Improving Automated Protocol Verification: Real World Cryptography

Dennis Jackson

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2020

Abstract

Analysing the security of cryptographic protocols by hand is a challenging endeavour. It requires substantial expertise, weeks of intensive effort and the resulting proof of security often arrives long after the protocol design has been finalised and even deployed. However, automated protocol verification tools, such as Tamarin and ProVerif, offer a compelling alternative as they require less expertise, promise quicker results and have been used to successfully analyse complex real world protocols such as TLS, 5G and Signal.

These tools are built on a idealised model of cryptography, termed the symbolic model, which requires strong assumptions about the properties of cryptographic primitives. Consequently, automated protocol analyses may miss attacks which rely on properties of real primitives which are missing from the symbolic model. Furthermore, some protocols are not amenable to automated analysis as they use cryptographic primitives for which no suitable symbolic model currently exists.

This motivates natural research questions: How closely does the contemporary symbolic model approximate the real world behaviour of common cryptographic primitives? Can we improve the accuracy and breadth of the symbolic model by developing new approaches to underlying formalism? In this thesis, we set out to address these questions for two popular cryptographic primitives: digital signatures and Diffie-Hellman groups.

The symbolic model of digital signatures was first published nearly two decades ago and is widely used. We uncover a startling mismatch between the standard cryptographic definition of signature scheme security and the symbolic description. We repair this mismatch through the development of a novel symbolic model which we use to discover a number of new attacks on deployed protocols. We also document a number of previous analyses which missed these attacks due to their traditional symbolic model.

Next we consider Diffie-Hellman groups. Unlike digital signatures, symbolic Diffie-Hellman models have evolved considerably in recent years and we review the progress in this area, highlighting two key limitations. Firstly, that these models only describe prime order groups, despite the popularity of non-prime order groups in practice. We develop new symbolic models to remedy this and use them to discover new attacks on real world protocols. Secondly, that there is no effective procedure for analysing protocols which make use of the full field structure of Diffie-Hellman exponents. We develop a new formalism which can be used to mechanically analyse such protocols and demonstrate its effectiveness on hand-worked examples.

Finally, we conclude by examining the methodology behind our approach. We argue that the techniques behind our new symbolic models can be applied to many other cryptographic primitives to yield more accurate symbolic models. We go on to summarise our contributions and sketch several promising lines of future work.

Acknowledgements

First and foremost, Cas, thank you for your enduring passion and boundless patience over the years. You steered me away from distraction, gave me the space to develop my own ideas and provided much needed perspective on the challenges along the way. I feel incredibly lucky to have worked so closely with you over the past few years and hope there are many collaborations to come.

I owe much to the Information Security Research Group at Oxford: Katriel Cohn-Gordon, Martin Dehnel-Wild, Luke Garratt, Kevin Milner and Nick Moore. From our coffee breaks, to college lunches and formal halls, you hugely enriched my time in Oxford. A special thank you to Daniel Woods, for our many fruitful discussions and shared love of analysis (in all its forms). I also want to thank Ralf Sasse for his enthusiasm, friendliness and sheer depth of knowledge.

I would like to recognise to the many people involved with the Oxford Centre for Doctoral Training in Cyber Security. In particular: David Hobbes, Andrew Martin and Maureen York. Thank you for keeping the ship sailing and making the Oxford process as straightforward as it was. A big thank you also goes to Andrew Simpson who took me in after Operation Auf Wiedersehen began and kept everything running smoothly in Oxford. I also owe much to the teachers at the Lakes School who gave me so many opportunities and challenges.

I want to thank my friends, near and far, for their support and encouragement over the years. Especially: Dan Bell for his company on many an adventure, Jonathan Robinson for always being quick with his wry humour and Sam Bowness for the passion he brings to every ~~argument~~ debate. Thank you to the mathematicians of 43B Brunswick Street, the 'inmates' of Loxley Block and the good people of Warwick Floorball Club for making my undergraduate experience everything it was. A special thank you to Mathilde Ytier without whom this journey would not have begun.

I cannot thank my family enough: Mum and Dad, Harry and Richard (and Tig!). Every journey home or trip away brought me a much needed respite and has given me so many special memories.

Finally, Lauren Durrell, for your endless support, patience and understanding. With all my heart, thank you.

Statement of Originality

All the work presented in this thesis is entirely my own, supervised by Prof. Cas Cremers, with the exceptions of:

- Chapter 2, which describes necessary background information on the symbolic model and TAMARIN. This chapter is drawn from previously published work and the sources are identified in each section.
- Chapter 3, which is the product of a collaboration with Cas Cremers, Katriel Cohn-Gordon and Ralf Sasse. I lead the research on this work and carried out the majority of it, including the identification of the relevant signature properties, the conception and development of the new symbolic models, the selection of the case studies and the implementation of the corresponding TAMARIN models.
- Chapter 4 contains my literature review of previously published symbolic Diffie-Hellman models.
- Section 6.2 of Chapter 6 which discusses the previous work on which the wider chapter is based.

A more detailed statement for each chapter is presented below the chapter heading.

Contents

List of Figures

List of Tables

1 Introduction

Cryptographic protocols are the foundation of all modern communication systems, yet analysing the security of such protocols remains a challenging endeavour. Although many historic protocols were only subject to informal evaluations, a formal security analysis is now considered an essential part of the design process.

A formal analysis is typically carried out by hand and yields a mathematical proof that the protocol under consideration meets a suitable security definition. However, significant expertise is required to develop such proofs and the analysis can take months of researcher time - often leading to protocols being implemented and deployed long before the first manual analysis is published. Furthermore, the security definitions can be difficult to translate into real world properties and their value is open to debate [103]. In some cases, serious flaws in well-studied protocols have been discovered years later [56, 97, 116], despite widely accepted proofs.

Several alternatives to manual analysis have been explored. In the tool-assisted setting traditional pen and paper proofs are translated into formal statements which can be machine-checked by a theorem prover [20, 22, 40]. Although this prevents mistakes in the proof development, the tool assisted approach is still in its infancy and typically requires *more* time and expertise than traditional manual proofs.

Another alternative is automated analysis, where a tool is provided with the protocol description and the intended security properties and asked to derive a proof or counter-example without manual guidance. Two of the most popular such tools are Tamarin and ProVerif which have been used to analyse complex real world protocols such as TLS, Signal and 5G [25, 67, 102].

This approach yields a considerably swifter verdict, allowing the results of the automated analysis to inform the design of a protocol before it is standardised, implemented and deployed. Furthermore, whilst the user must still be familiar with the tool and its input language, the expertise required is significantly lower than that required for manual analysis. However, the automated approach is not without drawbacks. Effective automation requires a more abstract model of cryptographic primitives than is used in manual and tool-assisted analysis. This approximation could lead to missed attacks if it omits additional properties an attacker can exploit in the real world. In addition, not all

protocols are amenable to automated analysis; there are some cryptographic primitives which cannot currently be represented in the symbolic model, meaning protocols which use such primitives cannot be described.

These issues motivate natural research questions. How well do symbolic models of cryptographic primitives match their real world analogues? How can symbolic models be developed or extended to support new classes of primitives and in turn new classes of protocols? In this thesis, we look to tackle these issues.

We first investigate digital signatures. They are an attractive starting point as they have had a stable, widely accepted cryptographic definition and symbolic model for nearly 20 years. Despite this long history, we discover a menagerie of behaviours, which are present and practically exploitable in real world signature schemes but not captured by their symbolic models.

After cataloguing these behaviours, we show how a traditional symbolic model can be extended to capture them. With our new models in hand, we look back at protocols that have previously been verified using the traditional symbolic model. We find new attacks on deployed protocols, which had previously been formally verified in traditional symbolic models. We then turn to repairing these protocols, however, our approach so far has no guarantee of completeness. This motivated us to develop an entirely new style of symbolic model, breaking away from traditional approaches which rely on equational theories. Our new model allowed us to convincingly verify protocols were secure in the presence of these signature properties, in addition to our earlier models which support effective attack finding. This work was published at ACM CCS 2019 and became Chapter 3 of this thesis.

Having tackled digital signatures, we then turn to Diffie-Hellman (DH) Groups. DH groups are a popular primitive, often used for authentication, but come with a number of pitfalls. Unlike digital signatures, the shortcomings of symbolic models of DH groups are already well documented and we discuss the rich history of symbolic DH models in Chapter 4. Notably, contemporary models do not account for group structure, for example the presence of additional subgroups or the use of invalid elliptic curve points. This means they can only faithfully capture protocols which use prime order groups with correctly validated group elements. However, for performance reasons, non-prime order groups are commonly used and often shortcuts are taken when validating group elements.

We then set out to extend the current symbolic model of DH groups in order to capture properties. This work also applies the methodology behind our new style of symbolic model, as again we express properties without altering the underlying equational theory. Our new models faithfully capture the behaviours of non-prime order groups and elliptic curves, including small subgroup elements, invalid curve points and their various mitigations. This leads to the discovery of new attacks on real world protocols, notably on Secure Scuttlebutt where the attack can only be found by combining both our new

signatures model and our new DH model. Wary that sometimes theoretical attacks do not translate into real world behaviour we implemented our attack and worked with the Scuttlebutt developers to mitigate it. This work was published at IEEE CSF 2019, where it won one of three 'distinguished paper' awards and forms the core of Chapter 5.

Another issue with contemporary DH groups, also discussed in Chapter 4, is that they only describe the multiplicative structure of group exponents. They do not capture the additive structure and consequently cannot represent protocols, such as Menezes–Qu–Vanstone (MQV) key exchange and Password Authenticated Key Exchange by Juggling (JPAKE), which require this additive structure to function.

We begin by considering what makes this problem particularly intractable. It transpires there are strong mathematical results that imply no solution can be found using traditional symbolic approaches through equational theories. Instead, we look back at previous work in this area in which we find a promising line of inquiry which has developed a novel proof methodology. We explore this result and go on to extend this approach in Chapter 6. We show how it could be integrated in a modern automated verification tool as a set of constraint solving rules. Our work in this area is still in progress and we conclude with hand worked examples and a program of future work.

Finally, in Chapter 7, we explore the methodology behind our new style of symbolic proofs, enumerate our contributions and identify several promising lines of future research.

1.0.1 Overview

Chapter 2 We introduce some general background material on cryptographic models, term rewriting and the Tamarin Prover.

Chapter 3 Our first substantial contribution in which we discover behaviour missed by the traditional model of digital signatures, provide new models fixing the omission and discover a number of new attacks on real world protocols.

Chapter 4 This Chapter introduces some general mathematical background on DH groups and the evolution of their symbolic models. We summarise the current state of the art in modern automated analysis of protocols using DH groups.

Chapter 5 Our second substantial contribution in which we introduce new symbolic models for DH groups suitable for capturing non-prime order groups and elliptic curves. We provide a number of case studies in which we find new attacks on real world protocols.

Chapter 6 Our third substantial contribution in which we explore how to extend the symbolic model of DH groups to capture the full field structure in the exponent. We recap previous work in this area with a particular focus on intractability results and a promising line of recent inquiry. We then present a new set of constraint solving rules and show how they can be applied in a hand worked case study.

Chapter 7 We conclude by drawing some high level conclusions, enumerate our contributions and identify several promising lines of future work.

2 Background

The following background content is drawn directly from similar theses and follows their presentation closely: Dehnel-Wild [69], Milner [130], Schmidt [142] and Meier [121]. Some additional content on equational theories and unification is drawn from [17].

2.1 Cryptographic Models

Any formal security analysis of a protocol must specify: the claimed properties, the assumptions under which the claims hold and how the claims can be formally proven from those assumptions. The assumptions, explicitly or implicitly, describe the capabilities of the attacker and the behaviour of the underlying cryptographic primitives. We refer to commonly used set of properties and assumptions as a model and often compare the relative merits of different models.

The computational model, commonly used in manual proofs, represents messages as bitstrings and cryptographic primitives as functions from bitstrings to bitstrings. The adversary is any probabilistic Turing machine. Cryptographic primitives are described by assumptions that the adversary cannot win a certain game with non-negligible probability (e.g. computing discrete logs) and then any attack on the protocol is reduced to the adversary winning this game, thus breaking the primitive. In particular, the computational model considers properties of primitives which are not explicitly stated, or even known in advance, provided they do not break security game.

Although the computational model is relatively strong as it makes few assumptions, it can be challenging to define security properties precisely in it, particularly with regard to dynamic corruption and session agreement. Furthermore, it is prone to human error as proofs in the computational model often require many case distinctions to be considered and it is rare for every case to be explicitly covered. There have been many instances of protocols "proven" secure in the cryptographic model, only to be later shown to exhibit serious vulnerabilities, for example [97], [104], [105] and [123] are only a fraction of the papers concerning flaws in MQV, fixes to the protocol and further problems found with the fixes.

In the symbolic model, sometimes called the Dolev-Yao model, cryptographic primitives are instead represented by function symbols which obey a set of universally quantified identities and the messages are terms composed of these symbols. This is is sometimes referred to as "perfect cryptography" assumption as we have to specify the identities our function symbols satisfy in advance and we assume no other identities hold. This entails considerably stronger assumptions than the computational model, but the trade off is balanced by two advantages. Firstly, the proof structure is amenable to automated reasoning, so proofs can be found and verified by machine, eliminating[1] the risk of a mistake in the proof. Secondly, the perfect cryptography assumption is arguably similar to the random oracle assumption, which is widely used in computational models, despite the fact that true random oracles are known to be impossible. Recent research has investigated how to weaken these assumptions, whilst retaining the amenability to automated proofs,

[1] Or at the very least seriously reducing the risk. A human is still required to specify the protocol in the first place, and the prover must be implemented correctly.

by deriving soundness results that show a proof in the symbolic model implies a proof exists in the computational model.[2] However, this line of work is still in its infancy.

There are a number of tools [39, 73, 78, 143], which can ingest protocol descriptions and security properties, then automatically derive proofs of security or attacks upon the protocol in the symbolic model. These tools could be compared on a number of different aspects, for example: the overall proof strategy, representations of interleavings, the protocol specification language, the property specification language, the equational theories the tool supports or whether the tool supports bounded or unbounded verification[3].

In general, protocol verification in the unbounded setting is known to be undecidable [160], and even in the bounded setting additional assumptions are often required to induce decidability. Due to the undecidable nature of the problem, unbounded tools are proven to be sound and complete, but cannot guarantee termination. These tools search for a trace demonstrating a security property violation, or proof that no violation occurs within the tool's framework. Each tool is required to model the behaviour of cryptographic functions, such as digital signatures, Diffie-Hellman groups or symmetric encryption. As this thesis is predominately founded on the symbolic model, we will discuss the necessary mathematical background next.

2.2 Mathematical Background

In this section we introduce the relevant mathematical background which is the foundation of the symbolic model. In particular, we define term algebras, equational theories and explain some of their applications.

2.2.1 Term Algebras

In a term algebra, some functions are only defined over restricted domains. We use a *sort* system to enforce these restrictions. Each sort is a unique label for a particular domain. For example, we might take the universe of the integers with sorts corresponding to negative and positive values.

Definition 1. An **order-sorted signature** $\mathbf{\Sigma} := (S, \leq, \Sigma)$ 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:

1. $\forall s \in S$ the connected component C of $s \in (S, \leq)$ has a top sort denoted $top(s)$ such that $c \leq top(s)\ \forall c \in C$.
2. $\forall f : s_1 \times \ldots \times s_k \to s \in \Sigma$ with $k \geq 1$ then $f : \top s_1 \times \ldots \times \top s_k \to \top s$.

