhaps indicating that the claim-making component (c) believes the *SessKey* in the protocol is of ground value x. This 'running' event can then be referred to in authentication properties, and compared to the relevant agreement claim event.

5.2.4.2 Execution notation

Functions: We overload the empty-set ($\varnothing$) to denote a function with empty domain, e.g., $th = \varnothing$ says that th has empty domain. For the instantiation of a function such as th, which is defined element-wise, (remembering that its signature is $th : (\mathrm{TID} \nrightarrow CompEvent^*)$, mapping thread IDs to sequences of component events), we use a dictionary-style notation, where for example $\{0 \mapsto l\}$ maps $tid = 0$ to the sequence of events l.

The function *Net* is also defined element-wise, and we recall that its signature is

$$Net : (Component \times Component) \to {}^\sharp Term$$

mapping pairs of components to multisets of ground terms. *Net* never has empty domain after the initialisation of the system, as its initial state (defined more fully in Definition 5.2.4.4), mapping each edge (or valid pair of components) to the empty multiset, is defined statically as Net_0:

$$Net_0 = \{(a,b) \mapsto {}^\sharp\varnothing \mid (a,b) \in E\}$$

Then, for other values of *Net*, we give the update of Net_0, mapping individual pairs to their multiset of terms. For example,

$$Net_0[(c,k) \mapsto {}^\sharp\left\{v^0\right\}, (b,c) \mapsto {}^\sharp\left\{x^1\right\}]$$

maps the edge (c,k) to the multiset containing the term v^0, and the edge (b,c) to the multiset containing the term x^1. Given $(k,c) \in E$, we can see that in this instance $(k,c) \mapsto {}^\sharp\varnothing$ (i.e., the empty multiset), and given $(x,y) \notin E$, we recall that $(x,y) \mapsto {}^\sharp\{\bot\}$, i.e., the non-existent buffer.

Note that as Net_0 is static, its update (e.g., $Net_0[(C,K) \mapsto {}^\sharp\{v^0\}]$) does not 'stick'. If, say, in step 4 of an execution the state of *Net* within the tuple is the same as Net_0, in step 5 it is $Net_0[(C,K) \mapsto {}^\sharp\{v^0\}]$, and then in step 6 it is then defined to be Net_0 again, this means that as of step 6, (C,K) maps to ${}^\sharp\varnothing$, **not** to ${}^\sharp\{v^0\}$.

A very similar situation is the case for *CM*, except that it maps a single component to a a set, not a pair of components to a multiset.

Sequence notation: We use Python-style slicing notation to allow us to make 'progress' through a sequence of events, by referring to the tail of a sequence, i.e., removing events from the beginning of a sequence. For a sequence $S = \langle s_0, \ldots, s_{n-1}\rangle$ of length $n = |S|$, we refer to the sequence of events starting at position i (with term s_i) by:

$$S[i:] = \langle s_0, \ldots, s_{n-1}\rangle[i:] = \langle s_i, \ldots, s_{n-1}\rangle$$

For all sequences, we observe that

$$S[0:] = S$$

and where $n = |S|$ that

$$S[n:] = \langle\rangle$$

The 'create' rule for a component c (on a protocol) assigns a sequence of events to the variable l; we therefore say $l_{tid}[0:]$ denotes the sequence of events assigned to l in thread *tid*, from the first event in the sequence (at position 0) to the end of the sequence ($|l_{tid}|$ in total). Then $l_{tid}[1:]$ denotes the same sequence l for thread *tid* as before, but starting after (and therefore excluding) the first event, and $l_{tid}[|l_{tid}|:]$ denotes the end of the sequence of events, or the empty sequence, $\langle\rangle$.

Example 5.2.4.10 (Sequence Notation)**.** For a thread with $tid = 3$, executing the role of component B in Example Protocol (defined in Section 5.2.3.1), where $|l_3| = 2$:

$$\begin{aligned}
l_3[0:] &= \langle \mathrm{recv}((C,B), \{\!|\, Y^3 \,|\!\}^a_{pk(B)}), \mathrm{send}((B,C), \{\!|\, (Y^3, Y^3) \,|\!\}^a_{pk(C)})\rangle \\
l_3[1:] &= \langle \mathrm{send}((B,C), \{\!|\, (Y^3, Y^3) \,|\!\}^a_{pk(C)})\rangle \\
l_3[2:] &= \langle\rangle
\end{aligned}$$

Substitutions: As before, we overload $\sigma = \emptyset$ to denote that σ does not contain any substitutions. We denote the n^{th} substitution within an execution of a system as σ_n. The substitution indicated is then a mapping from one term (often a variable) to another (often a ground term). For example, $\sigma_1 = [v^0, x^2 / X^1, Y^1]$ is the substitution which maps X^1 to v^0, and Y^1 to x^2.

Adversary: We describe the adversary in more detail in Section 5.3, but it suffices for now to say that the adversary's knowledge is strictly monotonically increasing, as once the adversary has learnt a term, nothing will convince them to forget it. Because of this, we record the adversary's knowledge as the difference in terms known to the adversary before and after an event, denoted by ΔAK.

5.2.5 Example Protocol execution

Example 5.2.5.1 (Execution of the Example Protocol)**.** We re-state the protocol Example Protocol (EP) from Section 5.2.3.1, including each component $a \in V$'s initial memory $\pi_0(EP(a))$, and the sequence of events the component will execute, $\pi_1(EP(a))$.

We choose the ordered tuple of components (K,C,B) s.t. $V = \{K,C,B\}$, the fresh term(s) $Fresh = \{v\}$ and the variables $Var = \{X,Y\}$.

$$\begin{aligned}
EP(K) = (&\{K,C,B,v\},\\
&\langle \text{send}((K,C),v), \text{recv}((C,K),(v,v))\rangle)\\
EP(C) = (&\{K,C,B,sk(C),pk(C),pk(B)\},\\
&\langle \text{recv}((K,C),X), \text{send}((C,K),(X,X)),\\
&\text{send}((C,B),\{\!|\, X \,|\!\}^a_{pk(B)}), \text{recv}((B,C),\{\!|\, (X,X) \,|\!\}^a_{pk(C)})\rangle)\\
EP(B) = (&\{K,C,B,sk(B),pk(C),pk(B)\},\\
&\langle \text{recv}((C,B),\{\!|\, Y \,|\!\}^a_{pk(B)}), \text{send}((B,C),\{\!|\, (Y,Y) \,|\!\}^a_{pk(C)})\rangle)
\end{aligned}$$

We now give an example execution, listing all the intermediate states and variables. Please see Figure 5.4.

i	*TraceEvent*: (TID × Event)	Net *after* event.	ΔAK	th *after* event.	$\Delta\sigma$
0	$\varnothing$	Net_0	$V \cup \{pk(C), pk(B)\}$	$\varnothing$	
1	$(0, \mathrm{create}_{EP}(K))$	Net_0		$\{0 \mapsto l_0[0\!:]\}$	
2	$(0, \mathrm{send}((K,C), v^0))$	$Net_0[(K,C) \mapsto {}^{\sharp}\{v^0\}]$		$\{0 \mapsto l_0[1\!:]\}$	
3	$(1, \mathrm{create}_{EP}(C))$	$Net_0[(K,C) \mapsto {}^{\sharp}\{v^0\}]$		$\{0 \mapsto l_0[1\!:],$ $1 \mapsto l_1[0\!:]\}$	
4	$(1, \mathrm{recv}((K,C), v^0))$	Net_0		$\{0 \mapsto l_0[1\!:],$ $1 \mapsto \sigma_1(l_1[1\!:])\}$	$\sigma_1 = [v^0/X^1]$
5	$(1, \mathrm{send}((C,K), (v^0, v^0)))$	$Net_0[(C,K) \mapsto {}^{\sharp}\{(v^0, v^0)\}]$		$\{0 \mapsto l_0[1\!:],$ $1 \mapsto \sigma_1(l_1[2\!:])\}$	
6	$(0, \mathrm{recv}((C,K), (v^0, v^0)))$	Net_0		$\{0 \mapsto \langle\rangle,$ $1 \mapsto \sigma_1(l_1[2\!:])\}$	
7	$(1, \mathrm{send}((C,B), \{\!\mid v^0 \mid\!\}^a_{pk(B)}))$	$Net_0[(C,B) \mapsto$ ${}^{\sharp}\{\{\!\mid v^0 \mid\!\}^a_{pk(B)}\}]$		$\{0 \mapsto \langle\rangle,$ $1 \mapsto \sigma_1(l_1[3\!:])\}$	
8	$(2, \mathrm{create}_{EP}(B))$	$Net_0(C,B) \mapsto$ ${}^{\sharp}\{\{\!\mid v^0 \mid\!\}^a_{pk(B)}\}]$		$\{0 \mapsto \langle\rangle,$ $1 \mapsto \sigma_1(l_1[3\!:]),$ $2 \mapsto l_2[0\!:]\ \}$	
9	$(2, \mathrm{recv}((C,B), \{\!\mid v^0 \mid\!\}^a_{pk(B)}))$	Net_0		$\{0 \mapsto \langle\rangle,$ $1 \mapsto \sigma_1(l_1[3\!:]),$ $2 \mapsto \sigma_2(l_2[1\!:])\}$	$\sigma_2 = [v^0/Y^2]$
10	$(2, \mathrm{send}((B,C), \{\!\mid (v^0, v^0) \mid\!\}^a_{pk(C)}))$	$Net_0[(B,C) \mapsto$ ${}^{\sharp}\{\{\!\mid (v^0, v^0) \mid\!\}^a_{pk(C)}\}]$		$\{0 \mapsto \langle\rangle,$ $1 \mapsto \sigma_1(l_1[3\!:]),$ $2 \mapsto \langle\rangle\ \}$	
11	$(1, \mathrm{recv}((B,C), \{\!\mid (v^0, v^0) \mid\!\}^a_{pk(C)}))$	Net_0		$\{0 \mapsto \langle\rangle,\ 1 \mapsto \langle\rangle,$ $2 \mapsto \langle\rangle\ \}$	

Figure 5.4: Sample execution of Example Protocol.

5.3 The adversary

We have seen an execution of a system *not* in the presence of an adversary; we now discuss the introduction of an adversary to a modelled system, and how it can interact with it.

Providing fine-grained, very precise control of adversarial capabilities is the purpose of this framework. Defending against a powerful Dolev-Yao adversary (or stronger) is desirable in many cases, but is also sometimes overly restrictive and unrealistic. We believe this to be especially true of partially compromised component-based systems, compared to e.g., internet key exchange protocols [54]. Here, we consider adversarial control or compromise on a per-buffer and per-component basis, rather than assuming all communications buffers are automatically controlled by the adversary. We see these descriptions of per-buffer (network channel) and per-component compromise as a natural way of generalising the case-study specific partial compromise actions described and used in Chapters 3 and 4.

We proceed as follows: first, in Section 5.3.1 we define an adversary's initial knowledge with regard to a protocol, followed in Section 5.3.2 by the range of specific actions ('adversary events') that an adversary might be able to use against specific components and/or buffers within a system. We describe Traces in Section 5.3.3, and then we introduce the running example of the Split Protocol in Section 5.3.4, giving a sample execution of a protocol in the presence of an adversary. Having described the range of possible adversary actions in Section 5.3.2, we then describe further tools and methodologies for restricting when and in which contexts these might be used within the execution of a protocol. These form more concrete threat models, and are described in Section 5.4. To combine the 'honest' execution model with the specific threat posed by an adversary of a certain strength or capability, we describe the total set of Valid Traces a system in the presence of a specific adversary might generate in Section 5.4.4. We then discuss and define Claim Events and Security Properties within protocols in Section 5.5, such as secrecy and authentication (in Sections 5.5.1 and 5.5.2). Finally, we define the correctness of a protocol's stated security properties in the form of claim events in Section 5.5.3. Our running example of the Split Protocol is referenced throughout, giving a worked example of one way in which an adversary might violate the secrecy of a term, demonstrating the incorrectness of the protocol's security claim event.

5.3.1 Initial adversary knowledge

The **adversary's knowledge** is one large set of terms, AK. This is in contrast to the way we describe the terms in specific buffers and components. The adversary's knowledge doesn't distinguish between from which buffer or component it originally learnt a term; intuitively, this is because any term the adversary learns from one component or buffer can potentially be replayed back or inserted into any other component or buffer under its control. Initially, the adversary knows the overall system protocol P, (and can therefore infer the system's layout), and the names and public keys (where they exist) of all components.

Definition 5.3.1.1 (Adversary Initial Knowledge)**.** For a protocol P, the adversary's initial knowledge $AK_0(P)$ is defined as follows:

$$AK_0(P) = V \cup \left\{ pk(a) \mid a \in V \right\}$$

At this stage, many adversary models would also add terms from permanently compromised actors or components to the 'initial adversary knowledge'. We *do not* model permanent compromises in this way, instead allowing an adversary to perform certain actions on a component (e.g., learning the contents of its memory) during the execution of a system, and in the case of a permanently compromised component, from the moment of creation of a particular component's thread. These component compromise actions are described in more detail in Section 5.3.2.2.

5.3.2 Adversary events

An adversary *Adv* can then perform a range of actions on certain components and buffers, to learn additional terms, and to gain some level of compromise over a system. These actions include:

- Reading the contents of a buffer.
- Inserting into, modifying, or deleting terms from a buffer.
- Reading the memory of a component (this may well include secret keys).
- Inserting terms into the the memory of a component.

We emphasize that this (and the following formal definitions) is merely a list of *possible* adversary actions available to an adversary. Which of these actions it is actually allowed to perform (or are considered valid) in any particular execution depends on the specific threat model, but this range of events serves to illustrate the power of the compromise actions which might be available to an adversary.

We define AA ('Adversary Allowed') to be the set of adversary events from $AdvEvent$ which are allowed *within a particular execution*, that is $AA \subseteq AdvEvent$.

5.3.2.1 Buffer compromise

Most basically, an adversary can read the contents of a buffer, by performing the **Buffer Reveal** (BR) action. This takes all the terms within a buffer between two components at a certain stage of the execution of the thread, and copies these terms into the adversary's knowledge. Note that without any clearly defined threat model to restrict this, an adversary can potentially perform this action as many times as it likes, even on all components and buffers within a system at every step if so desired. If performed on all buffers at every step, this would mimic a Dolev-Yao adversary's knowledge. These buffer compromise actions are directly comparable to the 5G secure channel compromise actions described in Section 4.13.1.

Definition 5.3.2.1 (Buffer Reveal)**.** The Buffer Reveal rule is therefore:

$$\frac{}{(Net, CM, AK, th) \xrightarrow{(Adv,\, \mathrm{BR}(a,b))} (Net, CM, AK \cup Net(a,b), th)}\,[\mathrm{BR}]$$

Buffer Reveal is a passive adversary action, as it does not have any direct impact on or change to any honest components or buffers: BR $\in$ *PassiveAdvEvent*.

An adversary's 'Buffer Reveal' action on a buffer is a single specific action at that point in the traces, not continual access to any buffers, due to the non-DY nature of our adversary. This action can be repeated multiple times to gain 'continual' access.

Another possible adversary action is the ability to inject any term that it knows (or can infer) into any buffer. The resultant system state after this action is similar to the result of the honest component's send action.

Definition 5.3.2.2 (Buffer Inject)**.** The Buffer Inject rule is therefore:

$$\frac{AK \vdash t}{(Net, CM, AK, th) \xrightarrow{(Adv,\, \mathrm{BI}((a,b),t))} (Net[(a,b) \mapsto Net(a,b) \,{}^{\sharp}\!\cup \{t\}], CM, AK, th)}\,[\mathrm{BI}]$$

Buffer Inject is an active adversary action, as it actively modifies a buffer: BI $\in$ *ActiveAdvEvent*.

An adversary can also potentially delete terms from a buffer before they are delivered to the recipient component.

Definition 5.3.2.3 (Buffer Delete)**.** The Buffer Delete rule is therefore:

$$\frac{t \in Net(a,b)}{(Net, CM, AK, th) \xrightarrow{(Adv,\, \mathrm{BD}((a,b),t))} (Net[(a,b) \mapsto Net(a,b) \,{}^{\sharp} \setminus \{t\}], CM, AK, th)}\,[\mathrm{BD}]$$

Buffer Delete is an active adversary action, as it actively modifies a buffer: BD $\in$ *ActiveAdvEvent*.

We observe that through the combination of Buffer Reveal, Buffer Delete, and Buffer Inject the adversary can also modify or replace arbitrary values within a buffer.

5.3.2.2 Component compromise

When an adversary compromises a component, it can either simply read the contents of the component's memory, or inject new terms. As with buffers, we say that the former is a type of passive compromise, and the latter active compromise. Another form of active compromise not previously used is that of modifying the script of the component, such that it performs a different sequence of events to the one it should do according to its original protocol script. These component compromise actions are directly comparable to DNP3 partial compromise as described in Section 3.5, and the 5G component compromise actions described in Section 4.13.2.

Definition 5.3.2.4 (Component Memory Reveal)**.** The Component Memory Reveal rule is therefore:

$$\frac{}{(\mathit{Net}, \mathit{CM}, \mathit{AK}, \mathit{th}) \xrightarrow{(\mathit{Adv},\, \mathrm{CMR}(a))} (\mathit{Net}, \mathit{CM}, \mathit{AK} \cup \mathit{CM}(a), \mathit{th})}\,[\mathrm{CMR}]$$

Similar to Buffer Reveal, this is a passive action, as it does not have any direct impact on or change to any honest components or buffers: CMR $\in$ *PassiveAdvEvent*.

As with Buffer Reveal, this is a single action in the traces; we can simulate an adversary having permanent compromise of a component by allowing it to perform the CMR rule at each step in the (perhaps otherwise honest) execution of the system. An adversary can also inject and delete terms from a component's memory.

Definition 5.3.2.5 (Component Memory Inject)**.** The Component Memory Inject rule is therefore:

$$\frac{\mathit{AK} \vdash t}{(\mathit{Net}, \mathit{CM}, \mathit{AK}, \mathit{th}) \xrightarrow{(\mathit{Adv},\, \mathrm{CMI}(a,t))} (\mathit{Net}, \mathit{CM}[a \mapsto \mathit{CM}(a) \cup \{t\}], \mathit{AK}, \mathit{th})}\,[\mathrm{CMI}]$$

Component Memory Inject is an active adversary action: CMI $\in$ *ActiveAdvEvent*.

The following table compares passive and active adversary compromise of both buffers and components.

	PassiveAdvEvent	*ActiveAdvEvent*
Buffer (*Net*)	BR	BI, BD
Component (*CM*)	CMR	CMI

Table 5.1: Passive and active adversary events

Now we describe adversary controlled component a sending the term t to the buffer between it and component b.

Definition 5.3.2.6 (Adversary Send)**.** The adversary send rule is therefore:

$$\frac{AK \vdash t}{(Net, CM, AK, th) \xrightarrow{Adv, \text{adv-send}((a,b),t)} (Net[(a,b) \mapsto Net(a,b) \,^{\sharp}\!\cup \{t\}], CM, AK, th)} \text{[adv-send]}$$

Any term the adversary can derive can be sent to a buffer to which it has access. The main distinction between this and the adversary's Buffer Inject operation is that Buffer Inject requires compromise of the buffer itself, not one of the end-point components; certain threat models might not allow compromise of certain buffers, but may allow compromise of the relevant component.

5.3.2.3 Continuous compromise

As described, all of these adversary compromise operations are individual operations performed at a particular point in a trace, on what is effectively a snapshot of the system; we can simulate continuous adversary compromise of a particular component or buffer by allowing the adversary to perform a 'one-off' operation on the same buffer or component after *every* honest operation or step of the protocol.

5.3.3 Traces

We have now defined the actions available to both honest components and the adversary, and seen how both can be used. From this situation, we can more easily define the whole execution of a system, and some tools we will use to analyse that execution.

Given a protocol and an adversary, we can define a system's possible behaviours either as a set of reachable states or a set of traces. For many of the security properties with which we are concerned, trace properties are more useful than state properties: while a system may reach a particular state via many different routes, each trace is unique.

Definition 5.3.3.1 (Finite execution of a system)**.** Let P be a protocol, and AA be a set of allowed adversary events (recalling that $AA \subseteq AdvEvent$, as defined in Section 5.3.2).

A finite execution $\mathcal{X}$ of a system with protocol P and allowed adversary actions AA is of the form:

$$\mathcal{X}_{P,AA} = [s_0, \alpha_1, s_1, \alpha_2, \ldots, \alpha_n, s_n]$$

where $s_0 = IS(P)$, s_i are system states of the form (Net, CM, AK, th), and α_i are *TraceEvents* of the form $(\text{TID}, Event)$. We then require that for all $i < n$, we have that $s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ is either an instance of a rule in the system's execution rules, or an instance of a rule in the allowed adversary events, AA, where the rule has the conclusion $s \xrightarrow{\alpha_{i+1}} s'$ such that all the premises hold.

Definition 5.3.3.2 (A single trace)**.** A trace, *tr*, is simply defined to be an ordered sequence of *TraceEvent*s (whose index starts at 0) representing the execution of a protocol.

Definition 5.3.3.3 (The set *Trace*)**.** If a finite execution of a system with protocol P and allowed adversary actions AA is of the form

$$\mathcal{X}_{P,AA} = [s_0, \alpha_1, s_1, \alpha_2, \ldots, \alpha_n, s_n]$$

then the corresponding trace of that execution would be $tr = [\alpha_1, \alpha_2, \ldots, \alpha_n]$. We therefore define $Trace(P, AA)$ to be the set of finite traces of the labelled transition system associated with the execution $\mathcal{X}_{P,AA}$ of a protocol P and allowed adversary actions AA. Therefore $tr \in Trace(P, AA)$ iff tr is a trace of an execution of a system under P, AA.

For every finite trace $tr = [\alpha_1, \alpha_2, \ldots, \alpha_n]$ of a transition system, there is a unique finite execution $\mathcal{X}_{P,AA}$. This means that from any point in any finite trace, we can deduce the exact system state, including the adversary's knowledge (the set AK).

5.3.4 Split Protocol

Having introduced the adversary, and the full range of possible actions it might take against a hypothetical system, we now combine an explicit set of adversarial capabilities with an actual instance of a system or protocol, in the form of the Split Protocol. This will be used as a running example in the following sections.

Intuitively, the idea of the Split Protocol is that it might be used in a situation where we wish to transmit a secret value between various components or parts of a system, but where one important but low-level component can only perform symmetric encryption, and does not possess a pre-shared secret with their intended communication partner. Some *other* (more powerful) components may have the ability to perform asymmetric key cryptography with pre-shared public keys. The main idea behind this protocol is to leverage multiple paths, such that an adversary needs to compromise more than one path or component to learn the secret value.

We define the components of the system by the ordered tuple (D, K, C, B) s.t.

$$V = \{D, K, C, B\},\ Fresh = \{r, x\},\ \text{and}\ Var = \{U, W, X, Y, Z\}$$

To extend the example given in Example Protocol (Section 5.2.3.1), D could perhaps be a Hardware Security Module (HSM), K a keyboard, C a CPU, and B a Bank. Intuitively, K is trying to transmit the fresh value x to B while maintaining x's confidentiality. This could happen in the situation where K (as a keyboard) is a very low-level device, that:

a) does not possess the ability to perform asymmetric encryption,
b) does not possess the ability to generate randomness, and,
c) does not have a pre-shared symmetric key with B.

Informally, the keyboard component K receives a (fresh) random number r from the HSM D and uses it as a symmetric key. K symmetrically encrypts x with r, and sends it to C. D independently sends the random number r to C (encrypted with B's public key), who then forwards both messages on to B, the Bank. B can now recover the random number r (decrypting asymmetrically with its private key), and uses r to decrypt (symmetrically) the value x.

We then formally define this system's protocol as follows:

$$\begin{aligned}
SP(D) &= (\{K,D,C,B,pk(B),r\},\\
&\qquad \langle \text{send}((D,K),r), \text{send}((D,C),\{\!|\, r \,|\!\}^a_{pk(B)})\rangle)\\
SP(K) &= (\{K,D,C,B,x\},\\
&\qquad \langle \text{recv}((D,K),U), \text{send}((K,C),\{\!|\, x \,|\!\}^s_U),\\
&\qquad \ \mathsf{claim}_\ell(K, secret, x)\rangle)\\
SP(C) &= (\{K,D,C,B,pk(B)\},\\
&\qquad \langle \text{recv}((D,C),\{\!|\, W \,|\!\}^a_{pk(B)}), \text{recv}((K,C),\{\!|\, X \,|\!\}^s_W),\\
&\qquad \ \text{send}((C,B),\{\!|\, W \,|\!\}^a_{pk(B)} \hat{}\, \{\!|\, \{\!|\, X \,|\!\}^s_W \,|\!\}^a_{pk(B)})\rangle)\\
SP(B) &= (\{K,D,C,B,sk(B)\},\\
&\qquad \langle \text{recv}((C,B),\{\!|\, Y \,|\!\}^a_{pk(B)} \hat{}\, \{\!|\, \{\!|\, Z \,|\!\}^s_Y \,|\!\}^a_{pk(B)})\rangle)
\end{aligned}$$

See Figure 5.5 for the message sequence chart of this protocol, and Figure 5.6 for the graph of the components and buffers between them.

We informally stated that the goal of the system is to allow K to transmit a value x to B while maintaining the confidentiality of the value transmitted. This has been stated in the message sequence chart as the claim event 'x secret'; claim events are described fully in Section 5.5. Informally, we mean that the goal of the protocol is that after completing the protocol, from K's point of view the adversary cannot deduce the value x.

Example 5.3.4.1. We now give a sample execution of the Split Protocol, including normal protocol actions from the honest components (*CompEvents*), and actions from the adversary (*AdvEvents*). In this execution, we restrict the adversary to passive actions only, and therefore we say the set Adversary Allowed or $AA = PassiveAdvEvent$. We show in Figure 5.7 that by choosing two particular, well timed actions, the adversary can violate the desired property of secrecy of the transmitted value x.

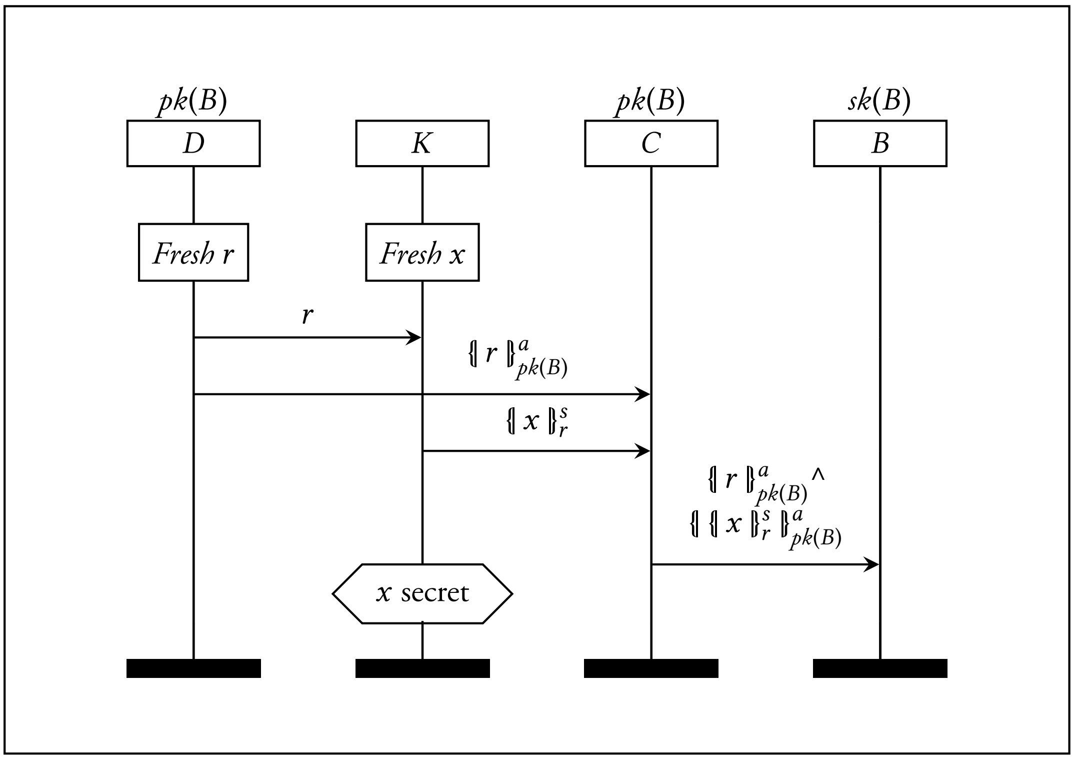


Figure 5.5: Message Sequence Chart of the Split Protocol

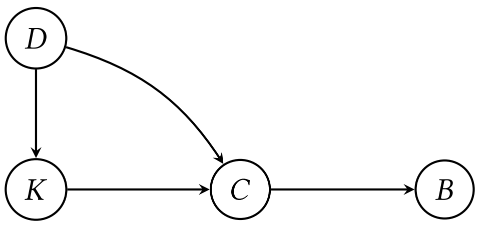

$$\begin{tikzcd}
D \arrow[d] \arrow[dr, bend left] & & \\
K \arrow[r] & C \arrow[r] & B
\end{tikzcd}$$

Figure 5.6: Network configuration of the system for the Split Protocol

i	*TraceEvent*: (TID × Event)	Net *after* event.	ΔAK	th *after* event.	$\Delta\sigma$
0	$\varnothing$	Net_0	$V \cup \{pk(B)\}$	$\varnothing$	
1	$(0, \text{create}_{SP}(D))$	Net_0		$\{0 \mapsto l_0[0:]\}$	
2	$(0, \text{send}((D,K), r^0))$	$Net_0[(D,K) \mapsto {}^{\sharp}\{r^0\}]$		$\{0 \mapsto l_0[1:]\}$	
3	$(Adv, \text{BR}(D,K))$	$Net_0[(D,K) \mapsto {}^{\sharp}\{r^0\}]$	r^0	$\{0 \mapsto l_0[1:]\}$	
4	$(0, \text{send}((D,C), \lbrace\!\vert r \vert\!\rbrace^a_{pk(B)}))$	$Net_0[(D,K) \mapsto {}^{\sharp}\{r^0\}$, $(D,C) \mapsto {}^{\sharp}\{\lbrace\!\vert r^0 \vert\!\rbrace^a_{pk(B)}\}]$		$\{0 \mapsto \langle\rangle\}$	
5	$(1, \text{create}_{SP}(K))$	$Net_0[(D,K) \mapsto {}^{\sharp}\{r^0\}$, $(D,C) \mapsto {}^{\sharp}\{\lbrace\!\vert r^0 \vert\!\rbrace^a_{pk(B)}\}]$		$\{0 \mapsto \langle\rangle$, $1 \mapsto l_1[0:]\}$	
6	$(1, \text{recv}((D,K), r^0))$	$Net_0[(D,C) \mapsto {}^{\sharp}\{\lbrace\!\vert r^0 \vert\!\rbrace^a_{pk(B)}\}]$		$\{0 \mapsto \langle\rangle$, $1 \mapsto \sigma_1(l_1[1:])\}$	$\sigma_1 = [r^0/U^1]$
7	$(1, \text{send}((K,C), \lbrace\!\vert x^1 \vert\!\rbrace^s_{r^0}))$	$Net_0[(D,C) \mapsto {}^{\sharp}\{\lbrace\!\vert r^0 \vert\!\rbrace^a_{pk(B)}\}$, $(K,C) \mapsto {}^{\sharp}\{\lbrace\!\vert x^1 \vert\!\rbrace^s_{r^0}\}]$		$\{0 \mapsto \langle\rangle, 1 \mapsto \langle\rangle\}$	
8	$(Adv, \text{BR}(K,C))$	$Net_0[(D,C) \mapsto {}^{\sharp}\{\lbrace\!\vert r^0 \vert\!\rbrace^a_{pk(B)}\}$, $(K,C) \mapsto {}^{\sharp}\{\lbrace\!\vert x^1 \vert\!\rbrace^s_{r^0}\}]$	$\lbrace\!\vert x^1 \vert\!\rbrace^s_{r^0}$	$\{0 \mapsto \langle\rangle, 1 \mapsto \langle\rangle\}$	

Figure 5.7: Sample execution of Split Protocol in the presence of an adversary.

We stop after step $i = 8$ because the adversary can now infer the localised term x^1 (because $AK \vdash \lbrace\!\vert x^1 \vert\!\rbrace^s_{r^0} \wedge AK \vdash r^0 \Rightarrow AK \vdash x^1$), and thus we have an example of a trace where the adversary has broken the desired property of x's secrecy. If we followed the execution to completion, we would see *TraceEvent* $(1, \mathsf{claim}_\ell(K, secret, x^1))$ in the trace. The adversary achieves this by performing operations on two objects, in this case performing Buffer Reveal operations on the buffers (D,K) and (K,C) in the execution of the system. We discuss restrictions on the type and number of compromises allowed per execution in Section 5.4.

Example 5.3.4.2 (Split Protocol Example Trace)**.** We recall the definition of a trace from Section 5.3.3, and combine it with our example execution of the Split Protocol in Figure 5.7. Recalling that in this example $AA = \mathit{PassiveAdvEvent}$, the individual trace exhibited by the execution $\mathcal{X}_{SP,AA}$ described in Figure 5.7 (up to step $i = 8$) is therefore:

$$\begin{aligned} tr = [&(0, \mathrm{create}_{SP}(D)), (0, \mathrm{send}((D,K), r^0)), (Adv, \mathrm{BR}(D,K)), \\ &(0, \mathrm{send}((D,C), \{\!|\, r \,|\!\}^a_{pk(B)})), (1, \mathrm{create}_{SP}(K)), (1, \mathrm{recv}((D,K), r^0)), \\ &(1, \mathrm{send}((K,C), \{\!|\, x^1 \,|\!\}^s_{r^0})), (Adv, \mathrm{BR}(K,C)), \ldots] \end{aligned}$$

5.4 Threat models and valid traces

Having computed which traces are *possible* during the execution of a system (in Section 5.3.3), we then we cull or thin out the set $\mathit{Trace}(P, AA)$ by establishing which traces we consider to be valid through comparison to an explicit threat model; we throw away those traces which we do not consider to be valid under the chosen threat model. Threat models therefore consist of restrictions on the set $\mathit{Trace}(P, AA)$, and we now elaborate on this further, giving formal definitions of some of the threat models available to us.

A threat model (or specification of adversary power) is defined as a predicate over the traces, determining which traces we allow as valid. The strongest threat model allows the set of all traces allowed directly by the semantics; notably, this includes traces where the adversary performs up to and including *all* valid compromise operations on *all* components and buffers after every honest operation, i.e., at *every* possible step. Then, weaker threat models are those where we consider a strictly smaller subset of allowable traces, restricting the set of valid traces to those where the adversary only performs certain actions (perhaps on certain components or buffers), or only performs certain actions no more than a certain total number of times.

Please note that the following threat models are merely examples, and that many more threat models (as long as they are predicates over traces) can be created to provide fine-grained control over the adversary's allowed capabilities.

After defining a series of possible threat models in Sections 5.4.1 to 5.4.3, we define the idea of Valid Traces in Section 5.4.4, i.e., those traces which are true or 'valid' under a particular threat model.

5.4.1 Component threat models

We now define a series of tools and threat models concerning adversary actions against components.

Definition 5.4.1.1 (Compromised Components)**.** We define the set *ComprComps* to be the set of components against which the adversary has performed one or more compromise actions in the given trace (for $tr \in Trace(P, AA)$).

$$\begin{aligned} ComprComps(tr) = \Big\{ c \;\Big|\; \exists e \in tr \wedge e \in \{ & (Adv, \mathrm{CMR}(c)), \\ & (Adv, \mathrm{CMI}(c, \_)), \\ & (Adv, \text{adv-send}((c, \_), \_))\} \Big\} \end{aligned}$$

Then the number of components compromised in a single trace is simply the number of elements in the set *ComprComps*(*tr*). Note that one component having multiple compromise actions performed on it throughout a run results in the same set *ComprComps* as if the adversary only performed one single compromise action on that component, and no others.

We can then use the number of compromised components to create threat models in the form of predicates, e.g.,

Definition 5.4.1.2 (*TM*: At Most One Compromised Component)**.** This threat model only allows at most one component to be compromised.

$$AtMostOneCC(tr) \iff \left|ComprComps(tr)\right| \leq 1$$

Definition 5.4.1.3 (*TM*: At Most n Compromised Components)**.** This generalises the previous threat model, allowing at most n components to be compromised.

$$AtMostNCC(tr, n) \iff \left|ComprComps(tr)\right| \leq n$$

N.B. This and many other threat models can become predicates upon only a trace (i.e., without the second argument, in this case n), by, for example, hard-coding a value n and wrapping the result in a new predicate, e.g.,

$$AtMost3CC(tr) \iff AtMostNCC(tr, 3)$$

We also allow the exclusion of certain components from compromise, e.g., because we might have formally verified the correctness of these components. We call this set of trusted components upon which the adversary is not allowed to perform any compromise actions *TrComps* ($\subseteq$ *Component*).

Definition 5.4.1.4 (*TM*: No Compromise of Components in *TrComps*)**.**

$$\mathit{NoComprOfSpecificComps}(tr, \mathit{TrComps}) \iff \mathit{ComprComps}(tr) \cap \mathit{TrComps} = \varnothing$$

Example 5.4.1.5 (Trusted Components in the Split Protocol)**.** From manual inspection of our running example, the Split Protocol, we can observe that x's confidentiality cannot possibly be maintained if one of the two components K or B is compromised, as the 'purpose' of the protocol is to transmit the value x from K to B. Therefore, at various stages of the protocol both components must be able to infer the value x directly from their memory.

It would therefore be reasonable to say that in the Split Protocol, we consider and require that K and B are trusted, or $\mathit{TrComps} = \{K, B\}$. This could be an analogue for saying that the code and/or hardware of these components have undergone extra thorough bug checking, or formal verification. We can hard-code the specific components K and B into a new threat model *NoComprOfTrComps* as follows:

$$\mathit{NoComprOfTrComps}(tr) \iff \mathit{ComprComps}(tr) \cap \{K, B\} = \varnothing$$

Similarly we can count and restrict the total number of adversary actions against components, regardless of which component they were performed on. We start by defining the multiset containing all adversary actions in a trace:

Definition 5.4.1.6 (All Adversary Component Actions)**.**

$$\begin{aligned}\mathit{AdvCompActions}(tr) = {}^{\sharp}\Big\{\, tr_i \;\Big|\; 0 \le i \le |tr| \wedge tr_i \in \{&(\mathit{Adv}, \mathrm{CMR}(\_)),\\ &(\mathit{Adv}, \mathrm{CMI}(\_,\_)),\\ &(\mathit{Adv}, \text{adv-send}((\_,\_),\_)),\}\Big\}\end{aligned}$$

Note that this is a multiset, as we need to be able to count (syntactically equivalent) repeated adversary actions at different points in a trace.

Example 5.4.1.7 (Syntactic equivalence of compromise actions)**.** With the adversary actions from a trace *tr*, where:

$$\begin{aligned}\alpha_3 &= (\mathit{Adv}, \mathrm{CMR}(C))\\ \alpha_7 &= (\mathit{Adv}, \mathrm{CMR}(A))\\ \alpha_9 &= (\mathit{Adv}, \mathrm{CMR}(C))\end{aligned}$$

we have the resulting syntactic equalities: $\alpha_3 = \alpha_9$, $\alpha_3 \neq \alpha_7$, $\alpha_7 \neq \alpha_9$. The multiset *AdvCompActions*(*tr*) on this trace would therefore be:

$$^{\sharp}\{(Adv, \mathrm{CMR}(C)), (Adv, \mathrm{CMR}(A)), (Adv, \mathrm{CMR}(C))\}$$

The total number of component compromise actions is simply the cardinality of the multiset.

Definition 5.4.1.8 (*TM*: At Most *n* Adversary Component Actions)**.** We define a predicate with respect to the multiset *AdvCompActions*(*tr*) restricting the total number of adversary actions against components allowed within a single trace:

$$AtMostNACA(tr, n) \iff \left|AdvCompActions(tr)\right| \leq n$$

We form a similar multiset and predicate concerning buffer compromise actions in Definition 5.4.2.5, and then combine these to establish and restrict the total number of adversary actions of any kind.

5.4.2 Buffer threat models

Similar to component threat models, we now define a series of tools and threat models concerning adversary actions against buffers.

Definition 5.4.2.1 (Compromised Buffers)**.** We define the set *ComprBuffers* to be the set of buffers against which the adversary has performed a compromise action in the given trace (for $tr \in Trace(P, AA)$).

$$\begin{aligned} ComprBuffers(tr) = \Big\{ (a,b) \;\Big|\; \exists e \in tr \wedge e \in \{&(Adv, \mathrm{BR}(a,b)),\\ &(Adv, \mathrm{BI}((a,b),\_)),\\ &(Adv, \mathrm{BD}((a,b),\_))\}\Big\} \end{aligned}$$

Then the number of buffers compromised in a single trace is simply the number of elements in the set *ComprBuffers*(*tr*).

Example 5.4.2.2 (Compromised Buffers in the Split Protocol)**.** From our example execution of the Split Protocol in Section 5.3.4, and the resultant trace *tr* given in Example 5.3.4.2, the set $ComprBuffers(tr) = \{(D,K),(K,C)\}$. The cardinality of this set is $\left|ComprBuffers(tr)\right| = 2$.

We also allow the exclusion of certain buffers from compromise. We call this set of trusted buffers upon which the adversary is not allowed to perform any compromise actions *TrBuffers* ($\subseteq (Component \times Component)$).

Definition 5.4.2.3 (*TM*: No Compromise of Buffers in *TrBuffers*)**.**

$$NoComprOfTrBuffers(tr, TrBuffers) \iff ComprBuffers(tr) \cap TrBuffers = \varnothing$$

We can then combine *TrComps* and *TrBuffers* together to form a set $Trusted = TrComps \cup TrBuffers$, i.e., those components and buffers which we declare the adversary may not compromise during a run.

Definition 5.4.2.4 (All Adversary Buffer Actions)**.** As with Adversary Component Actions, we define the multiset *AdvBufferActions* to be all adversary actions against buffers within a single trace:

$$\begin{aligned} AdvBufferActions(tr) = {}^{\sharp}\Big\{ tr_i \;\Big|\; 0 \le i \le \big|tr\big| \wedge tr_i \in \{ & (Adv, \mathrm{BR}(\_,\_)), \\ & (Adv, \mathrm{BI}((\_,\_),\_)), \\ & (Adv, \mathrm{BD}((\_,\_),\_))\}\Big\} \end{aligned}$$

Note that as with *AdvCompActions*, this is a multiset, as we need to be able to count (syntactically equivalent) repeated adversary actions at different points in a trace.

Definition 5.4.2.5 (*TM*: At Most *n* Adversary Buffer Actions)**.** We define a predicate with respect to the multiset $AdvBufferActions(tr)$ restricting the total number of adversary actions against buffers that are allowed within a single trace:

$$AtMostNABA(tr, n) \iff \big|AdvBufferActions(tr)\big| \le n$$

Example 5.4.2.6 (Adversary Buffer Actions in Split Protocol)**.** Returning to our running example of the execution of the Split Protocol (from Section 5.3.4) and the trace *tr* from Example 5.3.4.2, we see that

$$AdvBufferActions(tr) = {}^{\sharp}\{(Adv, \mathrm{BR}(D,K)), (Adv, \mathrm{BR}(K,C))\}$$

If given the threat model $AtMostNABA(tr, 1)$, this would be false, as:

$$\big|AdvBufferActions(tr)\big| \not\le 1$$

If given the threat model $AtMostNABA(tr, 2)$, this would be true.