[2] For a further discussion of the symbolic and computational models, see [41].

[3] In bounded verification, there is a fixed upper limit on the number of protocol sessions that are considered.

If there is only one sort in S, we say that $\mathbf{\Sigma}$ is unsorted and we identify it with Σ. We write Σ^k for the set of all k-ary function symbols in Σ e.g. Σ^0 is the set of constant functions.

Definition 2. For each sort $s \in S$, we assume there are pairwise disjoint, countably infinite sets of **variables** $\mathcal{V}_s$ and **constants** $\mathcal{C}_s$. We define the set of all variables as $\mathcal{V} := \cup_{s \in S} \mathcal{V}_s$ and all constants as $\mathcal{C} := \cup_{s \in s} \mathcal{C}_s$. We use $x : s$ to denote a variable from $\mathcal{V}_s$.

Definition 3. Let $\mathbf{\Sigma} := (S, \leq, \Sigma)$ be an order sorted signature and $X \subseteq \mathcal{V} \cup \mathcal{C}$ a set of variables and constants such that X and Σ are disjoint. Then the set $\mathcal{T}(\Sigma, X)$ or the **term algebra** of all $\mathbf{\Sigma}$-terms over X is inductively defined as

1. $X \subseteq \mathcal{T}(\Sigma, X)$
2. $\forall n \geq 0$, $\forall f \in \Sigma^{(n)}$ and $\forall t_1, \ldots, t_n \in \mathcal{T}(\Sigma, X)$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, X)$

Definition 4. We call an element of $\mathcal{T}(\Sigma, X)$ a **term**. A **position** p in a term t is a finite sequence of integers, corresponding to the path taken to reach some subterm when viewing the term as a tree. The empty sequence being denoted by [] and we write $t|_p$ for the **subterm** of t at p. If $p = []$ then $t|_p = t$. If s is a subterm of t we write $s \sqsubseteq p$. We write the set of subterms of t as $\mathit{subterms}(t)$. We define $\mathit{Vars}(t)$ to be $\mathit{subterms}(t) \cap \mathcal{V}$. We say a term is **ground** if $\mathit{Vars}(t) = \emptyset$. For terms t and u, with position p, we write $t[u]_p$ for the term t where $t|_p$ is replaced by u.

Definition 5. We call the functions $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, X)$ **substitutions** and define the set of such functions to be $\mathcal{S}$. We write a substitution of the variable x with the term y as $\{x \to y\}$. We denote the combination of substitutions as $\sigma \cup \sigma'$ which is defined as long as the σ and σ' agree on their intersection. We write $\sigma(t)$ for the application of a substitution to a term, omitting the brackets where no confusion can occur. We overload notation and allow one substitution to be applied to another in the natural way.

2.2.2 Equational Theories

Those interested in a full treatment of this area, should see Chapters 3,4 and 10 of [17] and the introductory chapters of [118].

Definition 6. An **equation** over a signature $\mathbf{\Sigma}$ is an unordered pair $\{s, t\}$ of terms $t, s \in \mathcal{T}(\mathbf{\Sigma}, \mathcal{V})$ which we write $s \approx t$. Given a signature $\mathbf{\Sigma}$ and a set of equations E, the **equational theory** $\approx_E$ is defined to be the smallest congruence on $\mathbf{\Sigma}$ containing all instances of equations of E.

A natural question is given a pair of terms and an equational theory, how can we determine if the terms are equivalent under the equational theory? For that, we turn to term rewriting.

2.2.3 Term Rewriting

Definition 7. A **rewrite rule** over a signature $\mathbf{\Sigma}$ is an ordered pair of terms (l, r) with $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$ which we denote $l \to r$. A **rewrite system** $\mathcal{R}$ is a set of rewrite rules. From $\mathcal{R}$ we define the **rewrite relation** $\to_{\mathcal{R}}$ with $s \to_{\mathcal{R}} t$ if there is a position p in s, a rewrite rule $l \to r \in \mathcal{R}$ and a substitution σ such that $s|_p = \sigma(l)$ and $s[\sigma(r)]_p = t$. We denote by $\to_{\mathcal{R}} *$ the transitive closure of $\to_{\mathcal{R}}$.

Definition 8. A rewrite system $\mathcal{R}$ is **terminating** if there is no infinite sequence $(t_i)_{i \in \mathbb{N}}$ of terms with $t_i \to_{\mathcal{R}} t_{i+1}$.

Definition 9. A rewrite system $\mathcal{R}$ is **confluent** if $\forall t, s_1, s_2 \in \mathcal{T}(\mathbf{\Sigma}, \mathcal{V})$ with $t \to^*_{\mathcal{R}} s_1$ and $t \to^*_{\mathcal{R}} s_2$ implies there exists $t' \in \mathcal{T}(\mathbf{\Sigma}, \mathcal{V})$ such that $s_1 \to^*_{\mathcal{R}} t'$ and $s_2 \to^*_{\mathcal{R}} t'$.

Definition 10. A rewrite system is **convergent** if it is both terminating and confluent. In this case, each term t has a **unique normal form** which we write $t \downarrow_{\mathcal{R}}$ i.e. it is the unique term t' such that $t \to^*_{\mathcal{R}} t'$ and there is no term t'' such that $t' \to^*_{\mathcal{R}} t''$.

Give an equational theory, we can orient it into a set of rewriting rules by assigning a direction to each equation. If the resulting system is convergent, we have an effective means of deciding equality between terms. We simply apply the rewriting system to each side and check if the unique normal forms coincide.

However, it may fail to be confluent or terminating. If it is not confluent, this is typically straightforward to resolve, and a set of additional rewrite rules can be derived which restore confluence without changing induced equational theory[4]. However, if the resulting rewrite system is not terminating, this is typically due to a non-orientable equation. For example:

$$f(x, y) \approx f(y, x)$$

This equation expresses the commutativity of f, but it cannot be orientated. We resolve this with the introduction of rewriting modulo an equational theory.

Definition 11. Given a convergent rewrite system $\mathcal{R}$ and an equational theory A, we define rewriting $\mathcal{R}$ modulo A to to be the relation $\to_{\mathcal{R},A}$ where $s \to_{\mathcal{R},A} t$ if there is a position p in s, a rewriting rule $l \to r \in \mathcal{R}$ and a substitution σ such that $s|_p \approx_A \sigma(l)$ and $s[\sigma(r)]_p = t$.

For some equational theories, notably ones expressing the associativity and commutativity of an operator, the resulting procedure for determining equality is decidable.

[4] See Section 6.2 of[17] for more

2.2.4 Unification

Definition 12. For an equational theory E, an E-unifier of two terms s and t is a substitution σ such that $\sigma(s) \approx_E \sigma(t)$. We term the set of such E-unifiers to be $\mathcal{U}(s,t)$.

Definition 13. A substitution σ is **more general** than a substitution σ' if there is a substitution δ such that $\sigma' = \delta\sigma$. We write $\sigma \lesssim \sigma'$ and say that σ' is an instance of σ. This relation is a quasi order (reflexive and transitive).

Definition 14. A substitution σ is a Most General Unifier (MGU) of s, t if σ is a least element of $\mathcal{U}(s,t)$ under $\lesssim$.

Definition 15. We use $\texttt{unify}_E(s,t)$ to denote an E-unification algorithm that returns a set of most general unifiers for s and t. For a fixed equational theory, we say the algorithm is **unitary** if the resulting set has cardinality at most one, **finitary** if the cardinality is always finite. For some equational theories, the algorithm may be **infinitary** where a set of most general unifiers has non-finite cardinality. Finally, there may not be a set of most general unifiers in which case we call it **nullary**.

2.3 The Tamarin Prover

The Tamarin Prover is a state of the art tool for the automated analysis of protocols in the symbolic model. In Chapters 3 and 5, we make extensive use of Tamarin and consequently develop the background of the Tamarin Prover here.

At a high level, Tamarin uses a term algebra with an equational theory to model cryptographic primitives and their properties. The execution of a protocol in an environment with an adversary is then represented as a labeled transition system. The state consists of messages on the network, the adversary knowledge, and the internal states of the protocol participants. Protocol and adversary interact by exchanging messages on the (adversary-controlled) network. Both protocol rules and adversary capabilities are specified as labeled multiset rewrite rules. These are used to define a transition system that specifies a set of traces, which model all possible sequences of events. The security requirements are then specified in a (guarded) fragment of first-order logic, expressed over a trace.

We now discuss the relevant aspects of Tamarin in greater detail. For further information on Tamarin see [143] and the Tamarin prover manual [156].

2.3.1 Equational Theory

As mentioned, Tamarin represents messages as terms and cryptographic primitives as function symbols on that term algebra. The term algebra has a topsort M and two disjoint subsorts *Fresh* and *Public*. These subsorts are syntactic sugar for defining variables which

are atomic and correspond to some constant. All equations in TAMARIN are specified in terms of the top sort M.

The properties of each cryptographic primitive are expressed as an equational theory. A number of equational theories for cryptographic primitives are built in to TAMARIN, but it also supports user defined equational theories, subject to certain restrictions. We refer to the collective equational theory employed by the user as EQT_{Usr}.

```
functions: senc/2, sdec/2
equations: sdec(senc(m,k),k) = m
```

Equational Theory for Symmetric Encryption

We declare the function symbols after the keyword `functions`, with their arity after a `/` and the equations after the keyword `equations`, where all non-declared symbols are interpreted as variables. Here the single equation expresses that decrypting a ciphertext with the correct key produces the original plaintext.

These user specified equational theories must satisfy a number of properties. In particular, they must be terminating and confluent and also enjoy the finite variant property. We introduced the first two properties in the preceding section. We will not discuss the finite variant property in detail, but in short it ensures TAMARIN can use a general and effective procedure to perform unification. Note that TAMARIN does not check these properties hold automatically and care must be taken, as using invalid equational theories will lead to non termination or incorrect results.

2.3.2 Labelled Transition System

TAMARIN uses labelled multiset rewriting to model security protocols. We first define how the transition system functions, then explain its semantic meaning in the next section.

Definition 16. Let there be two countably infinite sets FN and PN of **fresh and public names** respectively. Let $\mathbf{\Sigma}$ be an order sorted signature then we refer to our **terms** $\mathcal{T}$ as the term algebra $\mathcal{T}(\mathbf{\Sigma}, FN \cup PN \cup \mathcal{V})$.

The state of the transition system is given by a multiset of *facts*. Facts are special symbols that take any (fixed) number of terms as their arguments.

Definition 17. A **fact** is a tag drawn from a finite signature Σ_F and finite ordered tuple of terms from $\mathcal{T}$. Fact tags contain a name, an arity and a multiplicity. The multiplicity of a fact may be **linear** or **persistent**. A linear fact is consumed by a system transition, whereas a persistent fact is not. We generally write $\mathsf{Factname}(t_1, t_2, t_3)$ for a fact named `Factname` with three terms as its arguments. A persistent fact is distinguished by its name beginning with !.

Transitions between system states are given by labelled multiset rewriting rules.

Definition 18. A **labeled multiset rewriting rule** is defined by three ordered tuples of facts: its premises (or left hand side), its conclusions (or right hand side) and finally its actions (or labels). We give each rule a name, which does otherwise affect the transition system. We represent such a rule as:

$$rulename : [\ l\ ] \mathrel{-\!\![} \ a\ \mathrel{]\!\!\to} [\ r\ ]$$

```
rule name:
[ l ]--[ a ]->[ r ]
```

where `rule` is a keyword, `name` is an identifier, and `l`, `a`, `r` are multisets of facts representing the *premises*, *actions*, and *conclusions* respectively. All of these may contain variables. Some rules may have no associated action, and in that case we omit the action `[ a ]` in the rule description.

Definition 19. A *labeled transition system* is a set of labelled multiset rewriting rules and operates on ground terms, i.e., terms without variables. A rule is *applicable* in a given ground state when an instantiation of the premises of the rule is a subset of the current state. Applying a rule changes the state by removing the premises, and adding appropriately instantiated conclusions. The instantiated action facts are the action associated with this rule instance.

An *execution* is a sequence of states starting from the empty set and using *rule instances* to transition from one state to the next. Then, a *trace* is the sequence of ground action facts appearing at the rule instances in a protocol execution. The *timepoint* of a rule instance is simply its position in the trace.

2.3.3 Semantics

We use *PN* to refer to public constants known by all parties. We use *FN* to refer to freshly generated, random variables, known at their moment of creation only to one party. Linear facts model mutable state, whilst persistent facts model read only memory or statements about knowledge. A protocol is described by a set of labelled multiset rewriting rules.

We give certain facts special meaning:

- $\mathsf{Fr}(x)$ models the generation of a fresh name.
- $\mathsf{In}(x)$ models receiving a message.
- $\mathsf{Out}(x)$ models transmitting a message.
- $K(x)$ models the adversary learning a term.

Tamarin models a network in which all network traffic is controlled by the adversary. Consequently, the adversary learns the contents of all $\mathsf{Out}(x)$ facts and is the only party who can create $\mathsf{In}(x)$ facts. This is supported by a set of message deduction rules, special

rules which determine how the adversary can learn information from a message and create new messages.

We will not discuss the details of message deduction rules in this thesis, except to highlight that they can be partitioned into construction and deconstruction rules. Construction rules are for the construction of new terms from terms the adversary knows, including the generation of new fresh values:

$$AFresh : [\ \mathsf{Fr}(x)\ ]\!\!\rightarrowtail\!\!\!\!\!\!\rightarrowtail [\ Out(x)\ ]$$

Deconstruction rules are for deconstructing terms learned from the protocol, for example the decryption of a received ciphertext if the adversary knows the key. For both construction and deconstruction rules Tamarin can automatically generate the required rules from the protocol description and equational theory.

2.3.4 Security Properties

Security properties are logical statements about a particular trace. Given a security property, Tamarin automatically establishes whether there is a valid trace in the transition system which is a counterexample to the property, which it then returns to the user. Otherwise, Tamarin provides a proof that no such trace is possible.

Security properties are defined in a fragment of two-sorted first-order logic, with messages and timepoints, and quantification is possible over both. The atoms considered are then $\bot$, term (in-)equality `s = s'`, timepoint ordering $t_1 < t_2$, timepoint equality $t_1 = t_2$, or an action *Fact* at a timepoint t_1 written $Fact(terms)@t_1$, together with the usual first order logic connectives.

2.3.5 Restrictions

We additionally consider *restrictions* which limit the explored state space. Technically, restrictions are formulas just like security properties, and their semantics ensure that all traces must satisfy the restrictions. If a trace violates any restriction, it is immediately discarded. Restrictions are often used to model conditional rewriting rules, for example, to model inequality checks and that an action happens only once. More complex uses can enforce a passive (possibly message reordering) adversary by ensuring each received message by a protocol participant was previously sent by a different participant, and is only received one time.

2.4 Internals of the Tamarin Prover

2.4.1 Overview

In Chapter 6, we go beyond interacting with Tamarin as a 'black box' and instead must consider some details of how it operates. t a high level, Tamarin operates by transforming a security property into a satisfaction problem - given a set of constraints, is there a trace satisfying those constraints which can be reached by the labelled transition system? In this section, we discuss how these constraints are represented, how equational theories are integrated and how satisfaction (or its absence) is proven.

2.4.2 Executions

As previously mentioned, the state of the transition system is modelled as a finite mulitset of facts, which is modified step by step by executing rules. The transition system starts with an empty state.

Definition 20. A valid step of a transition system R with respect to an equational theory E is defined by the **transition relation** $\Longrightarrow_{R,E}$:

$$\frac{[\ l\ ]\!\!\rightarrowtail\!\![\ a\ ]\!\!\rightarrowtail\!\![\ r\ ] \in_E ginsts(R), lfacts(l) \subseteq^{\#} S, pfacts(l) \subseteq set(S)}{S \overset{set(a)}{\Longrightarrow}_{R,E} ((S \setminus^{\#} lfacts(l)) \cup^{\#} mset(r))}$$

where $lfacts(l)$ and $pfacts(l)$ respectively denote the linear and persistent facts in l. We use $\cdot^{\#}$ to represent the multiset extension of a set operator $\cdot$. We use $r \in_E R$ to denote that r is an element of R modulo the equational theory E. $mset$ treats the collection as a multiset.

With this notion we can now formally define the traces of a system:

Definition 21. The **traces** generated by the multiset rewriting system R under an equational theory E are defined as the sequence of action facts:

$$traces_E(R) := \{\langle A_1, \ldots, A_n\rangle | \exists S_1, \ldots, S_n \text{ such that } \emptyset \overset{A_1}{\Longrightarrow}_{R,E} S_1 \overset{A_2}{\Longrightarrow}_{R,E} \ldots \overset{A_n}{\Longrightarrow}_{R,E} S_n \\ \wedge \forall i \neq j. \forall x. (S_{i+1} \setminus^{\#} S_i) = \{\mathsf{Fr}(x)\}^{\#} \implies \\ (S_{j+1} \setminus^{\#} S_j) \neq \{\mathsf{Fr}(x)\}^{\#}\}$$

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

Notice that the sequence of rule instances is sufficient to determine the trace, as each global state can be reconstructed from it.

2.4.3 Dependency Graphs

TAMARIN makes use of dependency graphs to represent protocol executions. Each node in the dependency graph represents a particular rule instance in the trace. Edges are drawn between nodes to represent particular dependencies. For example, a premise (fact on the left hand side of a transition rule) will be connected to the conclusion (fact on the right hand side of a transition rule) of another rule and this induces a causal ordering between them.

Definition 22. 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 (under the equational theory). The edges $D \subseteq \mathbb{N}^2 \times \mathbb{N}^2$ and dg satisfies the conditions 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 is the index of a premise or conclusion fact.

1. $\forall((i, u), (j, v)) \in D$, $i \leq j$ and the conclusion fact (i, u) is equal under E to the premise fact (j, v).
2. Every premise of dg has exactly one incoming edge.
3. Every conclusion of dg with a linear conclusion fact has at most one outgoing edge.
4. Instances of $\mathsf{Fr}(x)$ in I are unique.

Definition 23. The **trace of a dependency graph** (I, D) where $I := \langle r_0, \ldots, r_n \rangle$ is defined as the sequence of sets $\langle set(acts(r_0)), \ldots, set(acts(r_n)) \rangle$. We write $Trace(dg)$

Definition 24. The **set of all dependency graphs** for R, E is denoted $dgraphs_E(R)$.

We now note that these dependency graphs are an alternative formulation of the original multiset rewriting semantics:

Theorem 3 of [122]. For every multiset rewriting system R and every equational theory E:

$$traces_E(R) =_E \{ Trace(dg) | dg \in dgraphs_E(R) \}$$

2.4.4 Constraint Systems

TAMARIN converts a security property into an initial constraint system, we will not detail this transformation here as it is not relevant for our discussion. However, we will discuss how this constraint system is described and solved.

Definition 25. A **constraint** is either:

- A trace formula
- A node constraint, written $i : r$ for an index i and a rule instance r.
- A premise constraint, written $f \triangleright_v i$ for a fact f occurring as the vth premise at index i.

- An edge constraint, written $(i, u) \rightarrowtail (j, v)$ for an edge between the uth conclusion at index i and the vth premise at index j.

These constraints are used to represent restrictions that must be met by a dependency graph to be a satisfying trace for the constraint system. A **constraint system** is a finite set of constraints. We denote the domain of constraints $\mathcal{C}$.

TAMARIN solves a constraint system using a constraint-reduction relation $\rightsquigarrow_{R,E}$. This relation refines the initial constraint system into either a solved constraint system containing a satisfying trace or a set of constraint systems which each contain a contradiction. This procedure constitutes a form of backwards search, TAMARIN begins with the events that must have happened in order to violate a property and works backwards to establish if such events are possible.

The constraint reduction relation is made up of a set of rules. *Case distinction* rules add additional constraints by splitting a constraint system into a set of constraint systems, each expressing a possible case. *Simplification* rules are rules which do not introduce case distinction. *Contradiction* rules can remove constraint systems entirely when there are contradictory constraints in the system. Each constraint reduction rule in the relation must be proven to be both *complete* and *sound*. That is, if the constraint system acted on by a rule has a solution, the solution is preserved by the constraint reduction rule. Similarly, if the constraint system has a solution after the rule is applied, then it also the solution prior to application.

We will not detail these rules in full here. However, we will discuss one particular constraint reduction rule which handles equality constraints. Equality constraints specify that two terms must be equal in any solved constraint system and are written $Eq(s, t)$.

Definition 26. The reduction rule $S_{\approx}$ is defined to be: where Γ is the original constraint

$$\frac{Eq(s,t), \Gamma}{\sigma_1\Gamma | \ldots | \sigma_n\Gamma} \, S_{\approx}$$

Figure 2.1: The Rule $S_{\approx}$

system. and $\{\sigma_1, \ldots, \sigma_n\} = \mathtt{unify}_E(s, t)$. Here $|$ separates distinct constraint systems and $\cup$ combines constraint systems. We apply a substitution to a constraint system in the natural fashion. Any variables introduced in the conclusion of constraint rule are implicitly existentially quantified unless defined otherwise.

This rule is crucial for TAMARIN as it solves the constraint system. It solves an equality constraint by using unification to introduce a case for every valid most general unifier. TAMARIN also applies a number of optimisations that allow it to reduce the computational overhead of this approach which are not relevant for our work.

3

Refined Models of Digital Signatures

The chapter is based on the paper:

Dennis Jackson et al. 'Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures'. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. 2019, pp. 2165–2180. DOI: 10.1145/3319535.3339813. URL: https://doi.org/10.1145/3319535.3339813

This paper was joint work with Cas Cremers, Katriel Cohn-Gordon and Ralf Sasse. I lead the research on this paper and the substantial contributions in this chapter are my own. My co-authors principally contributed to the initial conception of the work and the final write up of the paper.

The associated materials for this chapter, including sources for all of the models and case studies, are available at [92].

3.1 Introduction

In this chapter we explore the symbolic model of digital signatures and compare it with the corresponding computational definitions. We discover and describe an important mismatch between the symbolic and computational definitions. Using a series of examples based around the Station to Station protocol, we will see how this mismatch has important security implications. Consequently, we go on to develop a new family of symbolic models for digital signatures which support both efficient attack finding and protocol verification. Finally, we will show the effectiveness and efficiency of our models with a suite of new case studies, including the discovery of new attacks on DRKey, a 'next-generation Internet' routing protocol and WS-Security, a web standard for request-response messages.

3.2 Digital Signatures

3.2.1 Computational Definitions

We begin with the classical definition of signature schemes which we state informally. (The formal definitions can be found in [86, 99].)

Definition 27. A *digital signature scheme* is composed of three polynomial time algorithms:

1. KGen, a probabilistic algorithm, takes in the security parameter and produces a pair (sk, vk).
2. Sig, a probabilistic algorithm, takes in a private key sk and a message m and produces a signature s.
3. Vf, a deterministic algorithm, takes in a verification key vk, message m and signature s and outputs success or failure.

It is *correct* if $\mathsf{Vf}(vk, m, \mathsf{Sig}(m, sk))$ succeeds with high probability for all messages m and any (sk, vk) output by KGen.

The essential security definition that nearly all signature schemes are expected to meet is existential unforgeability against an adaptive chosen message attack:

Definition 28. A signature scheme is *existentially unforgeable under an adaptive chosen message attack* or *Existential Unforgeability under an Adaptive Chosen Message Attack (EUF-CMA)*-secure if no PPT adversary has a non-negligible advantage in this experiment:

1. The challenger generates a keypair and gives the public key to the adversary.
2. The adversary may adaptively query a signing oracle polynomially often which returns a signature on the chosen message.
3. The adversary wins if they can output a message and signature pair which is verified and the message was not previously given as input to the signing oracle.

This security definition captures *forgery resistance*: even an adversary that can adaptively query for signatures on messages of their choice cannot forge a signature for a different message.These definitions first appeared in 1988 and are widely accepted as the de facto standard.

3.2.2 Existing Symbolic Models

As might be expected, the symbolic model of digital signatures has remained essentially static since the origin of the field. The two most widely used models are:

```
functions: verify/3, sign/2, pk/1, true/0
equations: verify(sign(m, sk), m, pk(sk)) = true
```

Standard Signature Model used by Tamarin and ProVerif

```
functions: rvlSign/2, rvlVerify/3, getMsg/1, pk/1, true/0
equations: rvlVerify(rvlSign(m,sk),m,pk(sk)) = true
getMsg(rvlSign(m,sk)) = m
```

Signature Model with Message Recovery used by Tamarin and ProVerif

The equation in the standard model allows the protocol and adversary to verify signatures by applying `verify` to a claimed signature, alongside the expected message and public key, and test if the resulting term is equal to `true`. In these models of signatures, a public key is considered to be a function of a secret key, rather than the alternative notation where both are functions of a seed value.

These models have been widely used over the past 20 years, see [156, Page 37] for Tamarin's version, and [42, Page 14] for ProVerif's. Indeed the message recovery model appears verbatim in [39], the original ProVerif paper. Tools such as CPSA [73] and Maude-NPA [78] use similar models.

The ProVerif manual [42] also proposes an alternative signature model that is non-deterministic and has message and key recovery:

```
functions: spk/1, sign/3, getmess/1, checksign/2,getkey/1
equations: checksign(sign(m,k,r),spk(k)) = true
getmess(sign(m,k,r)) = m
getkey(sign(m,k,r))  = spk(k)
```

ProVerif's Probabilistic Model with Message and Key Recovery (Translated into Tamarin notation)

This removes the bijection between signatures and the messages they correspond to, allowing for more behaviour to be expressed. In this model it is possible to extract the message from a signature using the second equation, and to extract the public key from a signature using the third equation. We were unable to find a publication actually using it in practice.

3.3 Additional Signature Behaviours

In this section we describe four properties of signature schemes that can lead to real-world attacks: *key substitution*, *malleability*, *re-signing* and *colliding signatures*. We relate each property to the existing computational and symbolic definitions, then develop new symbolic models that can capture those behaviours and attacks not already considered. We stress that all of these behaviours are permitted by the standard definition of signature security (EUF-CMA) and are not the result of implementation mistakes. For each symbolic model we develop, we demonstrate an attack on an example protocol missed by the traditional model.

3.3.1 Running example: STS-MAC

We will use Station-to-Station (STS) [72] as a running example, as it is a simple and well-understood authenticated key exchange protocol. It has the additional benefit of a long history of being exploited using surprising signature properties like those we consider, see [38]. In STS (specifically the STS-MAC variant), two parties exchange DH ephemeral keys with a signature and a Message Authentication Code (MAC) in order to derive a shared symmetric key, as shown in Figure 3.1.

We write $x \leftarrow\!\!\$\, \mathbb{Z}_p$ to denote drawing a random number. $\mathsf{Sig}_{\mathsf{sk}_a}(t)$ means signing the term t with the private key sk_a associated with the public key pk_a. We assume the parties can authenticate each other's public key, for example through a certificate signed by a trusted third party, which we denote cert_a. We assume that before signing the public key, the trusted third party verifies ownership of a corresponding private key.

In the usual symbolic notation signing becomes `sign(t,ska)` for a private key `ska` (representing some sk_a) for which the verification key is `pk(ska)` (representing the corresponding pk_a). Similarly, by $\mathsf{MAC}_{g^{xy}}(\sigma_a)$ we mean the MAC created with key g^{xy} for term σ_a. The protocol's intended result is that both parties share the key K.

After a key agreement protocol like this completes, we expect some security properties to hold about the resulting key. In our verification in the following sections, we will consider the properties: *key secrecy*, *identity agreement*, and *strong session agreement*. We interpret key secrecy to mean that when two honest parties finish a protocol run with each other, the resulting key is secret from the adversary. Identity agreement means that, whenever two honest parties agree on a key, they also agree on each other's identity. Finally, strong session agreement means that if two honest parties finish a protocol run with each other, they agree on the transcript of that session. That means that every message sent by A (respectively B) was received by B (respectively A) and every message that was accepted by A (respectively B), was transmitted by B (respectively A) without alteration.

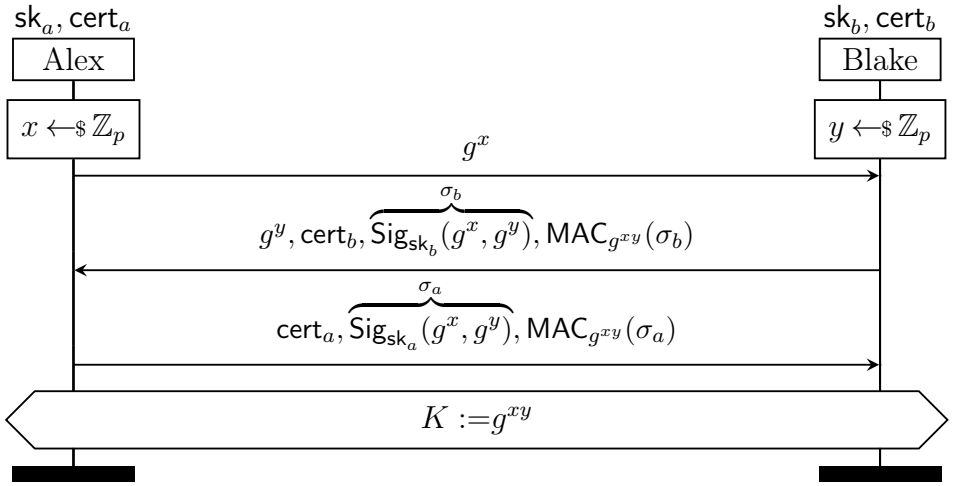


Figure 3.1: Sketch of the original STS-MAC protocol [72]

3.3.2 Key Substitution

In a *key substitution attack*, the adversary is given an existing signature, message and public key, and is able to construct a new public key (and possibly a new message as well) such that the honest signature will verify under the new public key (and new message).

This area of research has had at least three terminologies: Blake-Wilson and Menezes [38] called such attacks Duplicate Signature Key Selection (DSKS); Menezes and Smart [125] termed them Key Substitution attacks; and most recently Pornin and Stern [138] presented it as exclusive ownership. We follow this latest terminology.

3.3.2.1 CEO

The following property was first noted in [125], but we draw our definition from the later work [138].