5.4.3 General threat models

Having defined some example threat models concerning buffers or components specifically, we now define some threat models which incorporate restrictions on *both* components and buffers.

Definition 5.4.3.1 (*TM*: At Most *n* Adversary Actions)**.** We can then restrict the total number of adversary actions (against both components *and* buffers) within a trace as follows:

$$\begin{aligned} \mathit{AtMostNAA}(tr,n) \iff & \mathit{AtMostNACA}(tr,x) \wedge \mathit{AtMostNABA}(tr,y) \\ & \wedge (x+y=n) \wedge (x,y,n \geq 0) \end{aligned}$$

We can also easily create threat models that require a passive adversary, i.e., one that can only read the contents of buffers' and components' memories, rather than modify them.

Definition 5.4.3.2 (Active Adversary Actions)**.** First we define the multiset of actions within a trace which are active, i.e., all adversary actions except BR and CMR:

$$\mathit{ActiveAdvActions}(tr) = {}^{\sharp}\left\{ tr_i \;\middle|\; 0 \leq i \leq \left|tr\right| \wedge tr_i \in \mathit{ActiveAdvEvent} \right\}$$

Then we say that a trace that is limited to only passive attackers must not have any active actions in the trace:

Definition 5.4.3.3 (*TM*: Only Passive Attackers)**.**

$$\mathit{OnlyPassiveAttackers}(tr) \iff \mathit{ActiveAdvActions}(tr) = {}^{\sharp}\varnothing$$

As all of these threat models are or can easily become predicates on a single trace, these can be combined to make larger threat models. These larger threat models can then be used make a range of different statements about what type of properties a trace must (or must not) exhibit to be considered valid under said threat model.

5.4.4 Valid traces

We recall from Section 5.3.3 that a trace is an ordered sequence of *TraceEvent*s, and that *Trace*(P, AA) is defined to be the set of all finite traces associated with the execution $\mathcal{X}_{P,AA}$ of a protocol P and a set of allowed adversary actions AA. *Trace*(P, AA) is intuitively the full range of possible traces that a system can execute, given a protocol P and allowed adversary actions AA.

What this definition (and notably AA) does not capture is more expressive restrictions on the adversary's ability, as we have just described with Threat Models. Here, rather than not presenting an action as a possible choice for the adversary (from a given state, based upon meeting certain premises in that current state), we sometimes need to verify after the fact that the overall trace meets certain conditions. Then we can decide whether to accept the trace's validity, and therefore whether a violation of a security property *within this trace* might constitute a failure of the protocol to satisfy its security claims (as described in Section 5.5). As we have described in this section, a TM is a predicate defined on a single trace ($TM : tr \in Trace(P, AA) \to \{True, False\}$) that describes a specific threat model.

Definition 5.4.4.1 (Valid Traces)**.** We therefore define the set $ValidTraces$ ($\subseteq Trace(P, AA)$) to be the traces of an execution of P, AA which are allowed (i.e., return True) by a specific threat model TM. We separate the protocol P and the pair (AA, TM) as they represent the honest and adversarial parts of the model respectively. That is,

$$ValidTraces(P, (TM, AA)) = \left\{ t \;\middle|\; t \in Trace(P, AA) \wedge TM(t) \right\}$$

5.5 Security properties and claim events

Different systems have different aims and required properties to operate correctly and securely. Protocols have both functionality and security requirements; we make inline security claims to integrate the desired security requirements or properties in to the specification of a protocol itself. We will then see in Section 5.5.3 whether or not a security claim holds when attacked by an adversary with certain capabilities, as defined by a threat model.

Claim events are used within protocol specifications to make statements about the exact properties we wish the protocol to satisfy. We make claim events from the perspective of one specific component or actor within a protocol, as claim events are not able to guarantee a property from all points of view. For further discussion and illustration of why this is the case, we refer the reader to Chapter 4 of [74] by Cremers and Mauw.

Cremers and Mauw's definition of "Security Properties as Claim Events" is a useful starting point, but it necessarily relies upon the honesty of certain agents involved in the protocol: for a term to remain confidential, its intended recipient must not be dishonest, or controlled by the adversary. This restriction of security claims being valid only when communicating with honest agents is a form of implicit threat model, and has direct parallels with our use of the set *Trusted*. We however go significantly further in terms of specification of the threat models and range of restrictions placed on operations allowed by the adversary.

The operational semantics rules for claim and running events have already been defined in Definitions 5.2.4.8 and 5.2.4.9.

5.5.1 Secrecy

Secrecy is the security property whereby at the end of a run or trace, the adversary does not know, nor can it infer from its knowledge (*AK*) the term(s) we wish to remain secret. Where we make a secrecy claim, it is written as $\mathsf{claim}_\ell(c, secret, x)$ where c is the component executing the claim event, *secret* is the property we are maintaining, and x is the term whose secrecy we wish to uphold, or prevent the adversary from learning. Informally, the following definition says says, for all Valid Traces that have a secrecy claim event on a term, the adversary cannot infer that term.

Definition 5.5.1.1 (Secrecy)**.** Let P be a protocol, *TM* be a threat model, and *AA* be a set of allowed adversary actions. We say that the secrecy claim event $\gamma = \mathsf{claim}_\ell(c, secret, x)$ is *correct* if and only if:

$$\forall tr, x'.\Big(tr \in \mathit{ValidTraces}(P, (\mathit{TM}, \mathit{AA})) \wedge \forall c, i. tr_i = \mathsf{claim}_\ell(c, secret, x')\Big) \\ \implies \mathit{AK}(tr) \nvdash x'$$

Note that we use x and x' to distinguish between terms in the protocol specification, and the terms in the execution. Claim events at the protocol specification stage may be different to those in the actual execution of the protocol: if the term x used by the claim in the protocol specification is a fresh value, it will become localised within the execution (e.g., x^1). If the term is a variable (e.g. Y) it will be localised *and* instantiated into a ground term during execution; e.g., the function *localise* substitutes $[Y^2/Y]$, and then the variable substitution $\sigma_3 = [r^1/Y^2]$ makes the variable term ground. We use ℓ as an identifier to distinguish between multiple otherwise syntactically identical claims in the same trace.

5.5.2 Authentication

Authentication encompasses the family of properties which traditionally considers agreement on identities, roles, and data. There are many different forms defined in the literature (see especially [131]), but here we focus on two simple but powerful properties we used in Chapter 4.

Here we only consider two components agreeing non-injectively or injectively on a single term, t; the value of that term can be a data term (such as a key or random number), the identity of the communications partner, or the identity of another component altogether. We reason that the modeller can easily build other, stronger authentication properties involving e.g., multiple identities and data terms by combining multiple agreement properties.

Definition 5.5.2.1 (Non-injective agreement)**.** Let P be a protocol, TM be a threat model, and AA be a set of allowed adversary actions. We say that the non-injective agreement claim event $\gamma = \mathsf{claim}_\ell(c, \mathit{NI\text{–}agree}, c', t)$ is *correct* if and only if:

$$\forall tr, c', t'.\Big(tr \in \mathit{ValidTraces}(P, (TM, AA)) \wedge \forall c, i.tr_i = \mathsf{claim}_\ell(c, \mathit{NI\text{–}agree}, c', t')\Big) \\ \implies \exists j.tr_j = \mathsf{running}_\ell(c', t')$$

Definition 5.5.2.2 (Injective agreement)**.** Let P be a protocol, TM be a threat model, and AA be a set of allowed adversary actions. We say that the injective agreement claim event $\gamma = \mathsf{claim}_\ell(c, \mathit{Inj\text{–}agree}, c', t)$ is *correct* if and only if:

$$\forall tr, c', t'.\Big(tr \in \mathit{ValidTraces}(P, (TM, AA)) \wedge \forall c, i.tr_i = \mathsf{claim}_\ell(c, \mathit{Inj\text{–}agree}, c', t')\Big) \\ \implies \Big(\exists j.tr_j = \mathsf{running}_\ell(c', t')\Big) \wedge \Big(\nexists k, c''.tr_k = \mathsf{running}_\ell(c'', t') \wedge k \neq j\Big)$$

Lowe describes many other authentication properties in [131], but we believe these two basic properties give the modeller sufficient tools and flexibility to describe agreement over component identities and data terms. More than one of these events can be included in a protocol per component, or alternatively, the term t over which the components agree can have multiple component identities and data terms included within it. For example, to adapt an action fact from Chapter 4 a protocol trace could contain the event $\mathsf{running}_\ell(a, << a, b, c, d >, k\text{–}seaf >)$, where a, b, c, and d are component identities, and $k\text{–}seaf$ is a data term.

We note that many other definitions of (non-) injective agreement require agreement on the identities of both parties *and* the data term in question, but we default to flexibility: we believe that with the presented tools a modeller can easily construct terms within running and claim events which meet this and other requirements if so desired.

5.5.3 Correctness of security properties

We say that a protocol (with respect to a threat model and set of allowed adversary actions) maintains its claimed security properties if and only if all of its stated claim events are correct.

Example 5.5.3.1 (Correctness of the Split Protocol's security claim events)**.** Returning to our example of the Split Protocol, we now consider whether whether Split Protocol (*SP*)'s stated claim event $\phi = \mathsf{claim}_\ell(K, secret, x)$ is correct with respect to the threat model $TM_1 = AtMostNABA(tr, 2)$, and allowed adversary actions, $AA = PassiveAdvEvent$. In this case, we require that:

$$\forall tr, x'. \Big(tr \in ValidTraces(SP, (AtMostNABA(tr, 2), PassiveAdvEvent)) \\ \wedge \forall i. tr_i = \mathsf{claim}_\ell(K, secret, x') \Big) \implies AK(tr) \nvdash x'$$

for the claim event $\phi = \mathsf{claim}_\ell(K, secret, x)$ to be correct.

We recall from Example 5.3.4.2 that we found a trace, tr, which did not appear to maintain the secrecy of the desired term.

$$tr = [(0, \mathrm{create}_{SP}(D)), (0, \mathrm{send}((D,K), r^0)), (Adv, \mathrm{BR}(D,K)), \\ (0, \mathrm{send}((D,C), \{\!| r |\!\}^a_{pk(B)})), (1, \mathrm{create}_{SP}(K)), (1, \mathrm{recv}((D,K), r^0)), \\ (1, \mathrm{send}((K,C), \{\!| x^1 |\!\}^s_{r^0})), (Adv, \mathrm{BR}(K,C)), \ldots]$$

We can now manually verify whether this satisfies our criteria.

- This trace is a 'Valid Trace' of the Split Protocol under our chosen threat model: $tr \in ValidTraces(SP, (AtMostNABA(tr, 2), PassiveAdvEvent))$. We know this because:
  - This trace is a possible trace from an execution of Split Protocol, $tr \in Trace(SP, PassiveAdvEvent)$.
  - The only adversary actions in the trace are Buffer Reveal actions, and BR $\in PassiveAdvEvent$.
  - This trace is true under the threat model $TM_1 = AtMostNABA(tr, 2)$.
- In the given trace, $AK(tr) \vdash x^1$. When localised in this trace, x and $\mathsf{claim}_\ell(K, secret, x)$ become x^1 and $\mathsf{claim}_\ell(K, secret, x^1)$ respectively, so we say that (with respect to the threat model TM_1 and allowed adversary actions AA), the stated claim event $\phi = \mathsf{claim}_\ell(K, secret, x)$ is not correct.

We therefore say that the secrecy claim event is not correct, and that SP does not maintain its claimed security property of secrecy, under the given threat model and allowed adversary actions.

We can however find threat models and allowed adversary actions under which the Split Protocol does maintain its security properties.

Example 5.5.3.2 (Proof of correctness of secrecy claim under specific threat model)**.** We now prove that for the Split Protocol under a specified threat model, the secrecy claim $\phi = \mathsf{claim}_\ell(K, \mathit{secret}, x)$ is always correct.

First, we specify a Threat Model TM_2, and allowed adversary actions. We say the range of allowed adversary actions are unrestricted, and therefore $AA = AdvEvent$. In the threat model, we specify the set $Trusted = \{K, B\}$, and then say that the adversary may not compromise the components in this set, as these components will necessarily have direct, unencrypted knowledge of the term whose secrecy we are trying to maintain: if the adversary compromises either of these components, we cannot defend against this. This is modelled by the predicate:

$$ComprComps(tr) \cap \{K, B\} = \varnothing$$

(Please refer to Definition 5.4.1.4.)

Secondly, we imagine that an adversary has arbitrary control over one single channel or component, but no access to or influence over any others. This might be caused by a probe or implant into a system, reading the communications over a channel, as discussed by Brian Krebs in [124]. Alternatively, this could be caused by a single component having been designed and/or manufactured (before the system was built) to have a backdoor by an adversary or other malicious actor. This part of the threat model is represented by the predicate:

$$MaxCompromised(tr, n) \iff \Big(\big|ComprBuffers(tr)\big| + \big|ComprComps(tr)\big|\Big) \leq n$$

where we assign $n = 1$. This says that at most one buffer or component may have (potentially multiple) adversary actions performed against it. (See Sections 5.4.1 and 5.4.2 for the definitions of the sets *ComprBuffers* and *ComprComps*.) We therefore say our threat model TM_2 for this specific attacker (and associated proof) is as follows:

$$TM_2 = (ComprComps(tr) \cap \{K, B\} = \varnothing) \wedge MaxCompromised(tr, 1)$$

The adversary therefore can choose to perform compromise actions against any *one* of the components or channels depicted in Figure 5.8. Note that in our system model, the adversary may still only compromise at most one of these, but that component or channel then may be compromised as many times as the adversary likes within the execution of the protocol. We no longer refer to *AA* unless explicitly necessary as no restriction on the adversary's capabilities is created here.

We prove the correctness of the claim $\phi = \mathsf{claim}_\ell(K, \mathit{secret}, x)$ under threat model TM_2 (and allowed adversary actions *AdvEvent*) by contradiction. First, we recall the definition of

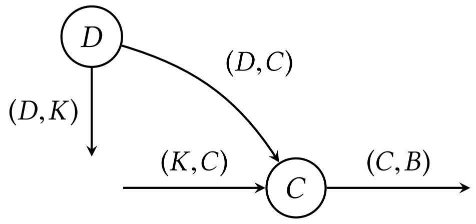


Figure 5.8: Components and channels the adversary may compromise under TM_2

the secrecy property, in the context of our threat model and allowed adversary actions. We say that the secrecy claim $\phi = \mathsf{claim}_\ell(K, secret, x)$ is correct if and only if:

$$\forall tr, x'.\Big(tr \in ValidTraces(SP, TM_2, AdvEvent)) \land \ \forall i.tr_i = \mathsf{claim}_\ell(K, secret, x')\Big) \\ \implies AK(tr) \nvdash x'$$

Lemma 5.5.3.1 (Adversary Knowledge of Private Keys)**.** In the Split Protocol, without compromising a particular component, the adversary cannot learn that specific component's private key.

Proof: By inspection of the protocol, no private keys (of type $sk(\cdot)$) are ever transmitted, and the adversary does not initially know any components' private keys. By inspection of the protocol, no private keys are ever transmitted by uncompromised components. Hence, the lemma holds. ■

Corollary 5.5.3.1 (Adversary Knowledge of $sk(B)$)**.** The adversary cannot learn $sk(B)$ under the Threat Model TM_2.

Proof: Since TM_2 disallows compromise of component B, Lemma 5.5.3.1 yields the confidentiality of B's private key. ■

We now assume there exists a Valid Trace tr in which $\exists i, k.tr_i = \mathsf{claim}_\ell(K, secret, x^k)$ and in which $AK \vdash x^k$. We show by contradiction that such a trace cannot exist. First, the adversary cannot construct x^k by encryption or pairing: under TM_2 the adversary cannot compromise component K to influence or choose the value of x^k, so x^k must be Fresh as dictated in the protocol. Because x^k is Fresh, the adversary cannot have x^k in its initial knowledge. As such, if the adversary is to learn x^k, it must do so by learning the term directly, or by deriving it by deconstruction, unpairing, or a combination of the two. If the adversary is able to construct terms and then deconstruct them to derive x^k, the adversary must have already known x^k.

There are therefore two sources of terms which the adversary might wish to learn within a system: components and buffers. Messages are placed into buffers by 'send' actions, and removed by 'recv' actions. We argue that from the protocol description, we only need to consider terms within 'send' actions, not 'recv', as new information (to an adversary) is only introduced on to a channel by a component's 'send' message, not 'recv' messages.

We recall that we say a term is 'instantiated' if it is both localised and ground, i.e., has no free variables, and is localised to a particular thread. We consider all terms sent in the protocol, and consider whether each uninstantiated term could unify to x^k, or whether the adversary could derive x^k from this term. We first consider all of the uninstantiated terms transmitted within messages, according to the protocol specification. These are:

$$r, \quad \{\!|\, r \,|\!\}^{a}_{pk(B)}, \quad \{\!|\, x \,|\!\}^{s}_{U}, \quad \{\!|\, W \,|\!\}^{a}_{pk(B)}, \quad \{\!|\, \{\!|\, X \,|\!\}^{s}_{W} \,|\!\}^{a}_{pk(B)}$$

so any instantiated terms within the execution of the protocol must take this form.

- The term r^ℓ (the instantiation of r for some $\ell \in$ TID) can never unify to x^k ($k \neq \ell$) as both are Fresh, ground terms generated by different threads. We must have that $k \neq \ell$ as under TM_2 the adversary is not allowed to compromise component K, and component K generates the Fresh term x^k. There are no subterms of r^ℓ if it is Fresh, and if it is not Fresh (due to being constructed by the adversary) we have already argued that the adversary cannot learn x^k by construction rather than deconstruction.

- The term $\{\!|\, r \,|\!\}^{a}_{pk(B)}$ can never unify to x^k itself because x^k is Fresh, (and therefore cannot have any subterms). The adversary can never decrypt $\{\!|\, r \,|\!\}^{a}_{pk(B)}$ by Corollary 5.5.3.1, so the instantiation of this term cannot be a possible route to the derivation of x^k.

- The same argument holds for the terms $\{\!|\, W \,|\!\}^{a}_{pk(B)}$ and $\{\!|\, \{\!|\, X \,|\!\}^{s}_{W} \,|\!\}^{a}_{pk(B)}$.

The only other terms in any components' memory not transmitted as a result of a 'send' action are $pk(B)$ and $sk(B)$. The adversary cannot learn $sk(B)$ by Corollary 5.5.3.1, and a term of the form $pk(t)$ (ditto $sk(t)$) can never be equal to a Fresh term.

We therefore claim that the only route the adversary can have to deriving x^k must be from the uninstantiated term $\{\!|\, x \,|\!\}^{s}_{U}$. When all terms are instantiated and the derivation is normalised, the final step of any possible derivation of x^k must therefore be of the form:

$$\frac{\{\!|\, x^k \,|\!\}^{s}_{t} \quad t}{x^k}$$

for some (instantiated) term t. For this trace to exist, we must have that both $AK \vdash \{\!|\, x^k \,|\!\}^{s}_{t}$ and $AK \vdash t$. We say that an accessible subterm is one which might potentially be derived by the inference rules. E.g., within $\{\!|\, t_1 \,|\!\}^{s}_{t_2}$, t_1 is accessible because $M \vdash \{\!|\, t_1 \,|\!\}^{s}_{t_2} \wedge M \vdash t_2 \Rightarrow M \vdash t_1$, but t_2 is *not*, because $M \vdash \{\!|\, t_1 \,|\!\}^{s}_{t_2} \wedge M \vdash t_1 \not\Rightarrow M \vdash t_2$.

- The term $\{\!|\, x^k \,|\!\}^s_t$: By inspection of the protocol, the only components or buffers which can ever contain a term $\{\!|\, x^k \,|\!\}^s_t$ as an accessible subterm are the components K, C or B, or the buffers (K,C) or (C,B). The adversary cannot compromise K or B under TM_2, and we have already argued that the adversary cannot learn $\{\!|\, x^k \,|\!\}^s_t$ from the terms $\{\!|\, \{\!|\, X \,|\!\}^s_W \,|\!\}^a_{pk(B)}$ or $\{\!|\, W \,|\!\}^a_{pk(B)}$ in the buffer (C,B). The adversary therefore must compromise one of the components or buffers C or (K,C) to learn or influence the term $\{\!|\, x^k \,|\!\}^s_t$.

- The term t: Again, by inspection of the protocol, the only components or buffers which can ever contain a term t as an accessible subterm are D, K, (D,K), (C,B). The adversary cannot compromise K under TM_2, and we have already argued that the adversary cannot learn t from the terms $\{\!|\, W \,|\!\}^a_{pk(B)}$ or $\{\!|\, \{\!|\, X \,|\!\}^s_W \,|\!\}^a_{pk(B)}$ in the buffer (C,B). The adversary therefore must compromise one of the components or buffers D or (D,K) to learn or influence the term t.

Under TM_2 the adversary may only perform compromise actions on one channel or buffer ($\notin$ *Trusted*). We have demonstrated that to learn or influence the term $\{\!|\, x^k \,|\!\}^s_t$, the adversary must compromise one of the set $\{C, (K,C)\}$, and to learn or influence the term t, the adversary must compromise one of the set $\{D, (D,K)\}$. It is plain to see that:

$$\{C, (K,C)\} \cap \{D, (D,K)\} = \varnothing$$

We conclude that the adversary cannot infer both $\{\!|\, x^k \,|\!\}^s_t$ and t in the same trace (as required to derive the term x^k), and hence we come to a contradiction. ■

5.6 Going forward

In this chapter, we have defined and presented a new framework for modelling component-based systems under partial compromise. We believe this easily extensible operational semantics gives modellers a rigorous and precise level of control over all of the components' interactions, the range of allowed adversary behaviours and specific threat model(s), and the security properties considered for each protocol model.

We have demonstrated the framework's functionality and utility with some protocol examples and their analyses, showing how minor changes to adversarial behaviour and abilities can have major consequences for the security of a protocol. We recognise that before this framework will become truly useful for many major real-world systems, the semantics will have to be extended to provide better support for both statefulness and unbounded looping over protocol rules.

While our analyses have been performed manually, we envision that these operational semantics could be incorporated into automatic tooling, perhaps by the creation of a compiler similar to Casper [132] which then automatically translates the precise component models, threat models, and security properties written in our semantics into Tamarin's specification language. We would then be able to take advantage of Tamarin's excellent backwards search algorithm and associated heuristics, in combination with our presented semantics' ease of modelling for and focus on partial compromises.

We believe that Tamarin's semantics and specification language are (with a lot of manual work) able to model everything we have described in this chapter. However, as we discovered especially in Chapter 4, doing so is often extremely cumbersome for the modeller, requiring in-depth knowledge and experience of both Tamarin's specification language and the use of macros over the models to re-create the desired range of partial compromise and fine-grained threat models.

The purpose of this chapter has been to build a theoretical framework which allows complex systems of components under partial compromise to be modelled with ease, and then similarly to be able to verify (or falsify) the system's claimed security properties. While we have achieved many of the theoretical elements of this, the real value of this will only become apparent when analysis becomes automatic: combining this theoretical framework with solid tool support and automated analysis methods will hopefully allow us to showcase the full benefits of this approach.

While it is almost impossible (and indeed, unworkable) to capture every meaningful detail of a real-world system in a single model, we believe that our chosen level of abstraction will allow us to perform useful analyses of the logical interactions between components. It is our hope that the framework presented in this chapter will be the means by which many new and exciting multi-component systems will be formally modelled and analysed, and that considering systems under attack from whole new ranges of previously unexplored adversarial capabilities will be of significant benefit to our understanding of these systems.

Don't just read it; fight it! *Ask your own question, look for your own examples, discover your own proofs. Is the hypothesis necessary? Is the converse true? What happens in the classical special case? What about the degenerate cases? Where does the proof use the hypothesis?*

— Paul Halmos

6 Related Work

We now discuss the literature related to partial compromise of network security protocols and systems (component-based or not), attacks where compromise of individual components within systems led to catastrophic violation of security properties, and historically proposed solutions to partial compromise from the literature. We then consider methods and tools for modelling and analysing protocols, before considering previous work relating to each of our specific analyses in Chapters 3 and 4.

6.1 Attacks

Before looking at existing approaches which aim to *solve* the issue of partial compromise, it is important to have an understanding of attacks and partial compromises of systems that have previously been successful. Flaws, exploits, back-doors, and attacks on protocols, components, and whole systems are discovered and published on an almost weekly basis, and we discuss a couple of the more relevant attacks in the following section [16,45,104,130,167]. Particularly of interest are attacks where total compromise was achieved by attackers, but where better overall system design (not just implementation) could have prevented the vulnerability.

Random number generators and software libraries, are two examples of components within larger systems where the individual compromise of this component has a major impact upon the security of the whole system. We briefly discuss these components' specific compromise, the broader impact that this has upon a larger system's security, and how this impact might be limited.

6.1.1 TLS

Attacks against SSL/TLS (whether against the specification, or individual implementations) have been shown on an alarmingly regular basis [142, 167]. Very often these attacks only compromise the confidentiality of messages passed between client and server, and don't have any further implications on the host's security. In some cases, however, the vulnerability can be used to gain far greater leverage over the host, completely destroying the authenticity of any connection the host makes, not just over a specific TLS session, or those to one specific server [182].

As an example, a Key Compromise Impersonation (KCI) attack has been demonstrated for TLS (up to version 1.2) that would allow an attacker who can place a client certificate on the user's computer to impersonate *any* website or server to that user [104, 183]. The client believes that their system is secure, and that TLS is working correctly when they visit any website, when in fact they have lost their expected and desired security properties. While this attack only works for static (non-ephemeral) Diffie-Hellman connections, it shows how the single point of compromise — where an adversary installs a malicious client certificate — destroys the security of all future network connections.

It is expected that an attacker who can place a client certificate to which they know the private key on a host can impersonate that host to any other website. Likewise, if a malicious server can place a public key for a website (to which they know the private key) on a host, they can impersonate that website or server to the user: these are both acceptable levels of partial compromise that are difficult to avoid, although in specific this context, Certificate Transparency goes a long way to prevent this [129]. However, compromising all future network connections through one malicious client certificate is unacceptable. We claim there is a reasonable implicit expectation that TLS should be more resilient than this, ideally requiring that individual server private keys are compromised before the confidentiality of sessions for that specific server can be violated, rather than for *any* connection.

As a further extension to this type of attack, Basin et al. show (and provide solutions to) a new category of attack, Actor Key Compromise (AKC), against a range of protocols [36]. AKC allows an adversary who has compromised e.g., an ISP's long-term secret keys not only to impersonate the ISP to any of its users as is expected, but to impersonate arbitrary users to the ISP.

It is worth noting that TLS 1.3 [154] is now believed to be resilient to both Key Compromise Impersonation and Actor Key Compromise attacks, as discussed in Section 4.13.2.3; neither DNP3: SAv5 or 5G-AKA alone achieve AKC resilience.

6.1.2 Random Number Generators

The use of Cryptographically Secure Pseudo-Random Number Generators (CS-PRNGs) is very at the core of many cryptographic primitives and security protocols. The security of many cryptographic protocols depends entirely upon the randomness provided to them by a system's (CS-)PRNG, where randomly generated values are often used either directly or indirectly as key material.

It has become apparent that some PRNGs (and specifically the standards defining them) have become the target of attack by various governments, most notably the "Dual EC pseudorandom number generator" [45, 60]. If successful, these attacks would have had severely detrimental effects on the security of any system using a weakened PRNG: "Break the random-number generator, and most of the time you break the entire security system" – Bruce Schneier [161, 163].

The perceived success of attacks on PRNGs illustrates that single points of failure within systems become obvious targets: if a system is designed in such a way that the compromise of any one individual component (such as the PRNG) can tear down the security of the remaining system or protocol, then this design choice should clearly be avoided. Instead, protocols have been designed that don't rely solely upon a PRNG for their security. Many protocols use a combination of output from the PRNG and a user's long-term secret key (feeding both in to a key derivation function, KDF) to create a session key, but this long term secret key may well also have been derived from the same PRNG [54, 67, 94, 123]. Some modern protocols derive new session keys by combining randomness with other sources of private information, e.g., maintained state or message history from previous sessions [61, 72, 94, 151]. This then prevents an adversary who has compromised the PRNG but not e.g., the main memory from compromising the confidentiality and/or authenticity of a user's sessions. The inclusion of state within the key derivation process allows protocols to exhibit even stronger security properties yet, such Post-Compromise Security [62].

An attacker who has access to, or who can predict the output of a system's PRNG has at least as much (if not strictly more) information and leverage over that system than one who cannot. A system where key material is derived directly and solely from the output of the PRNG cannot hope to defeat this attacker. If key material is derived from a secure key derivation function combining a number of sources, including the PRNG and state from previous sessions, the adversary's search-space is potentially lessened, but not by a large enough amount to allow a polynomial-time adversary to mount a successful attack. This gives a strictly lower level of security compared to an adversary who cannot predict the PRNG's output, but still high enough for most practical purposes.

We note in Chapter 4 that within the 5G-AKA protocol, the home network's ARPF is the sole component generating any randomness: at no point do any of the other components or participants including the UE generate any randomness which feeds into the derivation of the resultant session key K_{SEAF}. In contrast, while the DNP3: SAv5 protocol suite (described and analysed in Chapter 3) has a number of other failings with respect to partial compromise, it does require that all parties generate randomness in the form of challenge data. These individual terms are only used during each sub-protocol, and they do not feed into the derivation of e.g., the session keys.

6.1.3 Side-channel attacks

Side-channel attacks are those which consider the reality of implementation [189]. They take advantage of physical properties of the executing computers, or implementation choices, rather than vulnerabilities in the abstract design and description of the encryption algorithm or used protocol. These channels take many forms, but include **timing channels** (e.g., does the successful decryption of a key/cipher-text pair take longer than if it fails? Does computing on one value take the computation down a longer branch than another value? [120]), **power analysis channels** (e.g., does processing a '1' use a slightly different amount of power to processing a '0'? [136]), **electromagnetic channels** (e.g., does the device leak information about the secret data it is processing electromagnetically? [126]), and **fault-attack based channels** (e.g., does the device or software library leak useful information if it is caused not to function correctly? [53]). This is only a small selection of the types of known side-channel, and they are often very hard to defend against. As these attacks are heavily implementation specific, we consider this type of attack out of scope for our research. That said, one benefit of the fine-grained adversarial modelling presented in Chapter 5 is that specific actions (e.g., 'reveal the terms component x knows to the adversary') can be described and allowed very easily. The origin of this adversary ability is not of concern to the semantics: this cause could be from e.g., side-channel analysis.

6.2 Existing solutions, techniques, and approaches

Various methods, solutions, and techniques with the aim of reducing the impact of partial compromise have been suggested in the realms of hardware, software, and theory. We explore some of the more relevant ideas to our presented work in the following sections, starting with existing solutions, before considering relevant theoretical building blocks and techniques.

6.2.1 Existing solutions

Hardware solutions Silicon is significantly less mutable than software, and hence there have been a number of hardware-based approaches to levels and layers of system security that an attacker ought not be able to compromise. While there are a lot of potential solutions related to partial compromise, none address it fully.

Some solutions suggest giving the user an extra piece of hardware as a trusted root; this way trust can be built up from the starting point of this physical object, and its (hopefully) trusted code base. TPMs have classically been used for this, although ARM's TrustZone and Intel's SGX are alternative, more powerful Trusted Execution Environments [22, 58, 133]. The implicit assumption with an extra piece of trusted hardware is that this root of trust will never be compromised. Then, if other parts of a system are compromised, the trusted root should allow a user either to revert to a trusted state, or at least to maintain security for critical information such as long-term secret keys.

Pöpper et al. describe how the use of a separate hardware '**porter device**' (such as a mobile phone) can provably ensure forward secrecy under full compromise [153]. The threat model specified here is that all network communications are under a Dolev-Yao adversary initially, and that the adversary can completely compromise agents and their long-term secret keys *after* the protocol has finished. The porter device manages the key storage, and the only major functionality which has to be guaranteed is that the porter device will perform secure (session) key deletion after a pre-determined period of time: this means that the protocol and device are still vulnerable to complete device compromise during the network session, but not afterwards.

The separate stages of attacker and levels of compromise during and after the protocol (i.e., a Dolev-Yao attacker during the protocol, and a 'full-compromise' attacker after the protocol has finished) show that having an adaptable threat model or attacker is both realistic and modellable with existing tools. Secondly, the lower level of trust placed in the porter device is important: normally with an HSM, we expect to have to trust the HSM fully for the life of the device, and never allow it to be compromised. With a porter device and relevant protocols, it is sufficient to trust that the secure deletion operation really took place within the desired timeframe.

TPMs and TrustZone Trusted Platform Modules and ARM's 'TrustZone' technology are relatively mature hardware-based approaches to creating multiple forms of trust within a platform. They can be used for providing trust on the platform itself, and within network security protocols such as remote attestation.

TPMs provide the ability to attest to the software that is running on the system, and hence inform the user or a remote server of the state of the platform in a way that allows them to make a decision about the behaviour and trustworthiness of the platform [133]. Secondly, and of more direct use to the situation of partial compromise, TPMs can be used as forms of Hardware Security Modules, entrusted with encryption keys for network connections or disk-drives. This way, if the CPU, memory, or HDD of a system are compromised, it is completely plausible that the attacker will not be able to retrieve the encryption keys, and hence any confidential data. TPMs can act as encryption/decryption oracles, but due to speed concerns, this functionality is normally reserved for long-term secret- or master-keys, which are then in turn used to encrypt or decrypt other, non-master encryption keys stored less securely in main system memory.

One of the aims of TPMs is to ensure that "there should be no way that the trusted platform can be compromised simply through participating in network protocols": while this aim seems to have been at least partially achieved in terms of preventing new malicious software from running on a system, it in no way precludes the possibility of compromise through the presence of existing vulnerabilities in successfully attested, running software, such as the operating system.

The physical TPM chips are separate to the CPU. They are not Turing Complete as they don't have unrestricted IO, and they use protected memory. Rather than weakening the computer's capabilities, these limitations are instead beneficial: they allow guarantees to be made about the contents of the TPM, which in turn can give accurate reports about the state of the host PC. The TPM's Platform Configuration Registers are restricted in as much as they do not allow 'write' operations directly, only 'extend' (hash-chain update), or reset/wipe. This restricted capability, and various intermediate-level formal specifications of the TPM's design and implementation hopefully make it exceptionally difficult, if not impossible for an attacker to compromise the TPM.

A suggested 'partial compromise' application for the remote attestation capabilities of Trusted Network Connect (TNC) is that of home banking. The user opens a special (bank-created) virtual machine, which the bank remotely attests is correct (via the PC's TPM), and then the virtual machine itself attests that the bank is real: this methodology prevents large categories of attack, even when the host computer itself is compromised.

ARM's **TrustZone** is integrated into the design of most modern ARM processors, and provides a Trusted Execution Environment [22]. This allows for secure execution away from the normal, untrusted world of a computer's main operating system. Most importantly, it enforces (at hardware level) the separation of even system-level (kernel) processes in the insecure world from secure-world processes and memory, making security and privilege level orthogonal to each other. This can be used to keep encryption keys and confidential data secure, as well as performing sensitive processing away from the main OS, or running applications which need higher guarantees of security than a commodity operating system can provide.

This functionality (similar to an HSM) allows an otherwise comparatively insecure operating system to be considered a separate component from the secure TrustZone environment. This environment can be use as a more secure stoarge location and execution environment for both long-term and session keys, for use by network security protocols.

For guarantees of the resulting security, both of these solutions assume that there are no hardware-level backdoors or vulnerabilities which could be exploited; a lot of work has been done to mitigate against this and provide guarantees of the correctness and security of the designed and realised silicon [164].

While TrustZone is a useful and widely-distributed step towards a trusted execution environment (available within many modern mobile phones), there have been a large number of attacks against implementations of TrustZone and software residing within the TrustZone execution environment [17, 44, 168].

Software solutions While various hardware-based solutions are becoming more mature and more widely distributed, once designed and realised in silicon, they are completely fixed until the user purchases a new PC or device. Combining the benefits of purpose-designed, deliberately immutable hardware with flexible software that can rely upon the hardware as a root of trust will perhaps defeat the largest subset of attackers, while retaining a high level of flexibility. Even so, various pure-software solutions have been proposed; we consider the most advanced and most relevant one here.

seL4 is a microkernel whose functional correctness has been formally verified [119, 165]. The utility of a formally-verified microkernel such as seL4 is that it allows multiple components to be placed on the same physical hardware, and yet still to have some guarantees of security if one of these constituent components is remotely compromised. This could allow e.g., the home network's AUSF and ARPF from Chapter 4 to be run on the same hardware (as is allowed by the standard), without concern that compromising the credentials of one would immediately give the attacker privileged access to the other.

seL4's proofs of correctness have been machine checked in Isabelle/HOL, and separately provide guarantees both of correctness of operation (that the system is free of programming errors), integrity enforcement and authority confinement. Having proofs of correctness and confinement for the kernel provides guarantees that no malicious software should be able to compromise the kernel of the operating system: while this is an excellent result, and indeed defeats many entire classes of attack, this sort of formal analysis and proof of correctness is extremely expensive in terms of time and required expertise; it is not yet considered possible to attempt to prove similar properties of a full kernel, such as the Linux kernel.

An alternative situation where this approach is useful is maintaining separation between e.g., mutually distrusting components. These components could be as simple as a small amount of native code, or as complex as an entire operating system (e.g., Linux). seL4 enforces separation between these components, but inter-process communication can be enabled and pre-defined for very specific tight channels, allowing the system to take advantage of the knowledge that no other inter-process communication can occur *except* through these formal channels.

Multiple seL4 components can then be combined in novel ways to create entire systems of mutually distrusting elements, with very precisely defined communications channels and protocols. Formal analysis of the protocols for communication over these channels would need to be performed to re-gain any assurance of the components' individual and the system's overall security properties, but this is a much more approachable verification task than that of a whole operating system. Andronick et al. take advantage of the guarantees provided by seL4 to propose a framework to build secure, complex systems "in the presence of large untrusted components", using Isabelle/HOL to prove the correctness of information flow properties between components of the system [27].

6.2.2 Building blocks and techniques

As well as implemented whole-system solutions, there are many different theoretical techniques that have been developed to improve the security of components and systems. The following techniques have been useful in the setting of partial compromises.

Multi-factor authentication (MFA) This is the idea that no one single form of identification (whether something you have, know, or are) should be sufficient to authenticate a user to a system [55]. This can be implemented by requiring both a password — something you know — and a time-sensitive code sent out of band (e.g., over SMS) to a pre-registered mobile phone — something you have [20]. Multiple methods or factors of authentication give a basic layering whereby if one security factor is compromised, no access is gained by the adversary.

Multi-factor authentication is one of the earliest forms of authentication to recognise that a system should still operate securely under partial compromise: if a password is stolen or transmitted in plaintext, then this is not sufficient for an attacker to impersonate the user. For an attack to succeed, the attacker must also gain control of the other factors required, such as a mobile phone or hardware token, before being able to log in as the user. This is still a form of binary security: the attacker gains either complete access or no access at all.

Sometimes multiple different forms or factors are not necessary, but instead multiple channels suffice. The classic example of this is nuclear missile launches requiring authorisation from two separate keys (e.g., the Captain and Weapons Officer): here, two authorisation commands of similar type are communicated over two different channels. With only one out of two, nothing happens [184]. Another well-known (previously described) form of multi-factor authentication is the use of credit or bank cards, which require both the card and its associated PIN; having one on its own is not sufficient. We design and formally analyse a basic multi-channel protocol in Section 5.3.4, showing its utility against threat models allowing partial compromise.

Secret sharing Splitting keys, secrets, or other confidential information into different parts across separate components is an obvious way of limiting the damage of any one component's compromise. Ideally, the secret splitting would require a majority (if not all) of the components involved to collaborate before any of them learn anything about the secret. This isn't always feasible (e.g., where individual components must perform computations on the plaintext at regular intervals), but in situations where it is possible, a fair amount is known about schemes to provide these guarantees.

As early as 1979, Adi Shamir described how to share a secret in a (k, n) threshold scheme [166]. This has n pieces or recipients (where $n = 2k - 1$), and a guarantee that unless k or more of the pieces come together, no-one can recover any information about the secret (even with $n/2 = k - 1$ pieces). Pedersen then improves upon this scheme in 1992 [148] by giving each party a method for verifying that they have received the correct information, in a non-interactive way.

One of the main issues we observe with both of these approachs is that once the requisite threshold is reached, the secret is entirely revealed to an individually compromisable party.

Wu et al. presented a very promising construction in [185] from 1999, known as "Intrusion Tolerance via Threshold Cryptography"; in this scheme, the private key is split up across multiple share servers, and importantly is never reconstructed at any point for use in decryption or signing. This provides a very fault-tolerant construction, requiring an adversary to compromise at least t-out-of-k servers before learning any information about the private key.

While they were successful in demonstrating the implementation of such a scheme with acceptable performance figures, there remain concerns about the algorithms and associated performance for the initial secure generation of the private key shares. Once implemented, this overall scheme successfully makes the share servers into a decryption or signing oracle which is hard to compromise; this is a worthwhile and interesting construction.

The problem still remains that any individual client with authority to request decryption or signing is still a single point of failure. The paper presents a counter based method for detection of misuse (see also [140]), but this necessarily only alerts the client to compromise after the fact, rather than preventing compromise in the first place, which was the aim. Sadly very little work seems to have been done relating to this project after 1999; integrating a construction like this into modelling and analysis tools could be worthwhile future work.

Homomorphic encryption Gentry [96] proposes Fully Homomorphic Encryption, a method by which computation can be performed on ciphertexts, without revealing the key or plaintext to the party performing the computation. This technique could be used to solve many of our described problems in theory (perhaps allowing middle-boxes to contribute to a protocol without learning the involved secrets), but the practice and associated overheads are still a long way from reality and practicality.

This category of techniques could well be used to allow components to collaborate successfully, while preventing any one of them from breaching the confidentiality of important data. Unfortunately, due to the atomic nature of terms in symbolic modelling, neither homomorphic encryption nor secret sharing are a good fit for modelling and analysis within our verification tools and methods.

Reverse firewalls Mironov et al. suggest a novel way to protect communications between two parties over an untrusted network connection where one of the components or actors is compromised: an untrusted 'reverse firewall' sits at the boundary of a local network, and re-randomizes all encryption passing through it [141]. Dodis et al. then extend this to the message transmission context in [85]; this prevents a compromised component or actor within the network from leaking information to eavesdroppers, or from using poor randomness. The reverse firewall only needs to know the public keys of those involved, and does *not* need to know any private keys.

The first drawback to this scheme is that the type of asymmetric encryption used must be re-randomizable: they propose using ElGamal rather than RSA. Secondly, some protocols need to be modified, and thirdly, it does not work directly with hybrid encryption, i.e., using asymmetric encryption to transmit a symmetric key to another party, then using that symmetric key for the session. This can be resolved however by using a key *agreement* protocol, rather than one party blindly accepting a symmetric key proposed by the other party.