Definition 29. [138, Def 1] A signature scheme fails to provide *Conservative Exclusive Ownership (CEO)* if there is an efficient algorithm $\texttt{fake}(pk, (sig, m)_i)$ that given a public key and a sequence of message and signature pairs under that key, outputs a key pair (pk', sk') such that $pk \neq pk'$ and $\mathsf{Vf}(pk', m_j, sig_j) = \mathsf{true}$ for some j.

Some signature schemes, including Elliptic Curve Digital Signature Algorithm (ECDSA) (where the signatures have a fixed generator) have been proven to satisfy Conservative Exclusive Ownership (CEO) [125]. Other schemes, such as ECDSA (with signature specified generators) and RSA-PSS, do not satisfy CEO and a `fake` algorithm can be constructed for them [38]. Traditional symbolic models of signatures implicitly assume that CEO holds, because they do not include such a `fake` algorithm: each signature in the traditional model can only be verified by the (unique) public key that corresponds to the secret key used for signing.

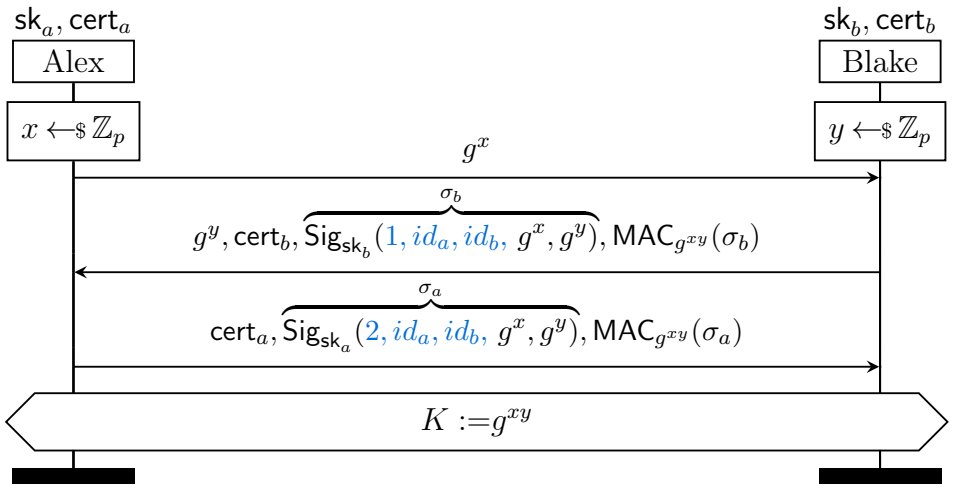


Figure 3.2: Sketch of STS-ID [38], which includes the identities of the communicating parties in the MAC.

To model this additional behaviour, we introduce a new abstract function `CEOgen` that models the existence of such an algorithm as formalised within the CEO definition. This function takes as argument a signature, and returns a private key x. We then add an equational theory that expresses that if x is output by the `CEOgen` function, then the corresponding public key $pk(x)$ can also be used to successfully verify the corresponding signature. The CEO inequality $pk \neq pk'$ is given by construction, because the terms representing the two public keys are necessarily distinct.

```
functions: CEOgen/1
equations: verify(sign(m, sk), m, pk(CEOgen(sign(m, sk)))) = true
```

No-CEO: Model for signature schemes that do not satisfy CEO.

If we add this equational theory, called no-CEO, to Tamarin and rerun our analysis of STS-MAC, Tamarin immediately discovers a Unknown Key Share (UKS) attack, violating identity agreement. In the attack, two parties A and B establish the same session key, but have different assumptions on whom they are talking to: A believes they share the key with B, but B believes they share the same key with a corrupted agent E. Thus, if A later receives a message from B encrypted with the shared key, they will incorrectly assume it was intended for them, although B believed they were sending it to E.

This attack was first reported by Blake-Wilson and Menezes [38] where they described it as a DSKS attack. It has also been referred to as "Strong Key Substitution" by [44, 125]. The adversary waits until A sends the final message to B, then using the no-CEO property produces a new public key, registering it with the Certificate Authority (CA), for which A's signature will verify. Note that the adversary cannot just replace the signature directly, since it is protected by the MAC, to which the adversary does not know the key. The adversary then replaces the associated certificate from A with their own. Consequently, B concludes they are talking to the adversary, even though the key is shared with A and in fact secret from the adversary.

Blake-Wilson and Menezes [38] also suggested the fix of including the identities under the signature, depicted as STS-ID in Figure 3.2. If we model this patched protocol in Tamarin together with our CEO-falsifying equation, Tamarin successfully proves each property, i.e. secrecy, identity agreement, and strong session agreement, thus verifying the fix in this model of signatures.

3.3.2.2 DEO

It was later discovered [19] that the adversary could also change the *message* that the signature verifies for and, more troubling, that this appears to be possible in practice whenever the original attack was possible. We mark in red the differences between the following definition and that of CEO.

Definition 30. [138, Def 2] A signature scheme fails to provide *Destructive Exclusive Ownership (DEO)* if there is an efficient algorithm $\texttt{fake}(pk, (sig, m)_i)$ that given a public key and a sequence of message and signature pairs under that key, outputs (m', pk', sk') such that pk', sk' are a key pair, $pk' \neq pk$, $m' \neq m_j$, and $\mathsf{Vf}(pk', m', sig_j) = \mathsf{true}$ for some j.

This is equivalent to Message Key Substitution [125]. [138] also defines Universal Exclusive Ownership (UEO) as the combination of both CEO and Destructive Exclusive Ownership (DEO).

To model this symbolically we adapt the no-CEO definition in the following way. We first note if there exists a `fake` function as in the DEO definition, then the adversary can choose a second, distinct message m'. Unlike in the pk' case, there the distinctness does not follow from the construction. We cannot directly model this distinctness by using equational theories. We therefore model it through Tamarin's support for restrictions and rules, which enable a form of conditional rewriting. A standard restriction is `Neq(x,y)`, which only enables the transition if $x \neq y$. We add an equation as before, but mark the function that models `fake` as a *private function*. The adversary cannot apply private functions itself. Instead, we give the adversary access to the function through a rule, which enables us to enforce the restriction. This leads to the following model:

```
functions: DEOgen/2 [private]
equations: verify(sign(m1, sk), m2, pk(DEOgen(m2,sign(m1, sk)))) = true

rule make_DEO_sk:
[In(<m2,sign(m1,sk)>)]
--[Neq(m1,m2)]->
[Out(DEOgen(m2,sign(m1,sk)))]
```

No-DEO: Model for signature schemes that do not satisfy DEO.

After adding this equation and rule to Tamarin, we rerun our analysis on STS-ID and discover a UKS attack. This attack was reported in a short paper by Baek and Kim [19]. The attack proceeds in much the same way as the attack on the original STS-MAC protocol.

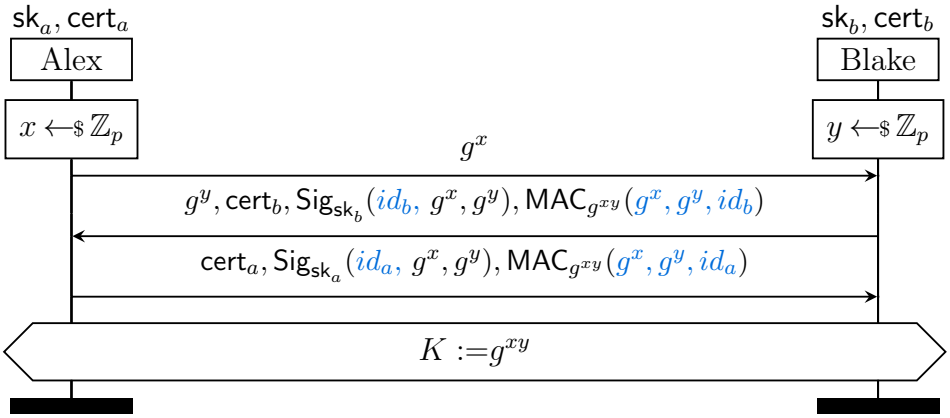


Figure 3.3: Sketch of STS-ISO [38], which includes the identities of the communicating parties.

The adversary waits until A sends the final message and then produces a public key for which A's signature will verify, but for a message altered to include the adversary's identity.

Chevalier and Kourjieh [54] considered the decidability of protocol verification in the context of CEO and DEO, but neither implemented their algorithm nor considered any other properties. Interestingly, some years later, STS-MAC and STS-ID were proven to be secure against UKS attacks using the traditional symbolic model of signatures in [143]. This was possible since the traditional symbolic model did not take these signature behaviours into account, thereby missing the previously published UKS attacks [19, 125].

Baek and Kim [19] recommended adoption of an alternative variant of STS-MAC, dubbed 'key agreement mechanism 7' in ISO/IEC 11770-3 [90], which we show in Figure 3.3 as STS-ISO. Here, instead of including the identities of the communicating parties under the signature, they are directly MACed. We analyse this protocol in Tamarin with respect to these DEO and CEO equations and find the protocol is proven secure under this model. We return to the security of this STS variant in the next section.

3.3.3 Malleability

Signature schemes that are provably secure in the standard sense of being EUF-CMA may still be *malleable*. If a signature scheme is malleable, successful verification does not preclude that the signature was modified. In contrast, non-malleability implies that if a signature is verified under an honest public key, the signature is the same as one produced by the honest party.

Non-malleability is not implied by the standard forgery definition, which only describes the difficulty of producing a signature which verifies under a different message. For example, ECDSA [154] and EdDSA are malleable. Interestingly, whether or not this poses a problem is the subject of dispute between signature scheme designers and implementers. For example, Ed25519 was originally designed without regard for malleability [29], whereas

the Internet Engineering Task Force (IETF) standardisation body decided to explicitly require implementations to enforce non-malleability [96].

In practice, the design of security protocols may implicitly rely on the assumption of non-malleability, while their instantiation may only use a EUF-CMA-provably secure signature scheme.

Symbolic model for malleability: We provide an additional capability to the adversary allowing them to make a new signature out of an old one. This can be done in several straightforward ways; the main insight is to explicitly express the "malleable" part of the signature construction.

This can be modeled in Tamarin's framework by extending the term model for signatures with an additional argument, abstracting the malleable information. The signature convention then becomes

$$sign(m, rep, sk)$$

where m represents the signed payload data, sk the signing key, and rep the malleable format. The corresponding verification remains similar to the existing one, in the sense that it ignores rep and works as before on m and sk. We also provide the adversary with an operation that allows them to change (only) the malleable part:

```
functions: mangle/2
equations: mangle(sign(m,r,sk),repnew)= sign(m,repnew,sk)
```

Malleability: Model for malleable signature schemes.

This additional capability enables the adversary, given a single signature, to produce an arbitrary number of different ones, that all verify under the same public key and message.

Using this model, we analyse STS-ISO. Now the property of strong session agreement fails, as the adversary can alter a signature whilst it is in flight, but both parties believe it is valid. Consequently one party accepts a message which was not transmitted by the other, breaking agreement.

Unlike the other properties in this section, malleability has long been accepted as problematic behaviour, leading to the introduction of a stronger definition of signature scheme security, Strong Existential Unforgeability under an Adaptive Chosen Message Attack (SEF-CMA) [6, 45]: instead of finding a new *message* with corresponding signature, the adversary just has to find a new valid *pair*, and may reuse a queried message. Signatures with SEF-CMA are not malleable, and implementing STS-ISO with such a scheme would provide strong session agreement. In §3.3.6 we analyse a protocol which achieves this without requiring a SEF-CMA scheme.

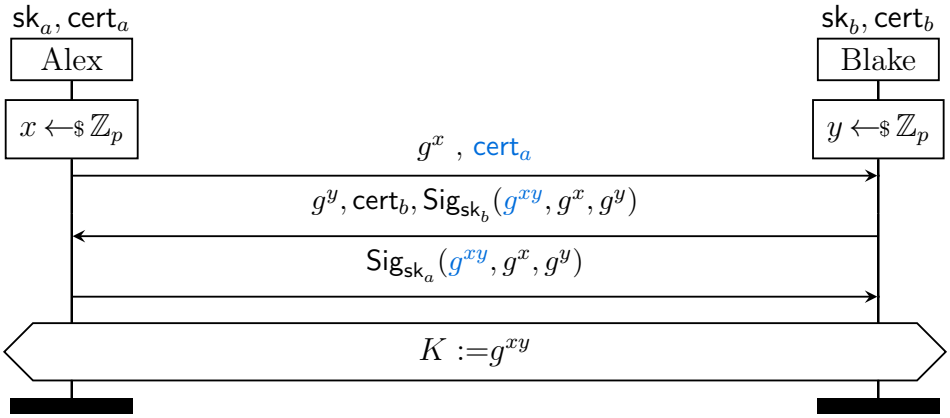


Figure 3.4: Sketch of STS-KSIG, a synthetic variant of STS-MAC which fixes the identities in the first message but drops the MAC in favour of signing the secret.

3.3.4 Re-signing

As we saw in §3.2.2, traditional symbolic models have considered the distinction between message-revealing and message-hiding signatures. However, in some signatures schemes a signature contains enough information for an adversary to produce their own signature on the same message under their own key. We model this by providing the adversary with the explicit capability to re-sign signatures, even if the signature is message-hiding and the message itself is secret.

```
rule: ReSign
[ In( sign(m,r,sk1), sk2 ) ]
-->
[ Out( sign(m,r,sk2 )) ]
```

Re-sign: Model for re-signing an unknown message.

To illustrate the implications, we introduce the following synthetic variant of STS-MAC which we call STS-KSIG and present in Figure 3.4. Here, identities are fixed in the first message. However, the protocol has dropped the MAC value in favour of directly signing the secret value.

Analysing this protocol with the key substitution model finds no attacks, since the responder's view initiator's key is fixed before the initiator discloses a signature. However, when we add the re-signing equation we immediately discover an UKS attack, as the adversary can form a new signature on the secret key from Blake's response, consequently claiming Blake's DH public key as their own, despite not knowing the shared key. Much like the UKS attacks on STS-MAC and and STS-ISO, this attack violates identity agreement.

3.3.5 Colliding Signatures

Stern et al. [154] give an algorithm to produce a signature and public ECDSA key against which *two* messages of the adversary's choice both verify. Ed25519 allows this behaviour to a much greater degree: selecting signature and public key values from the order-eight subgroup leads to verification passing for *any message* with high probability. The

design paper for Ed25519 [29, Page 7] explicitly notes this behaviour and argues it is not problematic, while implementations are split: LibSodium rejects low order elements [95], but several other implementations accept them.

Definition 31. We say a signature scheme has *non-colliding signatures* if it is computationally infeasible for an adversary to produce a public key and signature combination such that verification of more than one message succeeds with non-negligible probability. That is: it is infeasible to select private and public keys and a signature, sk, pk, s, such that there exist messages m_1, m_2, for which $\mathsf{Vf}(s, m_1, pk) = \mathsf{true}$ and $\mathsf{Vf}(s, m_2, pk) = \mathsf{true}$.

Colliding signatures violate two implicit properties that protocol designers sometimes rely on:

(i) if a signature verifies against a particular public key and message, then the signer knew the message that was signed; and

(ii) for a given signature and public key, there exists a unique message which will verify under it.

```
functions: weak/1
equations: verify(sign(m1,r,weak(x)),m2,pk(weak(x))) = true
```

Colliding Signatures: Model for colliding signatures.