While this still needs full implementation and realisation in the form of the reverse firewall itself, the main result is a compiler that takes an existing two-party protocol as input, and outputs a functionally equivalent protocol that admits a reverse firewall, 'preserving both functionality and security', preventing a compromised component from leaking information to an adversary.

This seems to be a natural solution for systems and protocols under partial compromise, but we have not yet studied or considered re-randomisable encryption within the symbolic modelling and analysis tools available to us at the moment. We believe it would be possible, and suggest that modelling and analysing this type of novel construction would be interesting future work.

Fault tolerance Fault tolerant systems are those that can continue to operate when some of their components are not operating correctly, or at all [99]. This is very often desirable for safety-critical systems: if an aeroplane's main computer crashes mid-flight, it is critically important for the backup to take over seamlessly. In some situations, a fault-tolerant solution allows for continued operation, but only at a lower level of functionality compared to normal operation. Fault tolerance research and design often considers the distinction between failing safely and failing badly: even if certain redundant components are no longer operating, does the system still function overall [59]? This can cause the opposite of the desired effect in security and cryptography, which instead needs systems to 'fail securely' [95, 157]: the most basic example of this is to deny access by default, or alternatively in the case of failure, to undo any changes and return to a secure state. In high-assurance cryptographic systems [135], if duplicated implementations are present for runtime verification and comparison purposes (perhaps to give some guarantee that encryption has been performed correctly), then 'failing open' upon compromise of a component or even non-malicious software failure could lead to a violation of desired security properties. Fault tolerance research often considers redundancy, removal of single points of failure [87], fault isolation [81], and fault containment [158]: all of these properties are potentially desirable for security and resilience under partial compromise, but must be designed to ensure that the system fails securely, rather than open.

Detection of attacks and compromises If a system cannot prevent compromise of some or all of its components, and is not otherwise tolerant of partial compromise, are there at least protocols or mechanisms which can automatically detect when a component or key has been compromised? Detection of failure or compromise has been a topic of interest for dependability and fault tolerance research for many years [32, 128], but this has seen limited interest in network security protocols until fairly recently. Milner's thesis [140] ("Detecting the Misuse of Secrets: Foundations, Protocols, and Verification") addresses this topic comprehensively, "developing foundations and constructions for security protocols that can automatically detect, without false positives, if a secret such as a key or password has been misused." This research additionally considers the required conditions for sound detection, and how this can allow for automatic revocation of compromised secrets, i.e., without the need for human intervention. Milner demonstrates that protocols which can detect misuse must be stateful; furthermore, some of the resultant modifications to Tamarin improving the ability to analyse stateful protocols automatically have benefited this thesis, especially Chapters 3 and 4.

Certificate Transparency [129] has been proposed as a method of making websites' TLS certificates publicly auditable; the idea is that this provides users with the ability to detect compromised or malicious Certificate Authorities: "Certificate Transparency makes it possible to detect SSL certificates that have been mistakenly issued by a certificate authority or maliciously acquired from an otherwise unimpeachable certificate authority. It also makes it possible to identify certificate authorities that have gone rogue and are maliciously issuing certificates." This technique is valuable for the safety and security of the wider internet, but requires a non-trivial amount of audit infrastructure in place, separate to any individual protocol implementation such as TLS.

ARPKI ("Attack Resilient Public-Key Infrastructure") [39] is an alternative proposal that "ensures that certificate-related operations, such as certificate issuance, update, revocation, and validation, are transparent and accountable." It combines fault tolerance and deterrence towards misbehaviour: it can tolerate compromise of $n - 1$ trusted entities, all operations are publicly auditable, and it has the benefit of having been designed with a formal model whose main security properties have been formally verified in Tamarin.

6.3 Verification, analysis, and assurance

Being able to describe, analyse, and understand systems are essential steps before being able to make any security claims. There are lots of different tools available for modelling and verification, and we consider the main options here.

Modelling of systems Describing a system and its desired security properties is an essential first step towards making guarantees about those security properties. Traditionally, claims about systems have been made in a binary fashion, simply separating the secure and insecure worlds, where no compromise is tolerated in the former, and complete compromise is implicit in the latter. Abadi creates typing rules for secrecy in the form of 'spi-calculus', with the guarantee that if a system type-checks, it does not leak secrets [12, 13].

Extending and improving upon this, Gordon and Jeffrey create a pi-calculus that not only introduces and formalises the idea of conditional secrecy, (i.e., that a message is secret unless certain principals are compromised), but that allows for the description of different levels of security, rather than just 'secure' or 'not-secure' [100]. These arbitrary levels of security then form an ordering, where no information from a higher level is ever allowed to flow to a lower level. While this is a novel and useful contribution, the security is focussed on information flows, not cryptographic definitions of security; they actively state that they 'do not address indirect flows'. They make a small attempt to introduce an extension to symbolic cryptography, but this is only a very minor and exploratory contribution, and only covers the most basic of non-nested cryptographic operations. This idea of conditional secrecy and separation between principals (cf. components) which are or are not allowed to be compromised is the starting point for threat models with Trusted Components (*TrComps*) in Section 5.4.1.

The choice of modelling language not only determines the method of proof or model checker required, but also determines what can and cannot be expressed, described, and verified in the system model. Different languages and model-checkers have different goals, strengths, and weaknesses.

We discuss modelling of protocols and systems using Tamarin's specification language in significant detail in Chapter 2; we give a range of alternative model checkers and their associated specification languages later in this section.

Threats and threat modelling After describing the honest system under study, we define the adversary or type of adversarial behaviour against which it must remain secure.

The standard starting point for attacker models within formal methods is the Dolev-Yao attacker [86]. In line with our stated 'symbolic modelling assumptions' from Section 2.3, messsages are atomic terms or symbols rather than a bit-stream, and most importantly, *the attacker controls the network*: they can choose to do anything to any of the messages they are passed by any of the end points, agents, or components that make up the network. Cryptography is assumed to be perfect, so for example the message $\{\!|\ x\ |\!\}^s_k$ (plain-text x symmetrically encrypted with key k) can *only* be decrypted by someone in possession of the term k; no adversary can just be extremely lucky and 'guess' x from the encrypted form after a certain amount of time.

As each actor simply passes their messages directly to the attacker, the attacker can then choose which agents to send any messages to (regardless of original intent), and the order they're received. The attacker can arbitrarily inject, modify, replay, or delete messages from the network. The implication here is that while a real-world adversary might be unlikely to have this all-powerful capability, if a protocol can withstand this strong attacker model, it must also be able to withstand any attack by a strictly less powerful adversary.

This strict attacker model makes sense when only considering network protocols: once a message has left the agent or end point, it has no further control over what happens to it; the attacker controls the network. The agent has no visibility of which route through a network, or indeed the wider internet the message may take, if it has been diverted benignly or maliciously, and it cannot prevent the message from being modified or replaced by a different one.

This same reasoning does not always match up quite so well with partial compromise: D-Y is really only capable of modelling compromise in a binary fashion. The 'strongest attacker' assumption still holds here, but it is more beneficial to our understanding of the strengths and weaknesses of a system's design to be able to learn which operations and guarantees can still be maintained under partial compromise, and which ones cannot. Graduated or non-binary compromise of sections of systems such as this does not align well with the D-Y model, and hence more malleable models are more appropriate.

Cremers et al. pit different attackers with steadily increasing capabilities against a series of protocols, allowing them to deduce a partial ordering of the relative strengths of these protocols in [34]. In many situations it is enough to know whether a protocol or system is secure against the strongest attacker; if it fails to reach this standard, it is useful to know against which strength of adversary it is still secure. In addition to the clever use of a hardware 'porter device', Pöpper et al.'s use of different attackers at different stages (discussed in Section 6.2) is a good example of attackers with different strengths and capabilities within the same model [153].

Giving the attacker extra capabilities under certain circumstances (e.g., in the eCK model, capabilities such as Session Key Reveal, Ephemeral Key Reveal, and Long Term Key Reveal [127]) allows us to learn more about the relative strengths of protocols. Importantly, these properties start to help us push back at the idea of binary security: an attacker can gain partial success by performing a 'Session Key Reveal' action, and will compromise the confidentiality of that session and all data in it, but (protocol dependent) they should not learn anything about the honest agent's other sessions, or their long-term secret key.

6.3.1 Formal analysis and automated verification tools

Having created a formal model of a system, we use formal verification and analysis tools to determine whether these models meet their security goals. We emphasise that verification results are only able to make statements about the models we create. Very often this is still useful, as this process can either find direct attacks which can be verified or falsified against a real system, or if 'verified', we can exclude large classes of logical attacks against a protocol.

Strand Spaces are a method for specifying network security protocols, and proving security properties about these protocols [176]. A 'strand' is one role's viewpoint or perspective on a protocol (whether honest or adversarial), and a 'bundle' is a combination of strands, which forms a representation of the overall protocol interactions. A 'strand space' is then a set of strands from both the honest roles and the adversary. Protocols are represented as strand space specifications, and protocol properties (such as secrecy or authentication) are represented as propositions about bundles over the strand space. Formal verification of these properties then consists of manually proving the propositions correct.

Kamil and Lowe use Strand Spaces to consider a series of protocols and abstractions of secure channels. This includes a formal analysis of TLS 1.0 in the strand spaces model [116], as well as a more theoretical approach to understanding the composition of application layer protocols with secure transport layer protocols [114, 115]. These secure channel analyses consider whether (when modelling overall systems) one can abstract away from the details of the lower level secure transport layer, or whether one *only* has to consider the properties of the application layer, relying directly upon the guarantees provided by the secure channel.

Continuing with this theme, Dilloway and Lowe use trace specifications (rather than strand spaces) to describe various security properties of transport protocols, giving a hierarchy of specifications for the resultant secure channels [80]. As well as exploring the protection provided by secure channels, this hierarchy is useful for the analysis of layered security architectures.

CSP is a commonly used language for describing patterns of interaction in concurrent systems, and is model checked with **FDR**; Casper is a related tool for protocol verification which outputs CSP$_M$ (the machine-readable dialect of CSP) and is then model checked with FDR [98, 105, 132]. While CSP is a very expressive and useful language, FDR suffers from issues with state-space explosion. As a result, high levels of expertise are needed to tailor models in such a way that this state-space explosion is minimised.

Casper is a "compiler for the analysis of [network] security protocols" built upon CSP, but does not allow for modification of the adversary's (or intruder's) capabilities or properties, beyond its identity and initial knowledge; the range of a protocol's possible security properties are pre-defined [132]. It takes significantly more work to describe a protocol or system in CSP than in Casper, but the added flexibility and capacity to describe a very large and distinct range of capabilities, interactions, and properties to both honest components and attackers makes CSP a better option for describing systems and channels subjected to partial compromise.

AVISPA is an industrially focussed symbolic tool that combines four back-end model checkers into one easy to use 'push-button' interface [180]. The security protocol to be verified and its desired security properties are written in a high level language (HLPSL), which is then converted to an intermediate format which the four different model-checking back-ends can then start working on. These model checkers are: SAT-MC, OFMC, CL-AtSe, and TA4SP. All support infinite search-spaces, but with bounded session verification, apart from TA4SP which performs unbounded verification. The range of different strengths and weaknesses of each of the back-end model checkers gives better likelihood of the overall system finding attacks on various protocols and systems, which was demonstrated by it finding new attacks on ISO-PK, IKEv2, SET, ASW and H.530 as early as 2006.

AVISPA normally assumes perfect cryptographic primitives and a Dolev-Yao attacker, but it is possible to specify some other forms of non-D-Y attacker. AVISPA can also deal with a small amount of state, although its precise limitations are not completely clear.

The modular separation of the input interface and language (HLPSL) from the final low-level format fed to each model checker is a useful design, as it means that development and changes in one do not affect the others; this is useful for allowing the easy development of input descriptions that do not follow standard system configurations, security goals, and D-Y-style attackers. The range of different model-checkers employed in the back end are likely to be beneficial in our situation compared to a single model-checker whose optimisations are specifically tuned and improved for one particular type of scenario or problem, such as internet-based network protocols.

AVANTSSAR is the successor to AVISPA, and similarly supports bounded protocol verification, additionally supporting the modelling of policies and business processes [31]. AVANTSSAR uses all of the same back-ends as AVISPA, but without TA4SP.

ProVerif is a symbolic, unbounded, automatic protocol verification tool in the Dolev-Yao model dating from as early as 2001; the verifier checks the satisifability of a horn-clause representation of the modelled protocol [50, 51]. ProVerif can accept protocol descriptions either as Prolog-style rules, or in an applied pi-calculus format, and can handle most modern types of cryptographic primitive including public-key cryptography, hashing, and Diffie-Hellman key exchange. It can give false attacks, but the authors claim it will never falsely verify security properties of a protocol. ProVerif can prove properties of the following types: secrecy, authentication (and other correspondence properties), strong secrecy ("the adversary does not see the difference when the value of the secret changes"), and "equivalences between processes that differ only by terms".

Scyther is a tool for the automatic, unbounded symbolic verification of security protocols [65]. It assumes a Dolev-Yao-style adversary, and does not support user-specified threat models. Scyther is ideal for analysis of network-based security protocols, but it was not designed with partial compromise, or situations where the adversary does not control the network in mind.

We have described **Tamarin** [138] in some detail in Chapter 2, and through its use demonstrated that it provides the necessary tools to the modeller and analyst for understanding and verifying systems under partial compromise. However, as we describe in Chapter 5, it is not designed specifically for this purpose: successfully creating and analysing protocol models and threat models of systems under partial compromise in TAMARIN requires a lot of in depth TAMARIN-specific knowledge; we believe that we could potentially achieve the results from Chapters 3 and 4 more easily if the modelling and analysis techniques we used had partial compromise 'built in' from the outset.

There are many other verification tools for different purposes, ranging from mathematically focussed interactive theorem provers such as Coq and Isabelle/HOL, through to code-checking bounded model checkers such as CBMC, which proves the safety of assertions in C code under a given bound [78, 125, 144]. To the best of our knowledge, none of these are as appropriate as the previously described tools for our purposes.

6.3.2 Computational models

In Chapter 1 we briefly touched upon the fact that we use symbolic models in this thesis, and not computational. This different modelling and analysis methodology is exceptionally valuable, but not directly suited to the size and complexity of protocols which we have considered.

Computational models have evolved from the game-based proof methodologies used by cryptographers to show that a particular primitive is secure. Computational models of key exchange protocols focus on a security experiment: we start with an underlying mathematical assumption, and then attempt to prove an indistinguishability property. Within Bellare-Rogaway's seminal framework [42] they attempt to bound the probability of distinguishing a real session key from random. This bound is usually expressed in terms of the probability of attacking the underlying cryptographic primitive.

As an example, one of these chosen assumptions is the Decisional Diffie-Hellman (DDH) assumption [52]. This states that for a multiplicative group G of order q, a generator g and values x, y, z chosen uniformly at random from $\mathbb{Z}_q$, if given g^x, g^y, g^{xy} and g^z, the adversary cannot distinguish between g^{xy} and g^z with any probability better than random. As discrete logarithms are thought to be hard to compute in most groups, the belief is that a probabilistic polynomial-time adversary will not be able to achieve this.[1]

To demonstrate the impossibility of distinguishing between the genuine key and a random bit-string, computational proofs often use a technique called 'game-hopping' [171]. This is a methodology for showing the equivalence of (or at least bounding the adversary's advantage between) different security games, while preserving or bounding the probability of some particular event.

Computational proofs are exceptionally useful for the analysis of many modern protocols, and are believed to provide very strong proofs of security. They are mostly performed manually (EasyCrypt and CryptoVerif notwithstanding), and are therefore are unsuited to

[1]Discrete logarithms are thought to be hard to *compute* in most groups, and likewise to distinguish as described above. The exception in the distinguishing (but not computing) case is groups with a non-trivial bilinear pairing, such as the Weil or Tate pairings.

analysing large, complicated systems with lots of different cases and large amounts of state. As the protocols and systems we have considered in this thesis fall very firmly into this category, we decided that using symbolic methods for modelling and analysis was more appropriate and more likely to produce useful results for our purposes.

6.4 DNP3: SAv5

We now consider the literature related to the industrial control system system, DNP3, and its Secure Authentication v5 protocol suite, as modelled and analysed in Chapter 3. There is relatively little academic literature considering DNP3; previous work either explores the broader security of DNP3, or, in contrast, restricts itself to analysis of SAv5's *Critical ASDU Authentication Protocol* in isolation.

East et al. 2009 provide an interesting and thorough taxonomy of the different types of attack against DNP3 in [91], but as this paper was published before SAv5 was standardised, it does not consider Secure Authentication.

Bratus et al. 2016 use LangSec to create an input-checking parser in [56] to ensure more secure implementations of SCADA/ICS protocols, applying this methodology to DNP3 in the form of a filtering proxy which validates DNP3 messages. This methodology focusses on implementation hardening for DNP3, which means their code demonstrates resilience to state-of-the-art black-box as well as white-box fuzz-testing tools. They do not consider the security of network protocol elements of DNP3.

Crain and Bratus 2015 consider methods for securing ICS/SCADA network security protocol *implementations* in [64], focussing on DNP3, which includes discussion of the Secure Authentication v5 protocol.

They look at various issues with the challenge-response vs. aggressive modes of the *Critical ASDU Authentication Protocol*, as well as the overall protocol's high levels of statefulness, and conclude: "DNP3: SAv5 contains a number of anti-patterns that will likely serve as a significant source of bugs. Vendors and standards bodies adding security to SCADA/ICS protocols should strongly favor a layered approach to security in which legacy protocol issues can be decoupled from SCADA object models and semantics."

Tawde et al. 2015 propose a 'bump-in-the-wire' solution for the key-management and encryption of critical packets within IEC/TS 62351-5 (the protocol suite upon which DNP3: SAv5 is based), but provide no formal analysis of this addition or the existing protocols [174].

Attacks Claimed: Amoah et al. 2014 and 2016. As discussed in detail in Section 3.7, this research uses Colored Petri-Nets to model and analyse both the non-aggressive and aggressive modes of this sub-protocol, discovering a denial of service attack in the non-aggressive mode [26], and a "replay attack" when the aggressive and non-aggressive modes are combined [24]. Both papers only consider the *Critical ASDU Authentication Protocol* in isolation. In Chapter 3 we conclude that this claimed attack is an artefact of a model that is too coarse, and is not possible in faithful implementations of the standard.

Separately, Amoah et al. then make the novel contribution of a method for *Critical ASDU Authentication* within the Broadcast or Unicast setting, in [25]. Amoah's 2016 thesis [23] supplements these papers by providing greater detail of the modelling and analysis of the *Critical ASDU Authentication Protocol*.

6.5 5G and mobile telephony protocols

The security and cryptography of mobile telephony protocols have been studied for many years, as there are clear privacy, financial, and security related incentives for many parties. These become even more important in light of the widespread use of mobile phones for voice, text, and internet based communication.

There have been many generations of mobile telephony protocols, both for basic connections and later for security. The first commercially automated cellular network started as early as 1979 in Tokyo by Nippon Telegraph and Telephone (NTT). The main global standards still in use today started with the Global System for Mobile communications (GSM), and are broadly named '2G', with the first commercially deployed 2G network starting in Finland in 1991 [107]. 3G (Universal Mobile Telecommunications System or UMTS) was released in 1999, and included significant security improvements over 2G. 4G (Long Term Evolution or LTE) was standardised in 2009 Many books, papers, and web-pages have been written on the history of telecommunication, GSM, and its successors [102, 107]. We consider 5G's predecessors here as the security protocols within 3G, 4G, and 5G are based closely on each of their previous versions, meaning that it is worth considering the standards and literature surrounding previous generations' protocols, such as [1,3,4,6,7,9–11,33,173,177,179,187,188]. One constant requirement for new versions of the standards has always been maintaining backwards compatibility.

1G networks were analogue, and did not employ any cryptography or security measures for authentication or encryption; as such, it was possible to eavesdrop, intercept, and/or modify messages and traffic. 2G networks employed basic security measures and cryptographic key distribution, requiring authentication of the UE/SIM to the network; notably, this did not require any authentication of the network (home or serving) to the UE, so fake base-stations were a real threat. Variants of the 2G protocols used either "GSM Authentication" for GSM [9], or "Cellular Authentication and Voice Encryption", CAVE-based authentication for the competing CDMA standard (Code-division multiple access) [1]. Similar to the variants of AKA protocols defined for UMTS and LTE, these have three main network entities involved in the authentication protocol: the Home Location Register (HLR), the Visitor Location Register (VLR, located in the serving network similar to the SEAF), and the Mobile Subscriber (MS, or in our descriptions, the UE).

In terms of security improvements, 3G networks introduced mutual authentication with the AKA protocol [6], and additionally required encryption between core-network entities rather than intra- and inter-network communication being in plaintext, as was the case in 2G. From a security perspective, the overall architecture stayed roughly the same, modulo renamings and minor changes.

Historically, versions of AKA have only been analysed after deployment, and typically used extremely simplified models. Versions of the original 3G AKA protocol were manually analysed in 1999 [2], using TLA and BAN Logic. Both these methods consider abstract models that are significantly coarser than modern techniques allow for, and consider only very weak threat models.

4G then introduced the EPS-AKA and EAP-AKA protocols in 2009 [11], and Release 8 more broadly updated the specification to require an all-IP internal network for signalling and data, rather than the historic (and relatively vulnerable) SS7 telephony signalling protocols, originally designed in 1975 [156]. Before this change in encryption and internal signalling protocols, it was possible for any one of thousands of legitimate mobile network operators to intercept calls or messages using this protocol. In spite of these improvements, end-to-end security for mobile network communications is still a long way off. 4G additionally introduces better cryptographic key separation and key renewal, with the aim of limiting exposure if session keys are compromised. Various successful implementations of downgrade attacks (taking advantage of the required backwards compatibility) are presented in [149]. The conclusion of these attacks was that users should disable protocols which don't require mutual authentication, only using 3G or 4G protocols.

As we describe in Chapter 4, **Tsay and Mjølsnes** present a similar attack to our discovered potential vulnerability in [177] from 2012, but for the older UMTS-AKA and LTE-AKA protocols, across a different boundary, and in the three-party context rather than four-party. This attack allows for a violation of authentication properties based upon session confusion, except in their attack, the confusion occurs between the serving network $\leftrightarrow$ home network boundary, (what we describe as the channel between the SEAF and AUSF) rather than our presented vulnerability where the race condition occurs entirely within the home network, i.e., between the AUSF and ARPF. The current 5G standards surprisingly do not even mention or consider mitigation (implicitly or explicitly) of this type of vulnerability caused by the messages between the AUSF and ARPF. The attack in [177] was found indirectly through use of CryptoVerif [49], and they make similar recommendations, gently discouraging protocol designers from relying on implementation-specific choices, and that similar designs should be discouraged: this advice from 2012 was apparently not heeded.

Køien proposes improvements to 4G's mutual authentication properties in [121], achieving full mutual, online authentication between parties, rather than the delegated authentication achieved by EPS-AKA. The paper observes that all previous AKA protocols delegate authentication authority from the home network to the serving network by forwarding on up to 5 unrevocable authentication vectors. This allows the home network to be completely offline during the challenge-response phase of the protocol: they reason that this requires too high a level of trust between network operators, and hence propose a protocol "Enhanced EPS-AKA" which is fully online for all parties. 5G-AKA now only forwards one authentication vector at a time.

Rupprecht et al. [155] provide a systematic framework of security research for existing attacks and defences in 2G, 3G, and 4G. As a consequence, they propose many open research questions for the security of 5G standards and implementations.

Arapinis et al. [28] analyse 3G's authentication protocols, discovering attacks against the privacy and linkability of subscriber identities. This modelling and analysis uses ProVerif, formally verifying the proposed solutions achieving unlinkability and anonymity.

O'Hanlon et al. [146] consider the interaction between 4G's authentication protocols and operator-backed WiFi services; they detail how the interaction between these can enable serious privacy violations, as well as their experiences reporting the discovered issues to the relevant stakeholders.

Hussain et al. [106] combine symbolic model checking with cryptographic protocol verification for 4G's attach, detach, and paging procedures, discovering 10 new attacks, including an authentication relay attack, allowing an adversary to spoof the location of a legitimate user.

5G seeks to improve the security properties present in 4G by strengthening the security and authentication properties between all three of UE, serving network, and home network. Various papers were presented to 3GPP for consideration and inclusion within the finalised standards, and this large range of documents and surveys was collected together in TR 33.899 [5]. **Arkko et al.** give an excellent proposal which hoped to improve the security of 5G further by introducing forward secrecy [30]; sadly this and many other similar proposals detailed in [5] were not accepted into the standard.

The most notable change between LTE and 5G is that of the subscriber's privacy: in LTE, the permanent USIM identifier or IMSI can be broadcast un-encrypted at the beginning of the EPS-AKA or EAP-AKA protocols, if the serving network does not have a TMSI for the subscriber; regardless, many privacy issues have been highlighted with the use of TMSIs [28]. In 5G, the SUPI ('Subscription Permanent Identifier', equivalent of IMSI) is ephemerally encrypted into a SUCI ('Subscription Concealed Identifier'). This solution was adapted from proposals by **Jimenez et al.** and **Norrman et al.** in [112] and [145] respectively, and provides significantly greater user privacy than before. As we state in Chapter 4, we did not model or analyse the privacy of the SUPI.

In very recent (concurrent) work [40], **Basin et al.** use a similar approach to ours to analyse 5G-AKA, but focus on different aspects. Basin et al. model and analyse a 3-party interpretation of the 5G-AKA protocol and its security properties, merging two major components (the AUSF and ARPF) to form a single 'Home Network' entity, similar to previous AKA versions. They discovered the authentication issues created by lack of integrity protection on the serving network's ID. In contrast, in Chapter 4, we consider the all four parties as defined in the protocol's specification. Basin et al.'s models provide detailed analysis of the counter re-synchronisation method and the privacy guarantees of 5G-AKA. They additionally model and analyse the 'Elliptic Curve Integrated Encryption Scheme' which 5G-AKA uses for SUPI concealment to ensure subscriber privacy. Beyond simple confirmation of these results, we do not consider privacy-specific properties within 5G. The attack we detail in Chapter 4 is not visible in their 3-party model.

An abundance of academic papers have proposed improvements to the AKA protocols over the years. Many, such as [14, 14, 18, 143] seek to do so by introducing various forms of asymmetric cryptography; none of these proposals have been accepted or introduced into 5G. We recognise that 3GPP is restricted by limitations and stakeholder requirements from the broader 5G ecosystem, and that these are a lot more restrictive than academic literature often appreciates. Bearing this in mind, we particularly encourage continued liaison and engagement between academia and industry to achieve these goals of stronger security together.

"Complexity is the worst enemy of security, and our systems are getting more complex all the time."

— Bruce Schneier

7 Conclusions

We set out hoping to combine the chaos and complexity of real-world, multi-component systems with the rigour of formal verification. Whether we would be able to achieve meaningful results within the timescales available was never a foregone conclusion: the problem of secrecy even in a *bounded*, simplistic, two-party protocol is famously undecidable [90]. Termination of our analysis algorithms, let alone achieving the specific outcomes we hoped for was therefore never guaranteed, especially when our objects of study were protocols deliberately outside the realm of well-defined academic constructs. Indeed, there are still results in the 5G component compromise section for which we never achieved termination.

Nevertheless, we persisted: we battled against exceptionally verbose and yet painfully under-specified standards and protocols clearly designed by committee, imprecise and often contradictory informal-language descriptions of behaviours, and regardless, modelled them as completely and realistically as we could. In spite of these barriers to accurate modelling and meaningful analysis, we believe we have firmly and positively addressed our first research question, showing in both Chapters 3 and 4 that it is possible to achieve significant verification results for major, real-world systems in realistic time frames.

Both our first and second research questions implicitly considered thresholds of simplicity. To achieve termination and therefore results, our models must be as simple as possible; for them to have value, they must be no simpler. Finding this sweet spot requires both expertise, and trial and error.

Through fine-grained, component-based modelling and analysis we showed a claimed attack in DNP3: SAv5 was false, and found potential vulnerabilities in 5G-AKA. We have successfully worked with 3GPP to ensure that the upcoming 5G standards are not vulnerable to our discovered race condition; we strongly reiterate that the security of any protocol or system must not depend on implicit engineering choices.[1] We conclude that both these positive and negative results clearly demonstrate the real-world value of *modelling* protocols in as much detail as possible, but within the limits of the tools' analytical capabilities. Walking that fine line requires knowledge of the capabilities and limits of the tools at our disposal, and the aims and motivation behind the protocol design and desired security properties.

Having pitted our two considered systems against the adversaries they were designed to fend off (with varying levels of success), a new challenger entered the ring: neither threat model as specified was especially strong in modern terms, so we felt easily justified in selectively giving the adversary more power and seeing how the protocols coped. As we discussed in the partial compromise results sections, neither protocol fared especially well.

With modern network intrusion techniques and a constant stream of vulnerabilities in major operating systems, perimeter defence is obsolete. Designing protocols whose entire security collapses when an adversary very slightly strays from its relatively weak threat model is not acceptable for modern protocols. DNP3: SAv5 and 5G-AKA were standardised in 2012 and 2018 respectively, and yet neither really takes advantage of modern cryptographic primitives or techniques for key exchange in any serious way.[2] They instead mostly settle for variations on a theme of symmetric key transport with hashing, which is a depressingly common trope in CNI/ICS protocols. We exhort the designers of protocols for systems such as these to push firmly back against legacy restrictions, and instead demand strong, modern cryptography.

Our modelling and analysis in both Chapters 3 and 4 demonstrates how far we have to go in terms of resilience under partial compromise. While we recognise that legacy and industry-specific constraints make significant changes in between versions of standards very hard, we state firmly that this encountered lack of resilience is no longer acceptable in elements of critical national infrastructure. These systems increasingly come under regular and sophisticated attack, so avoiding un-necessary single points of failure is essential.

Finally, we considered the formalisation of partial compromise. Rather than just modelling and analysing systems against normal D-Y-style threat models and increasing the adversary's capabilities in a relatively ad hoc manner, we sought to provide a theoretical framework for this analysis. Our operational semantics for partial compromise presented in Chapter 5 give the modeller an approachable yet powerful way to model a wide range of multi-component systems and protocols against a malleable array of fine-grained threat models.

We believe that creating a systematic approach which bakes flexible threat models and partial compromise in from the beginning will make partial compromise modelling and analysis easier to achieve, and additionally increase confidence in the depth, rigour, and accuracy of obtained results.

[1] We draw the interested reader's attention to the 'refinement paradox', see e.g., [97, 113].

[2] Such as bleeding-edge constructions like Diffie-Hellman from 1976 [79].

Future work

Having modelled and analysed two major systems which were broadly not resilient to partial compromise, there seem to be three natural routes for future work.

Firstly, it would be beneficial to model, analyse, and study in depth a major, real-world system which claims to be resilient to partial compromise. This would allow us to consider how this type of threat model is viewed by designers, and then evaluate more fully the utility and shortcomings of our operational semantics. We would ideally have performed this type of analysis within this thesis, but are not aware of any in-use CNI/ICS (or similar) protocols which make claims of this sort. Non-CNI/ICS systems which we believe may achieve this sort of partial-compromise resilience include Tor [82] and peer-to-peer solutions such as IPFS [43], but these have very often been designed with different primary purposes in mind (e.g., anonymity), and may only provide partial compromise resilience as a side effect: further study is required.

Secondly, to achieve this automatically within our presented operational semantics, the next steps would be to integrate these capabilities into a compiler from a partial compromise specification language into e.g., Tamarin's specification language; the gap here is mainly in implementation. As discovered through our exploration and research into DNP3: SAv5, large protocols are often complex both in terms of unbounded looping and their statefulness. Our presented framework is not yet mature enough to deal with these challenges; we therefore hope to extend the semantics to incorporate this capability more fully.

Finally, the development of new methods and techniques within protocols which actively create partial compromise resilience: we make suggestions throughout this thesis for specific improvements based upon our understanding of partial compromise in relation to best cryptographic practice, but these techniques have not originally been designed with partial compromise directly in mind. In Chapter 6 we considered an assortment of methods and techniques aiming to solve some of these problems, but evidently, few have been taken up. Starting with the intuition that a key-exchange protocol is just a form of secure multi-party computation we would hope to consider, study, and even design realistic and usable cryptographic and protocol-based methods which require multiple compromises of a system before meaningful violation of security properties is achieved.

We finish with two high level conclusions. Firstly, the time is now right for formal methods. Our research has shown that precise, fine-grained modelling of unwieldy and complex multi-component protocols is both possible and valuable in the real world. We strongly encourage industry to take full advantage of the state of the art, rather than it remaining hidden away in academia.

Secondly, we urge designers of real-world, multi-component systems and protocols to consider security under partial compromise from the start. When a sophisticated adversary cannot break a protocol on the wire, the next logical target is the end-points; with little hope of securing all end-points completely, it is essential that partial compromise resilience is also built into the protocols.

Appendices

A
Complete results for 5G-AKA protocol under partial compromise: $\mathcal{A}_{Stronger}$

In the following appendix we give further detail of the results of the full range of previously described lemmas, when under different channel and component compromise threat models. For the original results of the 5G-AKA protocol against the standard (no channel- or component-compromise) threat model, please see Section 4.9.

A.1 Channel compromise results

A.1.1 Naïve channels, with both channels D-Y compromised

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.1: Secrecy properties of 5G-AKA: naïve channels, both channels D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.2: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, both channels D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
UE	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.3: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, both channels D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
UE	✗	✗	N/A	✗	✗	✗
SEAF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.4: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, both channels D-Y compromised

A.1.2 Naïve channels, with SEAF ↔ AUSF channel only D-Y compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✓
	K	✓

Table A.5: Secrecy properties of 5G-AKA: naïve channels, SEAF ↔ AUSF channel only D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	T	✗	✗
AUSF	✗	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.6: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, SEAF ↔ AUSF channel only D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.7: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, SEAF ↔ AUSF channel only D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✓	✓	✗	✗

Table A.8: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, SEAF ↔ AUSF channel only D-Y compromised

A.1.3 Naïve channels, with AUSF $\leftrightarrow$ ARPF channel only D-Y compromised

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.9: **Secrecy** properties of 5G-AKA: naïve channels, AUSF $\leftrightarrow$ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.10: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, AUSF $\leftrightarrow$ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.11: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, AUSF $\leftrightarrow$ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	$\mathbf{K_{SEAF}}$	**Inj:** $\mathbf{K_{SEAF}}$
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.12: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, AUSF $\leftrightarrow$ ARPF channel only D-Y compromised

A.1.4 Naïve channels, with both channels readable-only compromised

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.13: Secrecy properties of 5G-AKA: naïve channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✗	✗	N/A	T	✗	✗
AUSF	✗	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.14: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.15: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	T	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.16: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, both channels readable-only compromised

A.1.5 Naïve channels, with SEAF $\leftrightarrow$ AUSF channel readable-only compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✓
	K	✓

Table A.17: Secrecy properties of 5G-AKA: naïve channels, SEAF $\leftrightarrow$ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	T	✗	✗
AUSF	✗	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.18: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, SEAF $\leftrightarrow$ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.19: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, SEAF $\leftrightarrow$ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	T	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.20: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, SEAF $\leftrightarrow$ AUSF channel readable-only compromised

A.1.6 Naïve channels, with AUSF ↔ ARPF channel readable-only compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.21: Secrecy properties of 5G-AKA: naïve channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.22: Authentication properties of 5G-AKA from the **UE's** point of view: naïve channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.23: Authentication properties of 5G-AKA from the **serving network's** point of view: naïve channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	T	T
ARPF	✗	✗	✓	✓	✗	✗

Table A.24: Authentication properties of 5G-AKA from the **home network's** point of view: naïve channels, AUSF ↔ ARPF channel readable-only compromised

A.1.7 TLS-like channels, with both channels D-Y compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.25: Secrecy properties of 5G-AKA: TLS-like channels, both channels D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.26: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, both channels D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.27: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, both channels D-Y compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.28: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, both channels D-Y compromised

A.1.8 TLS-like channels, with the SEAF $\leftrightarrow$ AUSF channel only D-Y compromised

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✓
	K	✓

Table A.29: Secrecy properties of 5G-AKA: TLS-like channels, SEAF $\leftrightarrow$ AUSF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	T	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.30: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, SEAF $\leftrightarrow$ AUSF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.31: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, SEAF $\leftrightarrow$ AUSF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	T	T	N/A	T	T	T
SEAF	✗	✗	✗	✗	✗	✗
ARPF	✓	✓	✓	✓	✓	✓

Table A.32: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, SEAF $\leftrightarrow$ AUSF channel only D-Y compromised

A.1.9 TLS-like channels, with the AUSF ↔ ARPF channel only D-Y compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.33: Secrecy properties of 5G-AKA: TLS-like channels, AUSF ↔ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.34: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, AUSF ↔ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.35: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, AUSF ↔ ARPF channel only D-Y compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.36: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, AUSF ↔ ARPF channel only D-Y compromised

A.1.10 TLS-like channels, with both channels readable-only compromised

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.37: Secrecy properties of 5G-AKA: TLS-like channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.38: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.39: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, both channels readable-only compromised

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.40: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, both channels readable-only compromised

A.1.11 TLS-like channels, with SEAF ↔ AUSF channel readable-only compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✓
	K	✓

Table A.41: Secrecy properties of 5G-AKA: TLS-like channels, SEAF ↔ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	T	✗	N/A	T	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.42: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, SEAF ↔ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.43: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, SEAF ↔ AUSF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	T	T	N/A	T	T	T
SEAF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.44: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, SEAF ↔ AUSF channel readable-only compromised

A.1.12 TLS-like channels, with AUSF ↔ ARPF channel readable-only compromised

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.45: Secrecy properties of 5G-AKA: TLS-like channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.46: Authentication properties of 5G-AKA from the **UE's** point of view: TLS-like channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.47: Authentication properties of 5G-AKA from the **serving network's** point of view: TLS-like channels, AUSF ↔ ARPF channel readable-only compromised

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.48: Authentication properties of 5G-AKA from the **home network's** point of view: TLS-like channels, AUSF ↔ ARPF channel readable-only compromised

A.2 Component compromise results

A.2.1 All SEAFs compromisable by the adversary: All-X

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✓
	K	✓

Table A.49: Secrecy properties of 5G-AKA: All SEAFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	T	✗	N/A	T	✗	✗
ARPF	T	✗	N/A	T	✗	✗

Table A.50: Authentication properties of 5G-AKA from the **UE's** point of view: All SEAFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
AUSF	✓	✓	✓	✓	✓	✓
ARPF	✓	✓	✓	✓	✓	✓

Table A.51: Authentication properties of 5G-AKA from the **serving network's** point of view: All SEAFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
SEAF	✗	✗	✗	✗	✗	✗
ARPF	✓	✓	✓	✓	✓	T

Table A.52: Authentication properties of 5G-AKA from the **home network's** point of view: All SEAFs compromisable by the adversary: All-X

A.2.2 All AUSFs compromisable by the adversary: All-X

Party	**Term**	**Result**
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✓

Table A.53: Secrecy properties of 5G-AKA: All AUSFs compromisable by the adversary: All-X

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.54: Authentication properties of 5G-AKA from the **UE's** point of view: All AUSFs compromisable by the adversary: All-X

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	✗	✗	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.55: Authentication properties of 5G-AKA from the **serving network's** point of view: All AUSFs compromisable by the adversary: All-X

Role/Term	**UE**	**SEAF**	**AUSF**	**ARPF**	**K_{SEAF}**	**Inj: K_{SEAF}**
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	T	T
ARPF	✓	✓	✓	✓	✗	✗

Table A.56: Authentication properties of 5G-AKA from the **home network's** point of view: All AUSFs compromisable by the adversary: All-X

A.2.3 All ARPFs compromisable by the adversary: All-X

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✗
AUSF	K_{SEAF}	✗
ARPF	K_{AUSF}	✗
	K	✗

Table A.57: Secrecy properties of 5G-AKA: All ARPFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	T	✗	N/A	T	✗	✗
AUSF	T	✗	N/A	T	✗	✗
ARPF	T	✗	N/A	T	✗	✗

Table A.58: Authentication properties of 5G-AKA from the **UE's** point of view: All ARPFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
AUSF	✓	✓	✓	✓	✓	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.59: Authentication properties of 5G-AKA from the **serving network's** point of view: All ARPFs compromisable by the adversary: All-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✗	✗	N/A	✗	✗	✗
SEAF	✓	✓	✓	✓	✗	✗
ARPF	✗	✗	✗	✗	✗	✗

Table A.60: Authentication properties of 5G-AKA from the **home network's** point of view: All ARPFs compromisable by the adversary: All-X

A.2.4 All SEAFs compromisable by the adversary apart from 'mine': Not-My-X

Party	Term	Result
UE	K_{SEAF}	✓
SEAF	K_{SEAF}	✓
AUSF	K_{SEAF}	✓
ARPF	K_{AUSF}	✓
	K	✓

Table A.61: Secrecy properties of 5G-AKA: All SEAFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	T	✗	N/A	T	✗	✗
ARPF	T	✗	N/A	T	✗	✗

Table A.62: Authentication properties of 5G-AKA from the **UE's** point of view: All SEAFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
AUSF	✓	✓	✓	✓	✓	✓
ARPF	✓	✓	✓	✓	✓	✓

Table A.63: Authentication properties of 5G-AKA from the **serving network's** point of view: All SEAFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
SEAF	✓	✓	✓	✓	T	T
ARPF	✓	✓	✓	✓	✓	T

Table A.64: Authentication properties of 5G-AKA from the **home network's** point of view: All SEAFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✓	✓	N/A	✓	✓	T
AUSF	✓	✓	N/A	✓	✓	T
ARPF	✓	✓	N/A	✓	✓	T

Table A.65: Authentication properties of 5G-AKA from the **UE's** point of view: All SEAFs compromisable by the adversary apart from 'mine': Not-My-X, **with SNID fix**

A.2.5 All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X

Party	Term	Result
UE	K_{SEAF}	✗
SEAF	K_{SEAF}	✓
AUSF	K_{SEAF}	✓
ARPF	K_{AUSF}	✗
	K	✓

Table A.66: Secrecy properties of 5G-AKA: All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	T	✗	N/A	T	✗	✗

Table A.67: Authentication properties of 5G-AKA from the **UE’s** point of view: All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.68: Authentication properties of 5G-AKA from the **serving network’s** point of view: All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
SEAF	✓	✓	✓	✓	T	T
ARPF	✓	✓	✓	✓	✓	T

Table A.69: Authentication properties of 5G-AKA from the **home network’s** point of view: All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✗	✗	N/A	✗	✗	✗
AUSF	✗	✗	N/A	✗	✗	✗
ARPF	✓	✓	N/A	✓	✓	T

Table A.70: Authentication properties of 5G-AKA from the **UE’s** point of view: All AUSFs compromisable by the adversary apart from ‘mine’: Not-My-X, **with SNID fix**

A.2.6 All ARPFs compromisable by the adversary apart from 'mine': Not-My-X

Party	Term	Result
UE	K_{SEAF}	✓
SEAF	K_{SEAF}	✓
AUSF	K_{SEAF}	✓
ARPF	K_{AUSF}	✓
	K	✓

Table A.71: Secrecy properties of 5G-AKA: All ARPFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✓	✗	N/A	✓	✗	✗
AUSF	✓	✗	N/A	✓	✗	✗
ARPF	✓	✗	N/A	✓	✗	✗

Table A.72: Authentication properties of 5G-AKA from the **UE's** point of view: All ARPFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
AUSF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.73: Authentication properties of 5G-AKA from the **serving network's** point of view: All ARPFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
UE	✓	✓	N/A	✓	T	T
SEAF	✓	✓	✓	✓	✓	T
ARPF	✓	✓	✓	✓	✓	T

Table A.74: Authentication properties of 5G-AKA from the **home network's** point of view: All ARPFs compromisable by the adversary apart from 'mine': Not-My-X

Role/Term	UE	SEAF	AUSF	ARPF	K_{SEAF}	Inj: K_{SEAF}
SEAF	✓	✓	N/A	✓	✓	T
AUSF	✓	✓	N/A	✓	✓	T
ARPF	✓	✓	N/A	✓	✓	T

Table A.75: Authentication properties of 5G-AKA from the **UE's** point of view: All ARPFs compromisable by the adversary apart from 'mine': Not-My-X, **with SNID fix**

B

Table of 5G acronyms

Acronym	Definition
AKC	Actor Key Compromise
5G-AC	5G Authentication Confirmation message
5G-AIA	Authentication Initiation Answer message
5G-AIR	Authentication Initiation Request message
5G-AKA	Fifth Generation Authentication and Key Agreement Protocol
5G-GUTI	5G-Globally Unique Temporary UE Identity (previously TMSI)
AMF	Core Access and Mobility Management Function
ARPF	Authentication credential Repository and Processing Function (This resides within the home network, and is often within an HSM)
AUSF	Authentication Server Function (within home network)
ECDH	Elliptic-Curve Diffie-Hellman
ECIES	Elliptic-Curve Integrated Encryption Scheme
EPS-AKA	Evolved Packet System Authentication and Key Agreement Protocol (A predecessor authentication protocol to 5G-AKA used by 4G/LTE)
HN	Home Network
HSM	Hardware Security Module
IMSI	International Mobile Subscriber Identity (LTE and older, now SUPI)
K	Long-term secret master key shared between USIM (within UE) and HN
K_{AUSF}	AUSF's anchor key
K_{SEAF}	SEAF's anchor key (derived directly from K_{AUSF})
KDF	Key Derivation Function (specified in [10])
MAC	Message Authentication Code
RAND	Random number generated by the ARPF
SEAF	Security Anchor Function (within serving network)
SIDF	Subscriber Identity De-concealing Function
SNID	Serving Network Identifier
SQN	Sequence Number
SUCI	Subscription Concealed Identifier (encrypted using ECIES)
SUPI	Subscription Permanent Identifier: previously "IMSI"
TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity (LTE and older)
TS	Technical Specification
TR	Technical Report
UE	User Equipment (e.g., mobile phone)
USIM	Universal Subscriber Identity Module (e.g., SIM Card)

Bibliography

[1] 3rd Generation Partnership Project 2 (3GPP2). 3GPP2 Specifications: TIA 41 X.S0004 Series, 2004. URL: https://www.3gpp2.org/Public_html/Specs/.