This model allows for the worst case behaviour, where a particular signature and public key will verify for any message without requiring the adversary to pick the messages they wish to collide in advance. We consider a (synthetic) variant of the previous protocol where the signatures are encrypted under the recipient's public key, dubbed STS-SCRYPT and shown in Figure 3.5, using the equational theory for colliding signatures. STS-SCRYPT patches STS-KSIG to prevent the adversary from re-signing a message; as well, the key substitution equations cannot be applied as there is no visible signature for the adversary to steal.

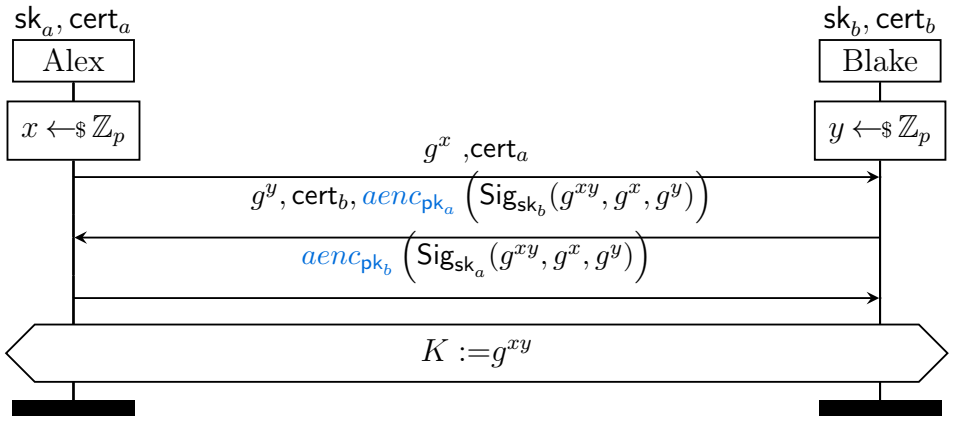


Figure 3.5: Sketch of STS-SCRYPT, which patches STS-KSIG by asymmetrically encrypting the signatures.

However, running Tamarin with our colliding signature equation reveals an attack. The adversary can simply register a colliding public key and asymmetrically encrypt their

own colliding signature. The resulting value will verify with high probability, even though the adversary does not know the message being signed.

3.3.6 Fixing STS-MAC

An alternative to STS-ID was proposed in [38], which we refer to as STS-KDF and show in Figure 3.6. STS-KDF works just like STS-MAC but then uses a Key Derivation Function (KDF) to bind the shared key to the identities of the participants, $KDF(g^{xy}, id_a, id_b)$, instead of using only the shared Diffie-Hellman secret g^{xy}.

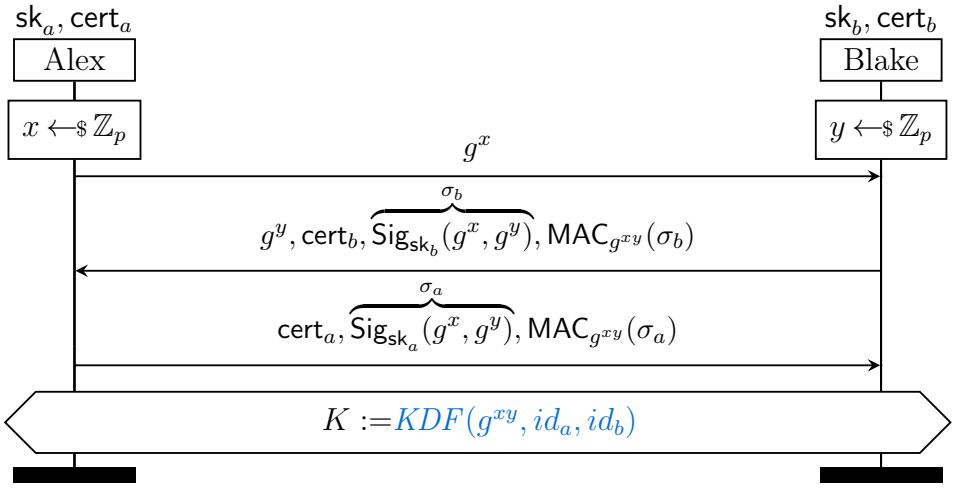


Figure 3.6: Sketch of STS-KDF [38], which patches STS-MAC by including the identities of the communicating parties in the KDF.

Noting that KDFs were poorly understood, the authors explicitly recommended using STS-ID over STS-KDF. It has been nearly two decades since this paper was published and KDFs have stood the test of time. We analyse STS-KDF in Tamarin, allowing the adversary to use all of the new properties we have discussed in the previous section and find that Tamarin proves that all three properties hold, making this protocol the only STS variant we have considered which satisfies all three security requirements (in this signature model) whilst only using EUF-CMA signatures. However, can we be sure no additional signature equations exist which will break this protocol? We return to this question in §3.4.

3.3.7 Summary

We summarise our analysis results in Table 3.1. We note that the traditional symbolic model fails to find any of the attacks we have discussed here. Our attack finding models only slightly increase running times, which implies the approach is tractable. This is better than we expected, considering the additional behaviours that Tamarin must consider, and that we did not introduce any new heuristics for this model.

STS-MAC variant	Signature Model	Security property Sec	IA	SSA	Time in seconds
MAC	Traditional	✓	✓	✓	14
	no-CEO	✓	●	●	35 ∗
	SVS	✓	●	●	23 ∗
ID	Traditional	✓	✓	✓	14
	no-DEO	✓	●	●	68 ∗
	SVS	✓	●	●	16 ∗
KSIG	Traditional	✓	✓	✓	13
	Re-sign	✓	●	●	46 ∗ †
	SVS	✓	●	●	30 ∗
SCRYPT	Traditional	✓	✓	✓	16
	Coll.	✓	●	●	25 ∗
	SVS	✓	●	●	23
ISO	Traditional	✓	✓	✓	17
	Mall.	✓	✓	●	34
	SVS	✓	✓	●	25
KDF	Traditional	✓	✓	✓	3
	All in §3.3	✓	✓	✓	19
	SVS	✓	✓	✓	9

Table 3.1: Verification results when applying our various Tamarin models to a number of distinct STS-MAC variants.
Sec, IA and SSA are respectively the security properties of key secrecy, identity agreement and strong session agreement.
✓ indicates that Tamarin successfully verified the property
● indicates that Tamarin found an attack
∗ indicates that attack finding was done in the bounded model
† indicates that the non-default `i` heuristic was used for the proof

We have seen several properties of signature schemes, none of which are implied by EUF-CMA, the traditional (and most prevalent) definition of signature scheme security, nor are they the result of implementation mistakes. In practice, many signature schemes do not meet these properties: in Table 3.2 we gave a list of widely-used signature schemes and whether the subtle behaviours are present.

Although from a protocol designer's perspective many of these properties are undesirable, they are prevalent in today's signature schemes, and we are not aware of any schemes which provably avoid all of the subtle behaviours described here. In the future, particularly in the post quantum setting, new signature scheme primitives may well avoid these subtle behaviours, but post quantum signatures are not yet standardised or deployed. Consequently, tooling that can analyse protocols using these signature schemes and capture the subtle behaviours is of considerable interest.

Signature scheme	CEO/DEO	No-Mall.	No-ReSign	No-Coll.
RSA-PKCSv1.5	● [138]	● [137]	● [93]	▲
RSA-PSS	● [138]	▲	● [113]	▲
DSA	● [138]	✓ [154]	▲	● [163]
ECDSA-FreeBP	● [38]	● [154]	▲	● [154]
ECDSA-FixedBP	✓ [125]	● [154]	▲	● [154]
Ed25519	✓ [89]	● [29]	✓ [29]	● [29]
Ed25519-IETF	✓ [89]	✓ [96]	✓ [29]	● [29]

Table 3.2: Subtle behaviours of concrete EUF-CMA-secure signature schemes. Columns refer to the security property, i.e., the *absence* of some unexpected behaviour: CEO/DEO (no DSKS), non-malleability, non-resignability and non-collidability. FreeBP (resp. FixedBP) means the signature scheme's base point is considered a parameter of the signature (resp. fixed in advance).
✓ means that the security property holds, and therefore the corresponding unexpected behaviour is not present.
● means that the behaviour is present. For example, ECDSA-FreeBP signatures are malleable and allow for DSKS attacks.
▲ means that we conjecture that the behaviour is absent, so the security property holds, but this has not been proven.

Unfortunately the existing definitions of symbolic signatures in current tools implicitly assume all of these subtle behaviours are absent, which means that they cannot discover the corresponding attacks. To remedy this, we presented symbolic models for the absence of these properties, which enable finding those "invisible" attacks.

3.4 A New Symbolic Model for Verification

In the previous section, we characterised a number of behaviours not captured by the traditional symbolic signature model and repaired the deficiency. However, each improvement is ad-hoc, designed to only capture a known behaviour, and provides no assurance that further, more subtle, behaviour has not been omitted. Thus, while the models in the previous section are extremely effective for attack finding, they raise the obvious question for verification: did we model enough behaviour, or are there more attacks that these models would miss?

In this section, we resolve this question through the development of an entirely new symbolic model for the verification algorithm of digital signatures, directly inspired by the standard computational security definition for signatures and which we call *Symbolic Verification of Signatures* (SVS). It is a symbolic model for signature verification that makes minimal assumptions, relying only on the implications of existential unforgeability.

3.4.1 Specification

Revisiting the definitions of *correctness* and *forgery resistance* from §3.2.1, we note the behaviour of the verification function is specified only when the public key is honestly

generated. To reiterate, the first requirement is to accept correctly generated signatures for honest public keys:

$$\forall pk, sk, m \in M : (pk, sk) \leftarrow \texttt{gen}() \Rightarrow \texttt{verify}(\texttt{sign}(sk, m), m, pk) = \mathsf{true}$$

The second implies that if a signature can be verified with an honestly generated public key for some message m, then m and the matching signing key were previously used to produce the signature:

$$\forall pk, sk, m \in M, s : (pk, sk) \leftarrow \texttt{gen}() \wedge \texttt{verify}(s, m, pk) = \mathsf{true} \Rightarrow \text{previously: } \texttt{sign}(sk, m)$$

We note that whilst `sign` can be a probabilistic algorithm, `verify` is implicitly assumed to be a deterministic algorithm.

There are no further requirements on the output of `verify` given by the standard computational definition. The traditional definition of symbolic verification, shown in §3.2.2, agrees with the computational definition but further specifies that `verify` implicitly outputs false in any situation in which it is undefined, i.e., verification against a malicious key. Many of the equations we gave in §3.3 essentially remedied this in an ad-hoc fashion, by specifying additional cases where `verify` could return true.

We now build a symbolic definition of `verify` that agrees with the computational one: where its output is not otherwise constrained, we let the adversary choose whether it returns true or false. This definition encompasses our previous equations for maliciously generated public keys, as well as further unknown equations, so long as they are not ruled out by the computational definition.

In a symbolic setting, we consider traces made by a series of transitions of a labeled transition system, which describes the state of the protocol at that point in the trace. When signature verification occurs in a trace, we now require the following constraints to be observed:

1. If the public key was honestly generated, the verification of a corresponding honest signature must succeed.
2. If the public key was honestly generated, the verification of a forged signature must fail.
3. If this particular message-signature-key triple has been verified before, the result is defined by the previous answer.
4. Otherwise, consider all possibilities—i.e., let the adversary decide.

We could model the first and second constraints purely in the term algebra. Similarly, the fourth constraint corresponds to allowing the adversary to send a value over a channel to the protocol. The third constraint is the interesting one because it requires storing (monotonic) state about previous queries. Succinctly, the verification output depends both on the "local query", the history of the trace, and the adversary's current choice.

Consequently our symbolic model must record whether public keys have been created honestly, and what verification checks have been made previously. We now give an implementation of this abstract specification in Tamarin.

3.4.2 Tamarin Implementation: User-Visible

We use the function signature as defined in §3.3.3. We allow for public key and message extraction as the modeller wishes. We omit the verification function and its associated equation, and will replace it with a different mechanism that makes minimal assumptions on the properties of the scheme.

In previous approaches, the signature verification function was encoded into the term algebra, and explicitly stated under which conditions signature verification returns true. Here, we will instead only specify *restrictions* for the signature verification results, and consider all possibilities in other scenarios. To implement this, we specify signature verification as an annotation on a protocol rule, which guards the transition according to a series of first order logical statements we give below. We begin by defining the trace annotations that we use for restriction checks. Each of these annotations can be added to Tamarin as an *action* in a mechanical fashion for the rule using it:

Honest Key generation Where the *protocol* honestly generates a public key pk, we label the corresponding transition with the action $Honest(pk)$. This specifies that the public key has been output by the generation algorithm for signatures and consequently, the various restrictions on the signature verification algorithm will apply if a signature is tested against this public key.

Signature Verification Where a protocol will only make a certain transition conditional on the result of a signature verification result (be it true or false), we will provide an action label for this occurrence. Unlike the $Honest(pk)$ label, we will later use *restrictions*, first order formulae, to restrict situations under which this transition can occur. When a protocol wishes to verify a particular signature term sig against a particular message and public key combination (labelled tm and tpk), we will write $Verified(sig, tm, tpk, result)$ where $result$ may be true or false depending on whether the transition should be allowed to occur.

Manipulation of Honest Signatures We also provide the equation for malleable signatures and the rule for re-signing we discussed in §3.3.3 and §3.3.4. Malleability allows an adversary to manipulate an honest signature and is therefore not part of our improvements to the signature verification algorithm. Re-signing has a very subtle usage for the adversary: if (i) the adversary compromises the private key of an honestly generated key pair, (ii) the signature theory is not message revealing, and (iii) the adversary is

in possession of a signature for an unknown message, then the adversary can use the re-signing rule to generate a new signature, under the compromised honestly generated key, for the unknown message. As this refers to the generation of a new signature under an honestly generated public key, it is orthogonal to our changes to the specification of signature verification for malicious public keys.

3.4.3 Tamarin Implementation: Internal

Syntactic Transformations Behind the scenes, we will perform a mechanical transform of
$Verified(sig, tm, tpk, result)$ into an action fact $Verified(sig, sm, spk, tm, tpk, result)$ using the extraction functions described in the listing below. We define $sm = e1(sig)$ and $spk = pk(e3(sig))$. This transformation is needed for purely technical reasons: Tamarin requires reducible functions to be specified in the action fact annotation rather than in restriction formulae.

```
functions: e1/1, e2/1, e3/1 [private]

equations: e1(sign(x,y,z)) = x
           e3(sign(x,y,z)) = z
```

Extraction Functions for Signatures. These are not used by the protocol or the adversary, just by the implementation.

These functions allow us to easily refer to the message and public/private key that a signature corresponds to. Note that in the event the signature is not honestly generated, these functions are still well defined, but simply do not yield a result (technically, they will not reduce).

We now provide a series of *restrictions* which restrict the traces that can occur. All of our restrictions concern the behaviour of the signature verification function.

Correctness This requirement follows directly from the requirement that an honestly generated public key, an honestly generated signature, and the correct message must verify as true.

$$\texttt{Correctness} : \forall sig, tm, tpk, t_1, t_2.\ Honest(tpk)@t_1 \wedge$$
$$Verified(sig, tm, tpk, tm, tpk, \mathsf{false})@t_2 \implies \bot$$

NoForgery Here we state that if a signature verification does succeed against an honest public key, then the signature must have been honestly produced.

$$\texttt{NoForgery} : \forall sig, tm, tpk, sm, spk, t_1, t_2.\ Honest(tpk)@t_1 \wedge$$
$$Verified(sig, sm, spk, tm, tpk, \mathsf{true})@t_2$$
$$\implies sm = tm \wedge spk = tpk$$

Consistency Verification is typically defined as a deterministic function, here we specify that repeated calls to verify will always return a consistent answer.

$$\begin{aligned}&\texttt{Consistency} : \forall sig, sm, spk, tm, tpk, r_1, r_2, t_1, t_2.\\&\mathit{Verified}(sig, sm, spk, tm, tpk, r_1)@t_1 \wedge\\&\mathit{Verified}(sig, sm, spk, tm, tpk, r_2)@t_2 \implies r_1 = r_2\end{aligned}$$

The result of this model is that if a particular transition is labeled with a verification annotation it will be allowed to occur unless it violates one of these three restrictions.

If we compare these restrictions to our earlier specification, we note the following: in the event that the signature is being verified against an honest public key, *Correctness* ensures honest signatures will be accepted and *NoForgery* ensures forged signatures will be rejected. Otherwise, the only rule that will apply is *Consistency* which simply ensures signature verification calls are deterministic. It is straightforward to see how this presentation matches our earlier specification, as this expresses our required properties to Tamarin directly. Consequently, this *Symbolic Verification of Signatures* (SVS) model allows the adversary to perform key substitution attacks, craft colliding signatures and many other known or unknown behaviours concerning maliciously chosen public keys.

3.4.4 Results

We now show that the above model is tractable in practice and we present results and running times in Table 3.1. We achieve automatic termination on every case study with Tamarin's default heuristic. Further, our SVS model appears to be faster in practice than our attack finding models. We suspect this is because our SVS model does not introduce any case splits unless the attacker's manipulation of a signature becomes relevant. Whereas the attack finding models require Tamarin to consider a separate case for each possible way a signature could have been constructed whenever a signature term is introduced.

This model, as close as it is to the computational definition, forces us to consider issues not normally raised in a symbolic analysis. Traditional symbolic tools often produce an attack trace that is practical in reality, as it consists of a series of explicit capabilities provided to the adversary. In contrast, our SVS model is closer to the computational model in the sense that the attack trace will consist of the adversary specifying certain signatures pass or fail verification, but providing no intuition on how an adversary may arrange for this to happen.

Consequently, both our SVS model and our earlier models for attack finding are independently of interest to protocol modellers. First, the SVS model (§3.4) should be used and if Tamarin returns a proof, it is within the strongest model of signature security we have described. However, should it return an attack trace, the modeller can use our *attack finding models* (§3.3) to effectively recover practical attacks that could be used in

reality. By using the attack finding model for each property separately, it is possible to isolate the signature behaviour which is leading to the attack and thus consider possible mitigations. We demonstrate this functionality on our case studies in the next section.

It is possible that our SVS model returns that an attack is found, but none of our falsification models yields an attack. In this case, we suggest it is best to think of the result as "Not Proven". There may well be an attack as the protocol requires behaviour of signatures not provided by the standard definition, yet TAMARIN is not aware of a method of crafting public keys or signatures to enable the attack in practice. This means one does not have the desired symbolic proof, but still gets the proofs for the falsification models, which is a much stronger guarantee than previously possible.

3.5 Further Case Studies

We now demonstrate the utility of our approach on more complex case studies. We automatically find three attacks on different real world protocols, two of which are novel and previously unreported. Furthermore, all three protocols have undergone previous formal analysis using the traditional model of digital signatures and reported to be secure:

(i) a known attack on an earlier version of the Let's Encrypt certificate issuance protocol, arising from a key substitution property;

(ii) a previously unreported attack on a WS-Security Handshake, arising from a key substitution property. This attack was missed despite having been analysed using ProVerif in two separate papers.

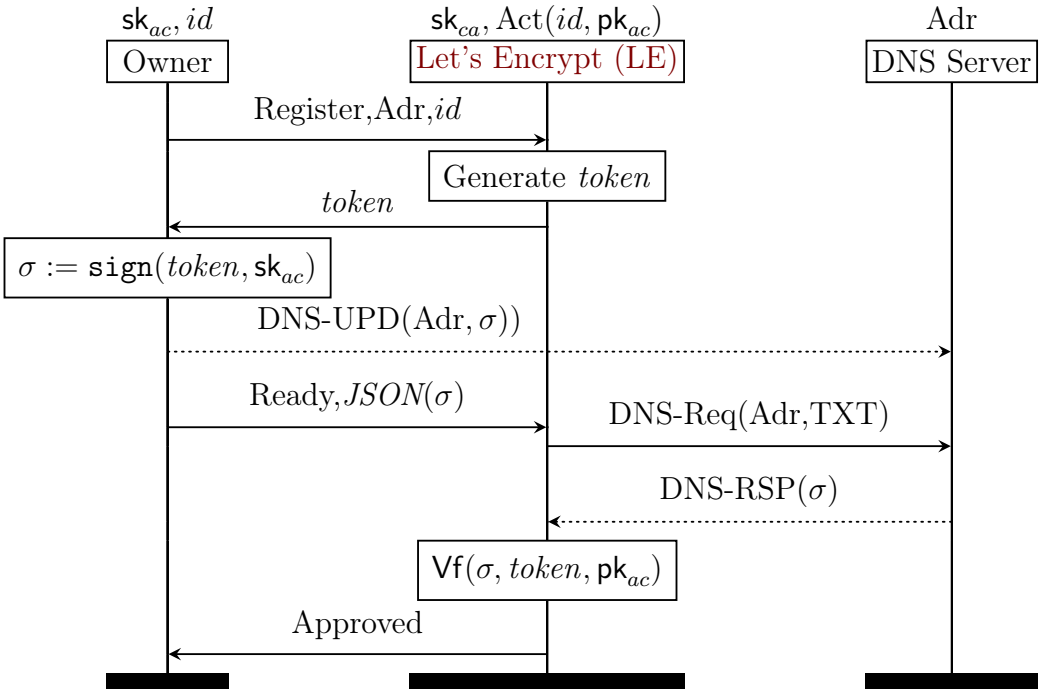
(iii) a previously unreported attack on DRKey, a key exchange protocol, arising from a weak signature property. This last attack was missed in a previous formal analysis, and allows us to violate the security claims of OPT, an origin and path tracing protocol which uses DRKey.

3.5.1 Let's Encrypt

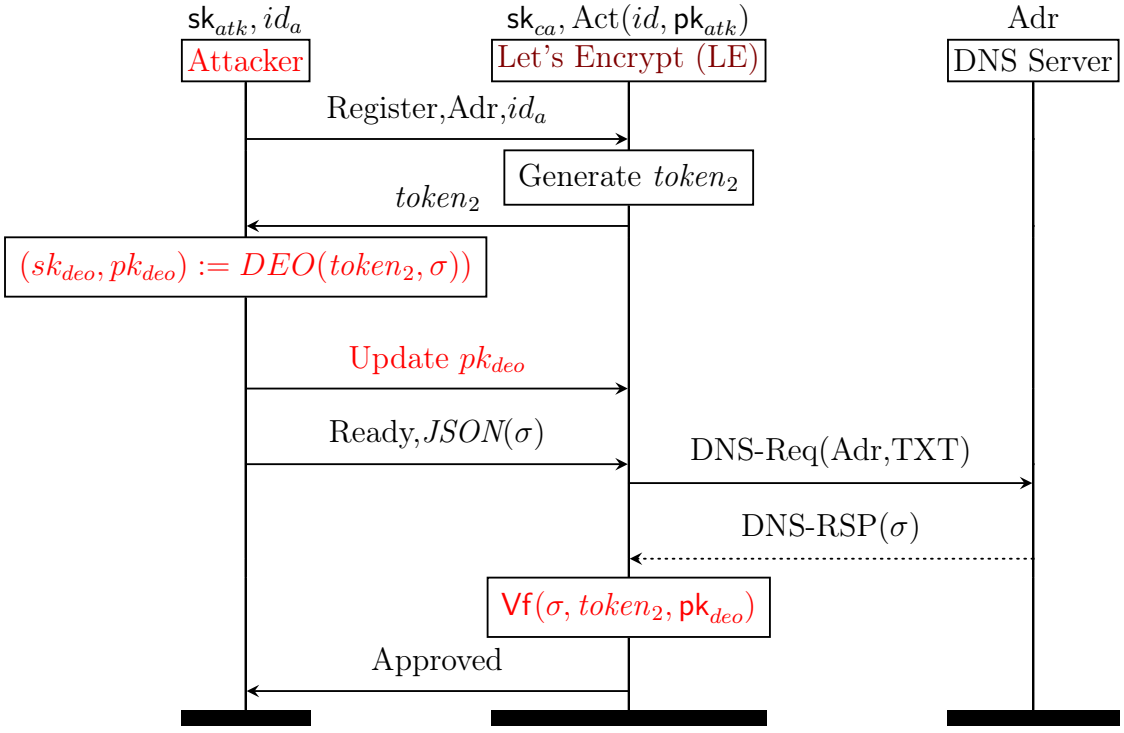
Let's Encrypt (LE) is the world's most popular Certificate Authority. To issue certificates automatically, it uses the ACME protocol for issuance, renewal and revocation, as standardised by the IETF [16]. ACME allows a website owner to prove ownership of a domain and request a certificate from a CA via a choice of signature-based challenge-response protocols. If the protocol succeeds, LE issues a certificate to the owner.

ACME went through a number of drafts prior to (and after) release. Draft Barnes 01 [1] was the first incarnation in which DNS challenges are completed by placing a nonce in a DNS record. The DNS challenge mechanism was then updated in Draft Barnes 03 [2] to be a signature over the nonce by the account holder's public key.

Draft Barnes 04 [3], only a minor refinement of 03, was then adopted by the ACME working group as IETF Draft 00 [14], at which point—only six weeks before LE was



(a) Normal Operation of the Automatic Certificate Management Environment (ACME) Draft 00 Protocol



(b) Attack on the ACME Draft 00 Protocol

Figure 3.7: ACME Draft 00 Let's Encrypt DNS Challenge Response Protocol. The dotted arrows indicate that the channel is assumed to be authentic.

loaded into major browsers' certificate stores—a signature key substitution attack was discovered and reported [4] to the IETF ACME mailing list. (IETF ACME Draft 00 is also known as Barnes Draft 04.) The attack allowed an active attacker to pass the ACME

challenge and receive a valid Transport Layer Security (TLS) certificate for any website using LE DNS Challenges, and thus intercept and modify any such website's TLS traffic. This prompted the DNS Challenge to be updated to a hash of the account public key and the nonce (known as a key thumbprint) in IETF Draft 02 [15]. This mechanism remains in use today [16].

The attack stems from Draft 00's use of a DNS-based signature challenge, shown in Figure 3.7a: the website owner requests a random nonce from LE, signs the nonce with a key to be used in the new certificate, and places the signature in a DNS record for the domain. LE then extracts and verifies the signature from the website's DNS records, concluding that the owner controls (i) the claimed private key and (ii) the DNS records for that website. Based on that conclusion, it issues a certificate for the corresponding public key.

In the attack, depicted in Figure 3.7b, suppose Alex has completed a LE ACME challenge as normal, and has placed the signature in their DNS records. The adversary can begin a new instance of the challenge response protocol with the CA, claiming ownership of Alex's website and receive a token to sign and display in Alex's DNS records. The adversary then performs a key substitution (no-DEO) attack on Alex's signature and the new token value (and updates their account key accordingly). Afterwards, they trigger the second phase of the protocol by sending the Ready message. LE retrieves Alex's signature and verifies it against the adversary's malicious public key. This succeeds and LE will now issue the adversary a certificate for Alex's website and the adversary's public key.

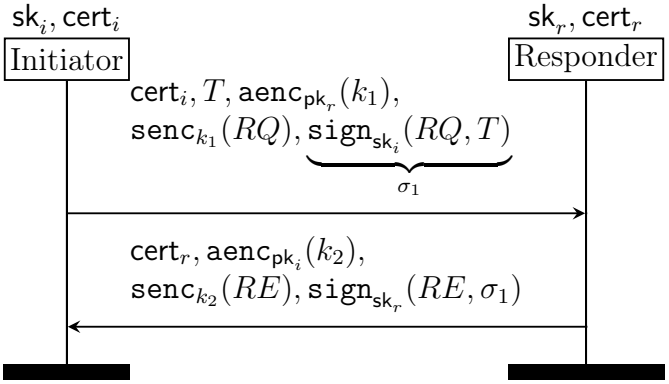
3.5.1.1 Analysis of ACME

We developed a Tamarin model of the vulnerable draft of the ACME certificate issuance protocol. Using our model from §3.4, we automatically find the reported attack. We check that using the traditional symbolic model of signatures Tamarin successfully verifies ACME, confirming that it misses this attack without our improvements. We also provide a Tamarin model corresponding to IETF ACME Draft 02 [15], the patched version of ACME. Although the IETF could have elected to use a signature scheme which provides DEO, they it felt it safer to forgo the use of signatures entirely, instead replacing the signed value with a hash of the account public key and the token. Using our SVS model, Tamarin verifies the attack is no longer possible. We collect these results in Table 3.4.

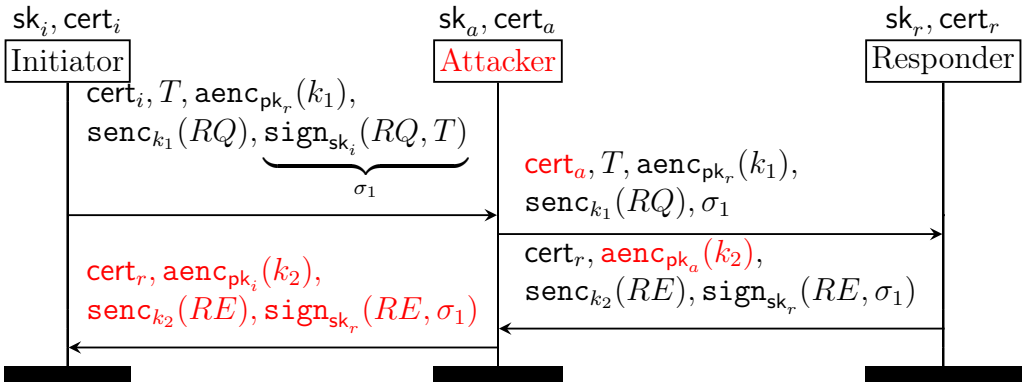
This example also illustrates the complementary uses of SVS and our attack finding models, such as 'no-DEO'. Whilst SVS reports an attack, the attack trace does not correspond exactly to the attack reported on the mailing list [4]—rather, the trace simply allows an adversary to successfully pass the verification directly, since this possibility is not excluded by the EUF-CMA definition. If we then use our 'no-DEO' equation from §3.3, Tamarin recovers the exact attack trace from the initial report. This demonstrates the utility of our two-pronged approach.

Previously, Bhargavan, Delignat-Lavaud and Kobeissi [30] presented a symbolic model of draft Barnes 01 [1] (which they refer to as ACME Draft 1) and draft IETF 00 [14] (referred to as ACME Draft 4) using ProVerif. Due to the traditional symbolic signature model, their analysis missed this attack. In fact, their analysis concluded the (vulnerable) draft IETF 00 satisfied stronger security properties than the earlier (secure) draft Barnes 01, which contradicts our findings.