[2] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.902: 3G Security: Formal Analysis of the 3G Authentication Protocol*, 1999.

[3] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.120: Security Objectives and Principles*, March 2001. Version 4.0.0.

[4] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.200: 3G Security; Network Domain Security (NDS); Mobile Application Part (MAP) application layer security*, June 2007. Version 7.0.0.

[5] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TR 33.899: Study on the security aspects of the next generation system (SPECIFICATION WITHDRAWN)*, August 2017. Version 1.3.0.

[6] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.102: 3G security; Security architecture*, March 2017. Version 14.1.0.

[7] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.105: 3G Security; Cryptographic algorithm requirements*, March 2017. Version 14.1.0.

[8] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.501: Security Architecture and Procedures for 5G System*, December 2017. Version 0.7.0.

[9] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 23.060: General Packet Radio Service (GPRS); Service description; Stage 2*, March 2018. Version 15.2.0.

[10] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.220: Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)*, April 2018. Version 15.1.0.

[11] 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3). *TS 33.401: 3GPP System Architecture Evolution (SAE); Security architecture*, January 2018. Version 15.2.0.

[12] Martín Abadi. Secrecy by Typing in Security Protocols. *J. ACM*, 46(5):749–786, September 1999. URL: https://doi.acm.org/10.1145/324133.324266, doi:10.1145/324133.324266.

[13] Martín Abadi and Andrew D. Gordon. A Calculus for Cryptographic Protocols: The spi Calculus. *Inf. Comput.*, 148(1):1–70, 1999. URL: https://doi.org/10.1006/inco.1998.2740, doi:10.1006/inco.1998.2740.

[14] Mohammed Aly Abdrabou, Ashraf Diaa Eldien Elbayoumy, and Essam Abd El-Wanis. Lte authentication protocol (eps-aka) weaknesses solution. In *Intelligent Computing and Information Systems (ICICIS), 2015 IEEE Seventh International Conference on*, pages 434–441. IEEE, 2015. URL: https://ieeexplore.ieee.org/abstract/document/7397256/, doi:10.1109/IntelCIS.2015.7397256.

[15] S. Adee. The hunt for the kill switch. *IEEE Spectrum*, 45(5):34–39, May 2008. doi:10.1109/MSPEC.2008.4505310.

[16] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella Béguelin, and Paul Zimmermann. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 5–17, 2015. URL: https://doi.acm.org/10.1145/2810103.2813707, doi:10.1145/2810103.2813707.

[17] Na-Young Ahn and Dong Hoon Lee. Countermeasure against Side-Channel Attack in Shared Memory of TrustZone. *CoRR*, abs/1705.08279, 2017. URL: https://arxiv.org/abs/1705.08279, arXiv:1705.08279.

[18] Kamal Ali Alezabi, Fazirulhisyam Hashim, Shaiful Jahari Hashim, and Borhanuddin M Ali. An efficient authentication and key agreement protocol for 4G (LTE) networks. In *Region 10 Symposium, 2014 IEEE*, pages 502–507. IEEE, 2014. URL: https://ieeexplore.ieee.org/abstract/document/6863085/.

[19] Alliance for Telecommunications Industry Solutions. Glossary. URL: https://glossary.atis.org/.

[20] Fadi A. Aloul, Syed Zahidi, and Wassim El-Hajj. Two factor authentication using mobile phones. In *The 7th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009, Rabat, Morocco, May 10-13, 2009*, pages 641–644, 2009. URL: https://doi.org/10.1109/AICCSA.2009.5069395, doi:10.1109/AICCSA.2009.5069395.

[21] Stéphanie Alt, Pierre-Alain Fouque, Gilles Macario-Rat, Cristina Onete, and Benjamin Richard. A cryptographic analysis of UMTS/LTE AKA. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 18–35, 2016. URL: https://doi.org/10.1007/978-3-319-39555-5_2, doi:10.1007/978-3-319-39555-5_2.

[22] Tiago Alves and Don Felton. TrustZone: Integrated hardware and software security. *ARM White Paper*, 3(4):18–24, 2004. URL: https://unsyncopated.com/mirror/information_quarterly_trustzone.pdf.

[23] Raphael Amoah. *Formal security analysis of the DNP3-Secure Authentication Protocol.* PhD thesis, Queensland University of Technology, 2016.

[24] Raphael Amoah, Seyit Ahmet Çamtepe, and Ernest Foo. Formal modelling and analysis of DNP3 secure authentication. *J. Network and Computer Applications*, 59:345–360, 2016.

[25] Raphael Amoah, Seyit Ahmet Çamtepe, and Ernest Foo. Securing DNP3 Broadcast Communications in SCADA Systems. *IEEE Trans. Industrial Informatics*, 12(4):1474–1485, 2016.

[26] Raphael Amoah, Suriadi Suriadi, Seyit Ahmet Çamtepe, and Ernest Foo. Security analysis of the non-aggressive challenge response of the DNP3 protocol using a CPN model. In *IEEE International Conference on Communications, ICC 2014*, pages 827–833, 2014.

[27] June Andronick, David Greenaway, and Kevin Elphinstone. Towards proving security in the presence of large untrusted components. In *5th International Workshop on Systems Software Verification, SSV'10, Vancouver, BC, Canada, October 6-7, 2010*, 2010. URL: https://www.usenix.org/conference/ssv10/towards-proving-security-presence-large-untrusted-components.

[28] Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 205–216, 2012. URL: https://doi.acm.org/10.1145/2382196.2382221, doi:10.1145/2382196.2382221.

[29] Jari Arkko, Vesa Lehtovirta, and Pasi Eronen. Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA'). *RFC*, 5448:1–29, 2009. URL: https://tools.ietf.org/html/rfc5448, doi:10.17487/RFC5448.

[30] Jari Arkko, Karl Norrman, Mats Näslund, and Bengt Sahlin. A USIM Compatible 5G AKA Protocol with Perfect Forward Secrecy. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 1205–1209, 2015. URL: https://doi.org/10.1109/Trustcom.2015.506, doi:10.1109/Trustcom.2015.506.

[31] Alessandro Armando, Wihem Arsac, Tigran Avanesov, Michele Barletta, Alberto Calvi, Alessandro Cappai, Roberto Carbone, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Gabriel Erzse, Simone Frau, Marius Minea, Sebastian Mödersheim, David von Oheimb, Giancarlo Pellegrino, Serena Elisa Ponta, Marco Rocchetto, Michaël Rusinowitch, Mohammad Torabi Dashti, Mathieu Turuani, and Luca Viganò. The AVANTSSAR Platform for the Automated Validation of Trust and Security of Service-Oriented Architectures. In *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, pages 267–282, 2012. URL: https://doi.org/10.1007/978-3-642-28756-5_19, doi:10.1007/978-3-642-28756-5_19.

[32] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Fundamental concepts of computer system dependability. In *Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments*, pages 1–16, 2001.

[33] Abdul Bais, Walter T Penzhorn, and Peter Palensky. Evaluation of umts security architecture and services. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 570–575. IEEE, 2006.

[34] David A. Basin and Cas Cremers. Know Your Enemy: Compromising Adversaries in Protocol Analysis. *ACM Trans. Inf. Syst. Secur.*, 17(2):7:1–7:31, 2014. URL: https://doi.acm.org/10.1145/2658996, doi:10.1145/2658996.

[35] David A. Basin, Cas Cremers, Jannik Dreier, Simon Meier, Sasa Radomirovic, Ralf Sasse, Lara Schmid, and Benedikt Schmidt. The Tamarin Prover Manual, 2016. Creative Commons: Attribution-NonCommercial-ShareAlike 4.0 International License. URL: `https://tamarin-prover.github.io/manual/book/001_introduction.html`.

[36] David A. Basin, Cas Cremers, and Marko Horvat. Actor Key Compromise: Consequences and Countermeasures. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 244–258, 2014. URL: `https://doi.org/10.1109/CSF.2014.25`, `doi:10.1109/CSF.2014.25`.

[37] David A. Basin, Cas Cremers, and Simon Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013. URL: `https://doi.org/10.3233/JCS-130472`, `doi:10.3233/JCS-130472`.

[38] David A. Basin, Cas Cremers, Kunihiko Miyazaki, Sasa Radomirovic, and Dai Watanabe. Improving the Security of Cryptographic Protocol Standards. *IEEE Security & Privacy*, 13(3):24–31, 2015.

[39] David A. Basin, Cas J. F. Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. ARPKI: attack resilient public-key infrastructure. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 382–393, 2014. URL: `https://doi.org/10.1145/2660267.2660298`, `doi:10.1145/2660267.2660298`.

[40] David A. Basin, Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, Ralf Sasse, and Vincent Stettler. Formal Analysis of 5G Authentication. *CoRR*, abs/1806.10360, 2018. URL: `https://arxiv.org/abs/1806.10360`, `arXiv:1806.10360`.

[41] David A. Basin, Sasa Radomirovic, and Lara Schmid. Modeling human errors in security protocols. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 325–340, 2016. URL: `https://doi.org/10.1109/CSF.2016.30`, `doi:10.1109/CSF.2016.30`.

[42] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 232–249, 1993. URL: `https://doi.org/10.1007/3-540-48329-2_21`, `doi:10.1007/3-540-48329-2_21`.

[43] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. *CoRR*, abs/1407.3561, 2014. URL: https://arxiv.org/abs/1407.3561, arXiv:1407.3561.

[44] Gal Beniamini. Trust Issues: Exploiting TrustZone TEEs. Google Project Zero Blog, 2017. URL: https://googleprojectzero.blogspot.co.uk/2017/07/trust-issues-exploiting-trustzone-tees.html.

[45] Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. Dual EC: A Standardized Back Door. In *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, pages 256–281, 2016. URL: https://doi.org/10.1007/978-3-662-49301-4_17, doi:10.1007/978-3-662-49301-4_17.

[46] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS. In *2014 IEEE Symposium on Security and Privacy*, pages 98–113, 2014.

[47] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, pages 154–170, 1999. URL: https://doi.org/10.1007/3-540-49162-7_12, doi:10.1007/3-540-49162-7_12.

[48] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In *Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, March 1-3, 1999, Proceedings*, pages 154–170, 1999. URL: https://doi.org/10.1007/3-540-49162-7_12, doi:10.1007/3-540-49162-7_12.

[49] Bruno Blanchet. A Computationally Sound Mechanized Prover for Security Protocols. *IEEE Trans. Dependable Sec. Comput.*, 5(4):193–207, 2008. URL: https://doi.org/10.1109/TDSC.2007.1005, doi:10.1109/TDSC.2007.1005.

[50] Bruno Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif. In *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*, pages 54–87, 2013. URL: https://doi.org/10.1007/978-3-319-10082-1_3, doi:10.1007/978-3-319-10082-1_3.

[51] Bruno Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016. URL: https://doi.org/10.1561/3300000004, doi:10.1561/3300000004.

[52] Dan Boneh. The Decision Diffie-Hellman Problem. In *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, pages 48–63, 1998. URL: https://doi.org/10.1007/BFb0054851, doi:10.1007/BFb0054851.

[53] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 37–51, 1997. URL: https://doi.org/10.1007/3-540-69053-0_4, doi:10.1007/3-540-69053-0_4.

[54] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2003. URL: https://doi.org/10.1007/978-3-662-09527-0, doi:10.1007/978-3-662-09527-0.

[55] John G. Brainard, Ari Juels, Ronald L. Rivest, Michael Szydlo, and Moti Yung. Fourth-factor authentication: somebody you know. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 168–178, 2006. URL: https://doi.acm.org/10.1145/1180405.1180427, doi:10.1145/1180405.1180427.

[56] Sergey Bratus, Adam J Crain, Sven M Hallberg, Daniel P Hirsch, Meredith L Patterson, Maxwell Koo, and Sean W Smith. Implementing a vertically hardened dnp3 control stack for power applications. In *Proceedings of the 2nd Annual Industrial Control System Security Workshop*, pages 45–53. ACM, 2016.

[57] Chris Brzuska and Håkon Jacobsen. A modular security analysis of EAP and IEEE 802.11. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, pages 335–365, 2017. URL: https://doi.org/10.1007/978-3-662-54388-7_12, doi:10.1007/978-3-662-54388-7_12.

[58] Somnath Chakrabarti, Rebekah Leslie-Hurd, Mona Vij, Frank McKeen, Carlos V. Rozas, Dror Caspi, Ilya Alexandrovich, and Ittai Anati. Intel® Software Guard Extensions (Intel® SGX) Architecture for Oversubscription of Secure Memory in a Virtualized Environment. In *Proceedings of the Hardware and Architectural Support for Security and Privacy, HASP@ISCA 2017, Toronto, ON, Canada, June 25, 2017*, pages 7:1–7:8, 2017. URL: https://doi.acm.org/10.1145/3092627.3092634, doi:10.1145/3092627.3092634.

[59] Richard B Chase and Douglas M Stewart. Make your service fail-safe. *MIT Sloan Management Review*, 35(3):35, 1994.

[60] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A Systematic Analysis of the Juniper Dual EC Incident. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 468–479, 2016. URL: https://doi.acm.org/10.1145/2976749.2978395, doi:10.1145/2976749.2978395.

[61] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A Formal Security Analysis of the Signal Messaging Protocol. *IACR Cryptology ePrint Archive*, 2016:1013, 2016. URL: https://eprint.iacr.org/2016/1013.

[62] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On Post-compromise Security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178, 2016. URL: https://doi.org/10.1109/CSF.2016.19, doi:10.1109/CSF.2016.19.

[63] Hubert Comon-Lundh and Stéphanie Delaune. The finite variant property: How to get rid of some algebraic properties. In *Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings*, pages 294–307, 2005. URL: https://doi.org/10.1007/978-3-540-32033-3_22, doi:10.1007/978-3-540-32033-3_22.

[64] J Adam Crain and Sergey Bratus. Bolt-on security extensions for industrial control system protocols: A case study of dnp3 sav5. *IEEE Security & Privacy*, 13(3):74–79, 2015.

[65] Cas Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 119–128, 2008. URL: https://doi.acm.org/10.1145/1455770.1455787, doi:10.1145/1455770.1455787.

[66] Cas Cremers. Session-StateReveal is stronger than eCK's EphemeralKeyReveal: using automatic analysis to attack the NAXOS protocol. *IJACT*, 2(2):83–99, 2010. URL: https://doi.org/10.1504/IJACT.2010.038304, doi:10.1504/IJACT.2010.038304.

[67] Cas Cremers. Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, pages 80–91, 2011. URL: https://doi.acm.org/10.1145/1966913.1966925, doi:10.1145/1966913.1966925.

[68] Cas Cremers. Key Exchange in IPsec Revisited: Formal Analysis of IKEv1 and IKEv2. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, pages 315–334, 2011. URL: https://doi.org/10.1007/978-3-642-23822-2_18, doi:10.1007/978-3-642-23822-2_18.

[69] Cas Cremers and Martin Dehnel-Wild. 5G AKA Tamarin Models, 2018. Included electronically. URL: https://www.secblue.net/ndss/5g-aka.zip.

[70] Cas Cremers, Martin Dehnel-Wild, and Kevin Milner. Secure Authentication in the Grid: A Formal Analysis of DNP3: SAv5. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*, pages 389–407, 2017. URL: https://doi.org/10.1007/978-3-319-66402-6_23, doi:10.1007/978-3-319-66402-6_23.

[71] Cas Cremers, Martin Dehnel-Wild, and Kevin Milner. DNP3 Secure Authentication v5 Tamarin Model (with Asymmetric mode of the Update Key Change Protocol), 2018. URL: https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/jcs18/.

[72] Cas Cremers, Luke Garratt, Stanislav Smyshlyaev, Nick Sullivan, and Christopher Wood. Randomness improvements for security protocols, draft-irtf-cfrg-randomness-improvements-00. Internet-Draft, Internet Engineering Task Force (IETF), March 2018. URL: https://www.ietf.org/id/draft-irtf-cfrg-randomness-improvements-00.txt.

[73] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of TLS 1.3: 0-rtt, resumption and delayed authentication. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 470–485, 2016. URL: https://doi.org/10.1109/SP.2016.35, doi:10.1109/SP.2016.35.

[74] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer, 2012. URL: https://doi.org/10.1007/978-3-540-78636-8, doi:10.1007/978-3-540-78636-8.

[75] Cas J. F. Cremers. On the protocol composition logic PCL. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 66–76, 2008. URL: https://doi.org/10.1145/1368310.1368324, doi:10.1145/1368310.1368324.

[76] Sander de Kievit. S3-181468-v3 - LS to CT4 on avoiding race condition in 5G AKA. 3GPP SA WG3 Meeting #91, 16 – 20 April 2018, Belgrade, Serbia. URL: https://www.3gpp.org/ftp/TSG_SA/WG3_Security/TSGS3_91_Belgrade/Docs/S3-181468.zip.

[77] Jean Paul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, Giorgia Azzurra Marson, Arno Mittelbach, and Kenneth G. Paterson. Unpicking PLAID - A Cryptographic Analysis of an ISO-Standards-Track Authentication Protocol. In *Security Standardisation Research - First International Conference, SSR 2014*, pages 1–25, 2014.

[78] The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2004. URL: https://coq.inria.fr.

[79] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976. URL: https://doi.org/10.1109/TIT.1976.1055638, doi:10.1109/TIT.1976.1055638.

[80] Christopher Dilloway and Gavin Lowe. Specifying secure transport channels. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium, CSF 2008, Pittsburgh, Pennsylvania, 23-25 June 2008*, pages 210–223, 2008. URL: https://doi.org/10.1109/CSF.2008.14, doi:10.1109/CSF.2008.14.

[81] Steven X Ding. *Model-based fault diagnosis techniques: design schemes, algorithms, and tools*. Springer Science & Business Media, 2008.

[82] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004. URL: https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/full_papers/dingledine/dingledine.pdf.

[83] DNP Users Group. A DNP3 Protocol Primer (Revision A), 2005. URL: https://www.dnp.org/AboutUs/DNP3%20Primer%20Rev%20A.pdf.

[84] NTT Docomo. S3-181190 - CHANGE REQUEST - Clarification for perceived potential AKA race condition. 3GPP SA WG3 Meeting #91, 16 – 20 April 2018, Belgrade, Serbia. URL: https://www.3gpp.org/ftp/TSG_SA/WG3_Security/TSGS3_91_Belgrade/Docs/S3-181190.zip.

[85] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message Transmission with Reverse Firewalls - Secure Communication on Corrupted Machines. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 341–372, 2016. URL: https://doi.org/10.1007/978-3-662-53018-4_13, doi:10.1007/978-3-662-53018-4_13.

[86] Danny Dolev and Andrew Chi-Chih Yao. On the Security of Public Key Protocols. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 350–357, 1981. URL: https://doi.org/10.1109/SFCS.1981.32, doi:10.1109/SFCS.1981.32.

[87] Kevin Dooley. *Designing Large Scale Lans: Help for Network Designers.* " O'Reilly Media, Inc.", 2001.

[88] Benjamin Dowling and Kenneth G. Paterson. A Cryptographic Analysis of the WireGuard Protocol. *IACR Cryptology ePrint Archive*, 2018:80, 2018. URL: https://eprint.iacr.org/2018/080.

[89] Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols. In *Principles of Security and Trust - 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 117–140, 2017. URL: https://doi.org/10.1007/978-3-662-54455-6_6, doi:10.1007/978-3-662-54455-6_6.

[90] N Durgin, J Mitchell, A Scedrov, and P Lincoln. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*, 1999. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.3546&rep=rep1&type=pdf.

[91] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. A Taxonomy of Attacks on the DNP3 Protocol. In *Critical Infrastructure Protection III - Third Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection.*, pages 67–81, 2009.

[92] Santiago Escobar, Ralf Sasse, and José Meseguer. Folding variant narrowing and optimal variant termination. *J. Log. Algebr. Program.*, 81(7-8):898–928, 2012. URL: https://doi.org/10.1016/j.jlap.2012.01.002, doi:10.1016/j.jlap.2012.01.002.

[93] Victor Fajardo, Jari Arkko, John Loughney, and Glen Zorn. Diameter base protocol. *RFC*, 6733:1–152, 2012. URL: https://doi.org/10.17487/RFC6733, doi:10.17487/RFC6733.

[94] Michèle Feltz and Cas Cremers. On the limits of authenticated key exchange security with an application to bad randomness. *IACR Cryptology ePrint Archive*, 2014:369, 2014. URL: https://eprint.iacr.org/2014/369.

[95] Michael Gegick and Sean Barnum. US-CERT: Failing Securely, December 2005. URL: https://www.us-cert.gov/bsi/articles/knowledge/principles/failing-securely.

[96] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009. URL: https://doi.acm.org/10.1145/1536414.1536440, doi:10.1145/1536414.1536440.

[97] Thomas Gibson-Robinson. On The Refinement Closure Of Information-Flow Properties. Master's thesis, University of Oxford, 2009.

[98] Thomas Gibson-Robinson, Philip J. Armstrong, Alexandre Boulgakov, and A. W. Roscoe. FDR3 - A modern refinement checker for CSP. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pages 187–201, 2014. URL: https://doi.org/10.1007/978-3-642-54862-8_13, doi:10.1007/978-3-642-54862-8_13.

[99] Oscar González, H. Shrikumar, John A. Stankovic, and Krithi Ramamritham. Adaptive fault tolerance and graceful degradation under dynamic hard real-time scheduling. In *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97), December 3-5, 1997, San Francisco, CA, USA*, pages 79–89, 1997. URL: https://doi.org/10.1109/REAL.1997.641271, doi:10.1109/REAL.1997.641271.

[100] Andrew D. Gordon and Alan Jeffrey. Secrecy Despite Compromise: Types, Cryptography, and the Pi-Calculus. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings*, pages 186–201, 2005. URL: https://doi.org/10.1007/11539452_17, doi:10.1007/11539452_17.

[101] Vipul Goyal. Secure composition of cryptographic protocols. In *Information Systems Security - 7th International Conference, ICISS 2011, Kolkata, India, December 15-19, 2011, Procedings*, page 71, 2011. URL: https://doi.org/10.1007/978-3-642-25560-1_4, doi:10.1007/978-3-642-25560-1_4.

[102] GSMA. Brief History of GSM & the GSMA, 2017. URL: https://www.gsma.com/aboutus/history.

[103] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *CHES 2004*, pages 119–132, 2004.

[104] Clemens Hlauschek, Markus Gruber, Florian Fankhauser, and Christian Schanes. Prying Open Pandora's Box: KCI Attacks against TLS. In *9th USENIX Workshop on Offensive Technologies, WOOT '15, Washington, DC, USA, August 10-11, 2015.*, 2015. URL: https://www.usenix.org/conference/woot15/workshop-program/presentation/hlauschek.

[105] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978. URL: https://doi.acm.org/10.1145/359576.359585, doi:10.1145/359576.359585.

[106] Syed Rafiul Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. LTEInspector: A Systematic Approach for Adversarial Testing of 4G LTE. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018*, 2018.

[107] Anton A Huurdeman. *The worldwide history of telecommunications*. John Wiley & Sons, 2003.

[108] IEC. IEC/TS 62351-2:2008, Power systems management and associated information exchange – Data and communications security – Part 2: Glossary of terms. *International Electrotechnical Commission*, 2008.

[109] IEC. IEC/TS 62351-5:2013, Power systems management and associated information exchange – Data and communications security – Part 5: Security for IEC 60870-5 and derivatives. *International Electrotechnical Commission*, 2013.

[110] IEEE. 1815-2012 - IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3). *IEEE Std 1815-2012 (Revision of IEEE Std 1815-2010)*, pages 1–821, 2012. URL: https://ieeexplore.ieee.org/document/6327578/.

[111] ISO/IEC. ISO/IEC 9798-1:1997, Part 1: General, 1997. URL: https://www.iso.org/standard/27743.html.

[112] Enrique Cobo Jimenez, Prajwol Kumar Nakarmi, Mats Näslund, and Karl Norrman. Subscription identifier privacy in 5g systems. In *International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2017, Avignon, France, May 17-19, 2017*, pages 1–8, 2017. URL: https://doi.ieeecomputersociety.org/10.1109/MoWNet.2017.8045947, doi:10.1109/MoWNet.2017.8045947.

[113] Jan Jürjens. Secrecy-preserving refinement. In *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings*, pages 135–152, 2001. URL: https://doi.org/10.1007/3-540-45251-6_8, doi:10.1007/3-540-45251-6_8.

[114] Allaa Kamil and Gavin Lowe. Specifying and modelling secure channels in strand spaces. In *Formal Aspects in Security and Trust, 6th International Workshop, FAST 2009, Eindhoven, The Netherlands, November 5-6, 2009, Revised Selected Papers*, pages 233–247, 2009. URL: https://doi.org/10.1007/978-3-642-12459-4_17, doi:10.1007/978-3-642-12459-4_17.

[115] Allaa Kamil and Gavin Lowe. Understanding abstractions of secure channels. In *Formal Aspects of Security and Trust - 7th International Workshop, FAST 2010, Pisa, Italy, September 16-17, 2010. Revised Selected Papers*, pages 50–64, 2010. URL: https://doi.org/10.1007/978-3-642-19751-2_4, doi:10.1007/978-3-642-19751-2_4.

[116] Allaa Kamil and Gavin Lowe. Analysing TLS in the strand spaces model. *Journal of Computer Security*, 19(5):975–1025, 2011. URL: https://doi.org/10.3233/JCS-2011-0429, doi:10.3233/JCS-2011-0429.

[117] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[118] John Kelsey, Bruce Schneier, and David A. Wagner. Protocol Interactions and the Chosen Protocol Attack. In *Security Protocols, 5th Workshop*, pages 91–104, 1997.

[119] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: formal verification of an OS kernel. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009*, pages 207–220, 2009. URL: https://doi.acm.org/10.1145/1629575.1629596, doi:10.1145/1629575.1629596.

[120] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996. URL: https://doi.org/10.1007/3-540-68697-5_9, doi:10.1007/3-540-68697-5_9.

[121] Geir M. Køien. Mutual entity authentication for LTE. In *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference, IWCMC 2011, Istanbul, Turkey, 4-8 July, 2011*, pages 689–694, 2011. URL: https://doi.org/10.1109/IWCMC.2011.5982630, doi:10.1109/IWCMC.2011.5982630.

[122] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 400–425, 2003. URL: https://doi.org/10.1007/978-3-540-45146-4_24, doi:10.1007/978-3-540-45146-4_24.

[123] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 546–566, 2005. URL: https://doi.org/10.1007/11535218_33, doi:10.1007/11535218_33.

[124] Brian Krebs. Secret Service Warns of 'Periscope' Skimmers. Krebs on Security Blog, 2016. URL: https://krebsonsecurity.com/2016/09/secret-service-warns-of-periscope-skimmers/.

[125] Daniel Kroening and Michael Tautschnig. CBMC - C bounded model checker. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, pages 389–391, 2014. URL: https://doi.org/10.1007/978-3-642-54862-8_26, doi:10.1007/978-3-642-54862-8_26.

[126] Markus G. Kuhn and Ross J. Anderson. Soft tempest: Hidden data transmission using electromagnetic emanations. In *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, pages 124–142, 1998. URL: https://doi.org/10.1007/3-540-49380-8_10, doi:10.1007/3-540-49380-8_10.

[127] Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings*, pages 1–16, 2007. URL: https://doi.org/10.1007/978-3-540-75670-5_1, doi:10.1007/978-3-540-75670-5_1.

[128] Jean-Claude Laprie. Dependable computing and fault tolerance: Concepts and terminology. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995, (Reprinted from FTCS, 1985)*. IEEE, 1985.

[129] Ben Laurie, Adam Langley, and Emilia Käsper. Certificate transparency. *RFC*, 6962:1–27, 2013. URL: https://doi.org/10.17487/RFC6962, doi:10.17487/RFC6962.

[130] Gavin Lowe. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995. URL: https://doi.org/10.1016/0020-0190(95)00144-2, doi:10.1016/0020-0190(95)00144-2.

[131] Gavin Lowe. A Hierarchy of Authentication Specifications. In *Proceedings 10th Computer Security Foundations Workshop*, pages 31–43, Jun 1997.

[132] Gavin Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1-2):53–84, 1998. URL: https://content.iospress.com/articles/journal-of-computer-security/jcs106.

[133] Andrew Martin. The ten-page introduction to Trusted Computing. Department of Computer Science Website, University of Oxford, 2008. URL: https://www.cs.ox.ac.uk/files/1873/RR-08-11.PDF.

[134] Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneel. A cross-protocol attack on the TLS protocol. In *ACM CCS'12*, pages 62–72, 2012.

[135] Vasilios Mavroudis, Andrea Cerulli, Petr Svenda, Dan Cvrcek, Dusan Klinec, and George Danezis. A touch of evil: High-assurance cryptographic hardware from untrusted components. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1583–1600, 2017. URL: https://doi.org/10.1145/3133956.3133961, doi:10.1145/3133956.3133961.

[136] Rita Mayer-Sommer. Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, pages 78–92, 2000. URL: https://doi.org/10.1007/3-540-44499-8_6, doi:10.1007/3-540-44499-8_6.

[137] Simon Meier. *Advancing automated security protocol verification*. PhD thesis, ETH Zurich, 2013. URL: https://doi.org/10.3929/ethz-a-009790675.

[138] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 696–701, 2013. URL: https://doi.org/10.1007/978-3-642-39799-8_48, doi:10.1007/978-3-642-39799-8_48.

[139] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. URL: http://cacr.uwaterloo.ca/hac/.

[140] Kevin Milner. *Detecting the Misuse of Secrets: Foundations, Protocols, and Verification*. PhD thesis, University of Oxford, 2018.

[141] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 657–686, 2015. URL: https://doi.org/10.1007/978-3-662-46803-6_22, doi:10.1007/978-3-662-46803-6_22.

[142] miTLS. A Zoo of TLS attacks, 2015. URL: https://mitls.org/pages/attacks.

[143] Hyeran Mun, Kyusuk Han, and Kwangjo Kim. 3G-WLAN interworking: security analysis and new authentication and key agreement based on EAP-AKA. In *2009 Wireless Telecommunications Symposium, WTS 2009, Prague, Czech Republic, April 22-24, 2009*, pages 1–8, 2009. URL: https://doi.org/10.1109/WTS.2009.5068983, doi:10.1109/WTS.2009.5068983.

[144] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. URL: https://doi.org/10.1007/3-540-45949-9, doi:10.1007/3-540-45949-9.

[145] Karl Norrman, Mats Näslund, and Elena Dubrova. Protecting IMSI and User Privacy in 5G Networks. In *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications, MobiMedia 2016, Xi'an, China, June 18-20, 2016*, pages 159–166, 2016. URL: https://dl.acm.org/citation.cfm?id=3021415.

[146] Piers O'Hanlon, Ravishankar Borgaonkar, and Lucca Hirschi. Mobile subscriber wifi privacy. In *2017 IEEE Security and Privacy Workshops, SP Workshops 2017*, pages 169–178, 2017. URL: https://doi.org/10.1109/SPW.2017.14, doi:10.1109/SPW.2017.14.

[147] Kenneth G. Paterson and Thyla van der Merwe. Reactive and Proactive Standardisation of TLS. In *Security Standardisation Research*, pages 160–186, 2016.

[148] Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991. URL: https://doi.org/10.1007/3-540-46766-1_9, doi:10.1007/3-540-46766-1_9.

[149] David Perez and Jose Pico. A practical attack against gprs/edge/umts/hspa mobile data communications. *Black Hat DC*, 2011. URL: https://media.blackhat.com/bh-dc-11/Perez-Pico/BlackHat_DC_2011_Perez-Pico_Mobile_Attacks-wp.pdf.

[150] Ray A. Perlner and David A. Cooper. Quantum resistant public key cryptography: a survey. In *IDtrust 2009, Proceedings of the 8th Symposium on Identity and Trust on the Internet, April 14-16, 2009, Gaithersburg, Maryland, USA*, pages 85–93, 2009. URL: https://doi.org/10.1145/1527017.1527028, doi:10.1145/1527017.1527028.

[151] Trevor Perrin. Axolotl Ratchet. Github Wiki, 2015. URL: https://github.com/trevp/axolotl/wiki(AccessedSeptember2015).

[152] Rachel Player. *Parameter selection in lattice-based cryptography*. PhD thesis, Royal Holloway, University of London, 2018.

[153] Christina Pöpper, David A. Basin, Srdjan Capkun, and Cas Cremers. Keeping data secret under full compromise using porter devices. In *Twenty-Sixth Annual Computer Security Applications Conference, ACSAC 2010, Austin, Texas, USA, 6-10 December 2010*, pages 241–250, 2010. URL: https://doi.acm.org/10.1145/1920261.1920297, doi:10.1145/1920261.1920297.

[154] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, draft-ietf-tls-tls13-28. Internet-Draft, Internet Engineering Task Force (IETF), March 2018. URL: https://www.ietf.org/id/draft-ietf-tls-tls13-28.txt.

[155] David Rupprecht, Adrian Dabrowski, Thorsten Holz, Edgar R. Weippl, and Christina Pöpper. On security research towards future mobile network generations. *CoRR*, abs/1710.08932, 2017. URL: https://arxiv.org/abs/1710.08932, arXiv:1710.08932.

[156] Travis Russell. *Signaling System #7*. McGraw-Hill Professional, 3rd edition, 2000. URL: https://dl.acm.org/citation.cfm?id=540665.

[157] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[158] Titos Saridakis. Design patterns for fault containment. In *Proceedings of the 8th European Conference on Pattern Languages of Programms (EuroPLoP '2003), Irsee, Germany, June 25-29, 2003.*, pages 493–520, 2003. URL: https://hillside.net/europlop/HillsideEurope/Papers/EuroPLoP2003/2003_Saridakis_DesignPatternsforFaultContainment.pdf.

[159] Benedikt Schmidt. *Formal analysis of key exchange protocols and physical protocols*. PhD thesis, ETH Zurich, 2012. URL: https://doi.org/10.3929/ethz-a-009898924.

[160] Benedikt Schmidt, Simon Meier, Cas Cremers, and David A. Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 78–94, 2012. URL: https://doi.org/10.1109/CSF.2012.25, doi:10.1109/CSF.2012.25.

[161] Bruce Schneier. Did NSA Put a Secret Backdoor in New Encryption Standard? Wired Magazine, 2007. URL: https://archive.wired.com/politics/security/commentary/securitymatters/2007/11/securitymatters_1115.

[162] Bruce Schneier. Random Number Bug in Debian Linux, May 2008. URL: https://www.schneier.com/blog/archives/2008/05/random_number_b.html.

[163] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. Surreptitiously Weakening Cryptographic Systems. *IACR Cryptology ePrint Archive*, 2015:97, 2015. URL: https://eprint.iacr.org/2015/097.

[164] Simha Sethumadhavan, Adam Waksman, Matthew Suozzo, Yipeng Huang, and Julianna Eum. Trustworthy hardware from untrusted components. *Commun. ACM*, 58(9):60–71, 2015. URL: https://doi.acm.org/10.1145/2699412, doi:10.1145/2699412.

[165] Thomas Sewell, Simon Winwood, Peter Gammie, Toby C. Murray, June Andronick, and Gerwin Klein. seL4 Enforces Integrity. In *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands, August 22-25, 2011. Proceedings*, pages 325–340, 2011. URL: https://doi.org/10.1007/978-3-642-22863-6_24, doi:10.1007/978-3-642-22863-6_24.

[166] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979. URL: https://doi.acm.org/10.1145/359168.359176, doi:10.1145/359168.359176.

[167] Yaron Sheffer, Ralph Holz, and Peter Saint-Andre. Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS). *RFC*, 7457:1–13, 2015. URL: https://doi.org/10.17487/RFC7457, doi:10.17487/RFC7457.

[168] Di Shen. Exploiting Trustzone on Android. BlackHat Technical Report, 2015. URL: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android-wp.pdf.

[169] Robert W. Shirey. Internet security glossary. *RFC*, 2828:1–212, 2000. URL: https://doi.org/10.17487/RFC2828, doi:10.17487/RFC2828.

[170] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. URL: https://doi.org/10.1137/S0097539795293172, doi:10.1137/S0097539795293172.

[171] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004. URL: https://eprint.iacr.org/2004/332.

[172] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and et al. Announcing the first SHA1 collision, 2017. URL: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html.

[173] Chunyu Tang, David A. Naumann, and Susanne Wetzel. Analysis of authentication and key establishment in inter-generational mobile telephony. In *10th IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, HPCC/EUC 2013, Zhangjiajie, China, November 13-15, 2013*, pages 1605–1614, 2013. URL: https://doi.org/10.1109/HPCC.and.EUC.2013.226, doi:10.1109/HPCC.and.EUC.2013.226.

[174] R. Tawde, A. Nivangune, and M. Sankhe. Cyber security in smart grid SCADA automation systems. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–5, 2015.

[175] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*, pages 160–171, 1998. URL: https://doi.org/10.1109/SECPRI.1998.674832, doi:10.1109/SECPRI.1998.674832.

[176] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Security and Privacy - 1998 IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 3-6, 1998, Proceedings*, pages 160–171, 1998. URL: https://doi.org/10.1109/SECPRI.1998.674832, doi:10.1109/SECPRI.1998.674832.

[177] Joe-Kai Tsay and Stig Fr. Mjølsnes. A vulnerability in the UMTS and LTE authentication and key agreement protocols. In *Computer Network Security - 6th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2012, St. Petersburg, Russia, October 17-19, 2012. Proceedings*, pages 65–76, 2012. URL: https://doi.org/10.1007/978-3-642-33704-8_6, doi:10.1007/978-3-642-33704-8_6.

[178] Lisa Vaas. NSA intercepts routers, servers to slip in backdoors for overseas surveillance. Sophos Naked Security Blog, 2014. URL: https://nakedsecurity.sophos.com/2014/05/14/nsa-intercepts-routers-servers-to-slip-in-backdoors-for-overseas-surveillance/.

[179] Klaus Vedder. GSM: Security, services, and the SIM. In *State of the art in Applied Cryptography*, pages 224–240. Springer, 1998.

[180] Luca Viganò. Automated Security Protocol Analysis With the AVISPA Tool. *Electr. Notes Theor. Comput. Sci.*, 155:61–86, 2006. URL: https://doi.org/10.1016/j.entcs.2005.11.052, doi:10.1016/j.entcs.2005.11.052.

[181] Vodafone. S3-180727 - pCR to TS33.501 - Session binding to prevent potential 5G-AKA vulnerability. 3GPP SA WG3 Meeting #90Bis, 26 Feb – 2 March 2018, San Diego, US. URL: https://www.3gpp.org/ftp/TSG_SA/WG3_Security/TSGS3_90Bis_SanDiego/Docs/S3-180727.zip.

[182] David A. Wheeler. Preventing Heartbleed. *IEEE Computer*, 47(8):80–83, 2014. URL: https://doi.org/10.1109/MC.2014.217, doi:10.1109/MC.2014.217.

[183] David Wong. Key Compromise Impersonation attacks (KCI). Blog Post, September 2016. URL: https://www.cryptologie.net/article/372/key-compromise-impersonation-attacks-kci/.

[184] Maj Gen Margaret H. Woodward. US Air Force Instruction 91-104: Nuclear Surety Tamper Control and Detection Programs, April 2013. URL: https://fas.org/irp/doddir/usaf/afi91-104.pdf.

[185] Thomas D. Wu, Michael Malkin, and Dan Boneh. Building intrusion-tolerant applications. In *Proceedings of the 8th USENIX Security Symposium, Washington, D.C., August 23-26, 1999*, 1999. URL: https://www.usenix.org/conference/8th-usenix-security-symposium/building-intrusion-tolerant-applications.

[186] Kaiyuan Yang, Matthew Hicks, Qing Dong, Todd M. Austin, and Dennis Sylvester. A2: analog malicious hardware. In *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, pages 18–37, 2016. URL: https://doi.org/10.1109/SP.2016.10, doi:10.1109/SP.2016.10.

[187] Paul Yousef. GSM-security: a survey and evaluation of the current situation. Master's thesis, Institutionen för systemteknik, Linköping Universitet, Sweden, 2004. LiTH-ISY-EX-3559-2004. URL: https://www.diva-portal.org/smash/get/diva2:19603/fulltext01.

[188] Muxiang Zhang and Yuguang Fang. Security analysis and enhancements of 3GPP authentication and key agreement protocol. *IEEE Trans. Wireless Communications*, 4(2):734–742, 2005. URL: https://doi.org/10.1109/TWC.2004.842941, doi:10.1109/TWC.2004.842941.

[189] Yongbin Zhou and Dengguo Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005:388, 2005. URL: https://eprint.iacr.org/2005/388.