3.5.2 WS-Security



(a) Normal Protocol Flow. T is a timestamp used to prevent replay attacks, RQ is the request payload and RE is the response payload.



(b) This attack violates request correlation and response secrecy. The Attacker passes off the Initiator's request as their own by replacing the certificate, can then learn the response and can even pass it back to the Initiator. Note that the responder does check the match of signature σ_1 and certificate cert_a, but is fooled due to no-CEO.

Figure 3.8: The WSS1.1-MA-X509-SE protocol from [167] and the attack we automatically discovered.

In 2004, the OASIS Consortium published the Web Services Security Standard [133], which defines a suite of protocols for securing XML web requests and responses without requiring the use of TLS (which was not yet widely deployed). This standard enjoyed considerable popularity until it was overtaken by SAML and later TLS based solutions. Nonetheless, it is still in use and supported by many enterprise frameworks such as gSOAP [84], Apache CXF [10], IBM Websphere [165], and Microsoft's WCF [128].

As well as suffering from a number of implementation flaws, primarily due to the complexities of XML parsing and canonicalisation [166], the complexity and popularity of the standard made it of considerable interest to the automated verification community [18, 32, 33, 34, 35], leading to the creation of verified cross compilers which could accept a protocol specification from the standard and produce both an automated proof of security using ProVerif and an executable implementation in F# [35].

The 1.0 standard published in 2004 was later superseded by the 1.1 standard released in 2006 [133]. One of the motivations for the updated standard was the introduction of *Signature Confirmation*, a mechanism for correlating requests and responses to prevent adversarial manipulation [33]. The principal idea of Signature Confirmation is that after receiving a signed request, the responder's signature should also cover the signature from the request.

Although the standard only directly defined a method for specifying particular message formats and how to parse them, a number of example handshakes and 'scenarios' were also provided. One such scenario which saw widespread adoption was WSS1.1-MA-X509-SE [167] which is depicted in Figure 3.8a. It supports a request response framework where each party holds a X.509 certificate and corresponding private key and claims to provide mutual authentication of the communicating parties, as well as binding requests and response together securely using signature confirmation. In addition to being the default setting in IBM's Websphere Platform [165], documentation of its use as a default can still be found for the Spring Framework [151], the Windows Communication Foundation [164], Oracle's Fusion Middleware [134] and Apache CXF [11].

In 2006, a team at Microsoft Research verified the design of this protocol and the benefits of signature confirmation using ProVerif [32]. In addition to proving the secrecy of requests and responses made in the protocol, they also proved 'request correlation', that every accepted response matched the intended request. This analysis was also followed up on in [35], where it was dubbed 'WS Request-Response' and the authors presented a tool for extracting ProVerif models of the protocol from F# implementations.

Using our new model for signature verification, we revisit this protocol in Tamarin. We automatically discover a number of attacks, the most devastating of which makes use of the no-CEO property to 'steal' a client's request and is depicted in Figure 3.8b. Not only does this violate the request correlation property that signature confirmation was introduced to ensure, but furthermore the attacker can learn the contents of the response to the honest request, violating the secrecy requirement.

There are many scenarios in which this would be damaging, notably if the request contained login credentials and the response a cookie or other secret authentication response. The previous analyses in ProVerif could not have discovered this attack, as they used the traditional symbolic model of signatures, which does not consider these types of attacks.

We stress that whilst we demonstrate our attack on this particular protocol, it is the very mechanism of signature confirmation which is flawed. Signing a signature does not (necessarily) create a unique binding to the contents or public key of the signed signature. Instead, it is much better practice to directly sign the original message and original public key. Using our SVS model, we verify that this proposal fixes the security issues in the original protocol.

3.5.3 DRKey and OPT

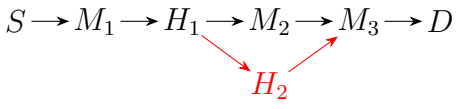
The "Dynamically Recreatable Key" Protocol (DRKey) was first published in 2014 [100] and was supported by a mechanised proof performed using Coq [168]. It is a lightweight key exchange protocol for routers on a packet-switched network to agree on symmetric keys, used as part of a secure routing architecture.

At a high level, DRKey participants generate directional symmetric secret keys, one for use with each other participant. They send both a public-key encryption and signature of the key to the recipient, thereby securely transporting and authenticating the keys to other participants. These keys are then used as part of a higher-level protocol called "Origin and Path Trace" (OPT) [100]. OPT aims to prevent malicious routers from altering the paths of packets through a wider network, using the keys generated by DRKey to authenticate each link in turn. One of OPT's security goals is that malicious routers should only be able to affect routes to their immediate neighbours:

> "When there are multiple **adjacent** malicious nodes on the intended path, a wormhole is present: an honest node down the path can only conclude that the packet has entered the hole via the first malicious node and exited from the last malicious node."

Zhang et al. [168, Section 6.2] presents a formal analysis and claims that this non-collusion property holds. We automatically find a previously unreported attack on this property with Tamarin. We also show that using the traditional model of digital signatures leads to a successful Tamarin verification which misses our attack.

We describe the attack using an example topology, in which S and D are an honest source and destination, H_1, H_2 are honest routers, and M_1, M_2, M_3 are malicious routers. S wishes to send a packet to D along the intended upper path shown in black. H_2 is an honest router, not on the intended path; the malicious routers collude to route the packet through H_2 on the lower path (in red) while S and D believe that it took its intended route via M_2. This violates the security requirement we quoted earlier, which requires that the packets travel the edge $H_1 \to M_2$, whereas due to our attack they will instead transit $H_1 \to H_2$.



The attack arises because of the ability to re-sign *secret* messages under a new key. The message used to pass keys in DRKey is:

$$\mathtt{aenc}_{\mathsf{pk}_{S,t}}(K_{H_2,S}), \mathtt{sign}_{\mathsf{sk}_{H_2}}(K_{H_2,S}, \mathsf{pk}_{S,t}, S)$$

Here we show the message produced by H_2, carrying keys intended for S in the session using the temporary public encryption key $\mathsf{pk}_{S,t}$. During the DRKey protocol, the adversary as M_1 can forge a message to H_2, claiming that S wishes to agree on keys for the lower path. When M_3 receives the DRKey message from H_2, containing a signature and an encrypted key, the adversary (as M_3) can re-sign the packet as if it came from M_2 and pass it on to D. This is possible even though the adversary does not know the message, and is a behaviour not captured by traditional symbolic models. Note that even if the DRKey implementation uses a signature scheme where re-signing is not possible, e.g., Ed25519, the colliding signature property can also be used to craft a signature for an unknown message. The rest of DRKey proceeds as normal, at the end of which S and D each hold a key they believe they share with M_2, but in fact they share this key with H_2. This constitutes a UKS attack on the DRKey Protocol.

The OPT protocol then prescribes a series of chained MACs such that honest routers can detect maliciously-routed packets, and that the destination can verify that the correct path was followed. However, in the above context, S intends to route a packet via M_2 but M_1 can maliciously alter the route to the lower path. Because S and D share a key with H_2, that they *believe* they share with M_2, neither of them can detect this malicious routing As a consequence of this attack, M_2 could bill S for routing packets, despite in fact offloading all of the transmission work to the unsuspecting H_2.

Perhaps a simpler form of misbehaviour would be rewriting the route $S \to M_1 \to M_2 \to M_3 \to D$ to $S \to M_1 \to \underline{H} \to M_3 \to D$. However, this does not technically contradict OPT's claimed security goals, while our example violates their non-collusion property. The malicious router M_1 is necessary in order to change the route of the packet in the first place.

We stress that whilst we demonstrate this attack on the example topology described above, it applies generally to any topology where an adversary can control two (or more) adjacent malicious routers and at least one router earlier in the chain. Besides the double billing attack we mentioned earlier, this attack could also be used to perform a denial of service attack on an honest router, by forcing additional packets to pass through it, despite the fact both source and destination believe their packets are travelling a different route.

As DRKey is intended for use in the SCION [135] internet architecture it is still under active development. The DRKey authors agree that the attack we found is serious and have modified their protocol according to our proposed fix. The prototype is already

updated and this will be reflected in an extension of their work, which is currently under submission for publication.

Our proposed fix for this protocol follows the intuition behind STS-KDF. We do not need to change any of the messages on the wire, instead, we apply a key derivation function which binds each key to the identity of the party who is using it, and the party they believe they share it with. This suffices to prevent any unknown key share attacks on DRKey, as honest parties will only agree on keys if they also agree on identities. Using our SVS model of digital signatures, Tamarin verifies the fix in only 7 seconds.

3.5.4 Summary

	Previous traditional verification			Attacks found in this work by adding our new signature models				
Protocol	**Ref**	**Year**	**Methodology**	**Properties violated**	**Model**	**Time (s)**	**Sect.**	**First reported**
X.509 Mutual Auth	[32]	2006	ProVerif	Correlation & Secrecy	no-CEO	5	§3.5.2	*This work*
WS Request-Response	[35]	2008	$F\# \rightarrow$ ProVerif					
STS-MAC-fix1	[143]	2012	Tamarin	Authentication	no-CEO	35	§3.3.2.1	[38]
STS-MAC-fix2	[143]	2012	Tamarin	Authentication	no-DEO	68	§3.3.2.2	[19]
DRKey & OPT	[168]	2014	Coq	Authentication & Collusion Resistance	Coll.	2640	§3.5.3	*This work*
ACME Draft 4	[30]	2017	ProVerif	DNS Validation	no-DEO	53	§3.5.1	[4]

Table 3.3: Summary of new findings using our signature models, compared to previous analyses that used the traditional symbolic model for signatures. The previous analyses did not discover any attack on these protocol properties. In contrast, our new approach efficiently finds attacks, including previously unreported ones.

In Table 3.3 we relate our attacks to previously published academic papers. Notably, we have uncovered new attacks on real world protocols that have previously undergone formal analysis. Each attack relies on a subtle signature scheme property, which previous analysis tools could not take into account. We have responsibly disclosed our attacks.

We give a brief summary of the performance of our case studies and their proposed fixes in Table 3.4, showing the overall tractability of our approach. Our combined approach (verification with SVS, attack finding with the equational model) demonstrates its utility here: SVS is both more efficient and finer-grained where the protocol verifies. In contrast, when there is an attack, our attack finding models are quickest.

In conjunction with a companion work, building symbolic models of non-prime order groups [64], we investigated some common cryptographic libraries' handling of Ed25519 signatures. We discovered that whilst LibSodium [110], Golang's NaCl Module [85], Project Everest's formally verified HACL [169] and Cloudflare's CIRCL [58] advertise the same API and 'drop in' compatibility, they are not consistent in their handling of Ed25519 signatures. Notably, LibSodium checks for and rejects 'low order points' which are used to construct colliding Ed25519 signatures. However, the other three libraries accept these points, allowing colliding signatures to be crafted. We reached out to the

Protocol	Signature Model	Analysis results	Time in seconds
WS-Security	Traditional	✓	3
	no-CEO	•	5
	SVS	•	12
WS-Security (fixed)	Traditional	✓	2
	All in §3	✓	12
	SVS	✓	13
LE-00	Traditional	✓	1
	no-DEO	•	53
	SVS	•	98
LE-02	Traditional	✓	1
	All in §3	✓	2
	SVS	✓	1
DRKey	Traditional	✓	240
	Coll.	•	2640
	SVS	•	Manual
DRKey (fixed)	Traditional	✓	4
	All in §3	✓	32
	SVS	✓	7

Table 3.4: Verification results on our further case studies.
✓ indicates that Tamarin successfully verified the property
• indicates that Tamarin found an attack.
Manual indicates that Tamarin's interactive mode was used to reconstruct the attack trace as Tamarin's built in heuristics did not terminate in a reasonable timeframe.

maintainers of each library and they have fixed (or agreed to fix) this issue, ensuring that protocol developers are not caught unaware.

3.6 Conclusions

In this work, we revisited many subtle behaviours of digital signature schemes, such as the possibility of key substitution and malleability, and showed how they fall between the cracks: their absence is not guaranteed by the classical EUF-CMA security definition for signatures, but at the same time their absence is assumed by modern automated protocol analyses. Yet the presence of such behaviours can lead, and has led, to critical attacks.

We developed a range of alternative signature models for use in modern tools. Our models capture a wide range of these behaviours and give a general theory for verification of their absence. We thereby provide the first automated procedure to show the absence or presence of attacks exploiting these subtle behaviours.

As a side effect of evaluating the effectiveness of our work, we found two new attacks

on protocols, which is remarkable for multiple reasons: the WS-Security protocols served as the basis of globally used technologies and were therefore under close scrutiny, and both WS-Security and DRKey were previously proven secure.

In the wider sense, our work increases the scope of attacks considered by automated analysis tools: future protocol analysis models that include our more accurate equations will be able to find more attacks, or show the absence of more attack types.

A more long-term question is whether it is possible to "close the gap" between falsification and verification, showing that any attack found in our general theory corresponds to a real attack on the underlying signature scheme itself.

4

Background on Diffie-Hellman Groups

This chapter is drawn from a literature review I completed as part an earlier intermediate report on my thesis. It has been extended with additional background material and related work which was published as part of

Cas Cremers and Dennis Jackson. 'Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman'. In: *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*. 2019, pp. 78–93. DOI: 10.1109/CSF.2019.00013. URL: https://doi.org/10.1109/CSF.2019.00013.

In this chapter we introduce background material on Diffie-Hellman groups. A DH group is a mathematical structure with special cryptographic properties. We briefly discuss the underlying mathematical foundations and introduce some terminology from Unification Theory. Then we review the history of symbolic Diffie-Hellman models and compare the most widely used contemporary models.

4.1 Mathematical Background on DH Groups

In this section we first recap some definitions and terminology from group theory which will be useful in the subsequent chapters. We also introduce some well known theorems which we will need later. Then we discuss how Diffie-Hellman groups are instantiated in practice and some of the behaviour specific to these instantiations.

4.1.1 Group Theory

Diffie-Hellman groups are a particular class of abelian groups that enjoy some useful cryptographic properties. We begin with some concepts from elementary group theory, a reader without previous exposure may find *Introduction to group theory* by Bogopolskij [43] useful.

Definition 32. Let G be a non-empty, finite, set. Let $\cdot$ denote a binary operation on G, let $\_^{-1}$ be a unary operation and $1 \in G$. We say $(G, \cdot, \_^{-1}, 1)$ is a **finite abelian group** if it has the following properties:

1) Closure $\quad \forall a, b \in G \,.\, a \cdot b \in G$
2) Commutativity $\quad \forall a, b \in G \,.\, a \cdot b = b \cdot a$
3) Associativity $\quad \forall a, b, c \in G \,.\, (a \cdot b) \cdot c = a \cdot (b \cdot c)$
4) Identity $\quad \exists 1 \in G \;.\; \forall a \in G \,.\, a \cdot 1 = a$
5) Invertibility $\quad \forall a \in G \;.\; \exists b \in G \,.\, a \cdot b = 1, a^{-1} = b$

For brevity, we will typically omit additional parameters and simply refer to a G as a group when the other elements are clear from context.

Definition 33. We define **exponentiation** on a group element a by an integer x to be the result of repeated applications of the group operation. That is:

$$a^x = \overbrace{a \cdot a \cdot \ldots \cdot a}^{|x| \text{ times}}$$

If x is negative, we replace a with a^{-1}.

The quintessential feature of a Diffie-Hellman group is that performing the inverse of exponentiation is computationally intractable, i.e. given g^x, it is hard to recover x. This is known as the discrete logarithm problem.

Definition 34. Let $(G, \cdot, \_^{-1}, 1)$ be a group. Let H be a subset of G. If $(H, \cdot, \_^{-1}, 1)$ forms a group, we say that H is a **subgroup** of G and respectively G is a **supergroup** of H.

Definition 35. Let G be a group and S be a subset of G. We say S **generates** G if any element in G can be expressed as a (finite) combination of elements S under application of $\cdot, -^{-1}$. Elements in S are called **generators** of G. If G can be generated by a single element, we say G is **cyclic**. For an element g of G, let n be the smallest natural number such that $g^n = 1$ where 1 is the identity element of G, we say the **order** of g is n and write $|g| = n$. Similarly, we say the order of G is the number of elements in the set. Note that an element of order k generates a subgroup of order k.

Definition 36. Let $(H, \cdot)$ and $(G, +)$ be two groups, a **group isomorphism** from G to H is a bijective function $f : G \to H$ such that for all $u, v \in G$ we have that $f(u + v) = f(u) \cdot f(v)$. We write $G \cong H$.

An isomorphism is a structure preserving map between groups, notably it preserves the algebraic properties of each group, however it does not necessarily preserve the computational properties; an isomorphism need not be effectively computable.

Note that if $|G|$ is prime, then an element is necessarily either a generator or the identity element and hence G is cyclic. Furthermore, if $|G| = n$ is cyclic then $(G, \cdot, \_^{-1}, 1)$ is isomorphic to $\mathbb{Z}/n\mathbb{Z}$ the integers modulo n.

Definition 37. Let $(G, +)$ and $(H, \cdot)$ be a group, their **direct product** $G \times H$ is defined as:

1. The set of elements is the Cartesian product of the underlying sets. Elements are ordered pairs (g, h).
2. The binary operation on $G \times H$ is defined component wise: $(g_1, h_1) \cdot (g_2, h_2) = (g_1 + g_2, h_1 \cdot h_2)$

The following well known theorems will also be of use:

Theorem 1 (Lagrange's Theorem)**.** *Let G be a finite group and H a subgroup of G. The order of H divides the order of G. It immediately follows that for $g \in G$, $|g|$ divides $|G|$.*

Theorem 2 (Cauchy's Theorem)**.** *Let G be a finite group and p a prime number dividing the order of G. Then G contains an element of order p (and hence a subgroup of order p)*

Theorem 3 (The fundamental theorem of finite abelian groups)**.** *Let G be a finite abelian group of order n. Let the unique factorisation of n into distinct prime powers be given by $n = p_1^{a_1} \dots p_k^{a_k}$. Then:*

1. *$G \cong A_1 \times \dots \times A_k$ where $|A_i| = {p_i}^{a_i}$*

2. *For each* $A \in \{A_1, \ldots, A_k\}$ *with* $|A| = p^a$

$$A \cong \mathbb{Z}_{p^{b_1}} \times \ldots \times \mathbb{Z}_{p^{b_t}}$$

with $b_1 \geq b_2 \geq \ldots \geq b_t$ *and* $b_1 + \ldots + b_t = a$

3. *The decomposition given above is unique.*

4.1.2 Families of DH Groups

In practice, Diffie-Hellman groups are typically instantiated as a group over either a Finite Field or an Elliptic Curve. We discuss the family specific properties and definitions in the following sections.

4.1.2.1 Finite Fields

Finite Fields were the original family of groups which were used for Diffie-Hellman schemes [70]. Let p be a prime number, then the multiplicative group of the finite field $(\mathbb{Z}/p\mathbb{Z})^*$ consists of the set of the non-zero elements, i.e. the integers (technically, equivalence classes of integers *mod* p) between 1 and $p-1$ and the group operation is multiplication *mod* p. This group has order $p-1$ and is hence even. From the Fundamental Theorem of Finite Abelian Groups (FTFAG) we know that the factors of $p-1$ are crucial in understanding the internal structure of the group. There have been two common choices. The first and most popular is that of picking p such that it is a safe prime, with $p = 2q + 1$ and q a prime number. This means $p - 1 = 2q$ and hence necessarily there are only two subgroups: one of order 2 and one of order q. The second choice is to use a DSA group[1] where $p = rq + 1$ for some large r, often much larger than q. Here the group structure is much more complex with many small subgroups.

Group elements have a very simple representation as integers in the set $1, \ldots, p-1$ and the group operation is defined over this entire range. However, typically a protocol intends to operate in the prime order subgroup and it is only possible to tell if a group element is also a member of this prime order subgroup with careful checks.

4.1.2.2 Elliptic Curves

Using Elliptic Curves for Diffie-Hellman groups began to grow in popularity over the past 10 years and have recently become the most popular family of Diffie-Hellman groups. Although they are significantly more complex to implement than finite field groups, they offer much improved performance, as well as much smaller key sizes. Elliptic curves are defined by a curve equation and a finite field of particular size. Group elements are given by the following definition:

[1] DSA (Digital Signature Algorithm) was standardised by NIST in 1994 and used such groups

Definition 38. [60, p. 13.1] Let K be a field and let $a_1, a_2, a_3, a_4, a_6 \in K$ be elements such that the discriminant of the polynomial given in the equation below is not zero. Then the set of points $E(K)$ on the elliptic curve $E = (a_1, a_2, a_3, a_4, a_6)$ are those which satisfy the equation:

$$\left\{(x, y) \in K^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\right\} \cup \{\mathcal{O}\}$$

For this work, we do not need the full details of the group operation, or how these parameter values are selected in this paper. It is possible to choose the order of an elliptic curve group with much more control than in the finite field case. Typically, the group order is chosen to be prime or nearly-prime.

Pairs of elements of K^2, written (x, y) which satisfy the above equation are considered to be points on $E(K)$, along with the distinguished identity element. Elliptic curve addition formulae are used to implement the group operation and typically operate on both the x and y coordinate. However, single coordinate ladders also exist [50, 131], which given an x coordinate and integer n can calculate x^n without needing the y component.

Every elliptic curve E over a field K has an associated quadratic *twist* E', which is another elliptic curve such that E and E' are isomorphic over the algebraic closure of K [60, §13.1.5]. We do not have the space to cover the full definition, proof of existence and uniqueness and other properties of the twist, see [60] for further details. However, we do need an important property, when K is a finite field [60, p. 13.17]:

$$\forall x \in K \ . \ \exists y \in K \ . \ (x, y) \in E(K) \vee (x, y) \in E'(K)$$

That is, for any x value, there is some y value such that (x, y) is on the curve or its twist. Of course, there are many more more pairs (x', y') which are not on either the curve or its twist.

4.1.2.3 Notation

There are two common notations for Diffie-Hellman groups. When referring to finite fields, typically multiplicative notation is used where we write the group operation as $\cdot$ and describe repeated applications as exponentiation. However, in the elliptic curve setting we use additive notation where the group operation is written $+$ and repeated applications are described as scalar multiplication. *To avoid ambiguity, we use multiplicative notation throughout this paper, regardless of the underlying structure.*

4.2 Symbolic Models of DH Groups

We will now examine how the state of the art in equational modelling of Diffie-Hellman groups has advanced, from the early work of the 1990s to tools still under active development.

4.2.1 Field Structure

In order to model an abelian group symbolically, it transpires that the exponents form a finite field. In particular notice that performing exponentiation, $(g^x)^y$ becomes multiplication in the exponents $g^{x \times y}$, whereas performing a group multiplication, $g^x \cdot g^y$, becomes addition in the exponents g^{x+y}. Note that we consider exponentiation (repeated multiplication) as a distinct operation from multiplication because of the hardness of inverting exponentiation in the groups used for public key cryptography. Algebraically, a field is described by:

Definition 39. A **field** is a set with the signature $\Sigma_F = \{\cdot^{(2)}, +^{(2)}, \div^{(1)}, -^{(1)}, 1^{(0)}, 0^{(0)}\}$ satisfying the following axioms:

$(a + b) + c = a + (b + c)$	Associativity of Addition
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$	Associativity of Multiplication
$a + b = b + a$	Commutativity of Addition
$a \cdot b = b \cdot a$	Commutativity of Multiplication
$a + 0 = a = 0 + a$	Additive Identity
$a \cdot 1 = a = 1 \cdot a$	Multiplicative Identity
$a + (-a) = 0 = (-a) + a$	Additive Inverse
$a \cdot (\div a) = 1 = (\div a) \cdot a$ if $a \neq 0$	Multiplicative Inverse
$a \cdot (b + c) = a \cdot b + a \cdot c$	Left Distributivity

So whilst the group itself only satisfies the axioms of an abelian group, this immediately induces the field structure on the exponents of elements in the group.

4.2.2 Early Tools

The symbolic model has its roots in model presented by Dolev and Yao in 1983 and sometimes is even given the name the “Dolev-Yao" model. After an initial buzz, interest faded as it was realised that most “interesting" classes of protocols were undecidable [160]. In 1996, Lowe generated a resurgence in the field that persists to the present day after finding an attack on the Needham Schroeder Protocol that went undiscovered for 17 years [116]. Lowe’s tool and contemporary approaches like NRL [120] and others [5, 59, 140, 153] did not support protocols which used Diffie-Hellman Groups.

The first research to introduce symbolic Diffie-Hellman groups appeared in the early 2000s. Between 2001 and 2003, Boreale and Buscemi published three papers on the symbolic model: [46, 47, 48] in which they introduce the first symbolic Diffie-Hellman groups. They model the inverses of terms and the inherent associativity and commutativity of the exponents, they do require the base to be fixed. A more serious limitation is that

they assume the number of products that can appear in any exponent has a fixed upper bound, which is certainly not true in practice. Furthermore, they do not provide a way to show this approximation is correct for any class of protocols.

In 2003, Chevalier et al. introduced a new decision procedure for verifying a limited class of protocols that use modular exponentiation [55]. They improved on Boreale and Buscemi's work by allowing an unlimited number of factors in their exponents and introducing support for arbitrary bases in exponentials. However, they do not allow the adversary to invert a known element although the protocol is allowed to perform these operations. As it is a decision procedure their approach is automatable, but they do not appear to have implemented their tool. Additionally, the limitations on the applicable protocols is very restrictive, for example no more than one variable can be introduced with each new message. There are also restrictions on what messages orders are allowed, for example a protocol sending the following sequence xy, x, y, appears to be forbidden. Goubault-Larrecq, Roger and Verma also raised the restriction on limited numbers of products, but don't consider inverses at all [87].

Millen and Shmatikov advanced the field further when their 2003 paper [129]. In their work, they model an abelian group structure carefully and allow the adversary access to its equational properties. Their protocols are broader than those of Chevalier. They also support exponentiation with a constant base and allow the adversary to use inverses. This work is a significant step forward because they identify the importance of unification in their work and use it to derive a decidability result in the bounded setting. There are some limitations to their approach, in particular protocols must be determinate (although indeterminate protocols can be made determinate by given the attacker the ability to control their non-determinate input).

Kapur, Narendran and Wang published an important theoretical contribution in 2003 [98]. In their paper, they consider several different equational theories that approximate modular exponentiation and investigate the unification problem for each. They also follow a clear separation of concerns, rather than attempt to provide an incremental improvement for symbolic model verifiers, they instead focus purely on the equational theory. They identify the target structure as an abelian group with a monoid and shows unification in this area is undecidable. Although this is the first time the result appears in the symbolic verification literature, this long been known in a more general context. An abelian group with a monoid is better known as a commutative ring and unification under this theory has been known to be undecidable since Hilbert's 10th problem was answered in the negative[2]. This result shows that supporting both group exponentiation and group multiplication leads to undecidable unification. In addition to this "no-go theorem", the paper provides a unification algorithm for the new theory which is described below.

[2]Discussed further in Section 6.2.1

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \quad \text{Associativity}$$
$$x \cdot y = y \cdot x \quad \text{Commutativity}$$
$$x \cdot 1 = x \quad \text{Identity}$$
$$x \cdot x^{-1} = 1 \quad \text{Inverses}$$
$$x^1 = 1 \quad \text{Exp1}$$
$$1^x = 1 \quad \text{Exp2}$$
$$(x \cdot y)^z = (x^z) \cdot (y^z) \quad \text{Exp3}$$
$$(x^y)^z = (x^z)^y \quad \text{Exp4}$$

In 2004, Lynch and Meadows considered an alternative approach [117]. They showed that introduced commutativity to the equational theory leads to an exponential blow up, that is unifying n variables can lead to 2^n most general unifiers. In the face of this considerable increase in complexity, the paper shows an approximation technique which avoids the use of commutativity, but is only sound for a limited class of protocols.

Whilst the other tools in this section perform verification with an unbounded number of sessions, the AVISPA toolset [13] considers a bounded number of sessions. In 2006, Turuani published the CL-ATse[3] tool [161], which can be used as part of the AVISPA toolset or individually. CL-Atse was primarily designed to support equational properties in a modular fashion, with a particular emphasis on XOR and Exponentiation. Although the support is not as flexible as the later Maude-NPA incarnation (CL-ATse requires specific unification algorithms to be provided for each disjoint theory which it then combines automatically), it a significant advance.

In 2007, Escobar et al. [80] provide a procedure (proven to be sound and complete although of course not necessarily terminating) that can handle any set of (user specified) terminating and confluent rewriting rules modulo associativity and commutativity[4]. They directly build on the work of Kapur et al [98], although they take a different route than Turuani to the same goal, by using narrowing they can avoid having to derive a new unification algorithm for each user specified theory and can instead build such an algorithm automatically. In particular, they model a Diffie-Hellman key exchange [71] and handle the associativity and commutativity correctly. In 2010, Sasse et al. followed up this work by applying it to a rich equational theory built out of XOR, asymmetric encryption and associative pairs [141]. Although they note their approach is less computationally efficient than writing a dedicated unification algorithm for each theory and combining them, the reduction in manual labour is significant.

[3] Constraint-Logic-based ATtack SEarcher
[4] They also require the finite variant property which is discussed further in [61]

4.2.3 Contemporary Tools

This now takes us up to the modern tools which support Diffie-Hellman models. There are four tools in widespread contemporary usage and we discuss each in turn.

4.2.3.1 ProVerif's Model

ProVerif includes a standard DH model, which has been extended in two different directions. Unfortunately, neither extension is compatible with the other, so we describe all three here.

Standard Model This model is included in the main ProVerif tool and is the most commonly used [42, Page 41].

$$g^{xy} = g^{yx}$$

g is a constant and the equation expresses the commutativity of exponentiation. By default, ProVerif uses a coarse model of Diffie-Hellman groups, modelling the bare minimum required to allow protocols to function, and considerably restricts the attacker's power.

There is no representation of *exp*'s associativity, inverses, or the identity element. This can lead to missed attacks. For example, if the adversary learns the term g^{xyz} and x, the adversary cannot deduce g^{yz}. Similarly, commutativity only applies to the base term. E.g:

$$g^{xyz} = g^{yxz} \neq g^{zxy} = g^{xzy}$$

Küsters and Truderung's Extension In 2009, Küsters and Truderung developed a technique to allow ProVerif to handle a more granular model of DH groups [107]. They provide a pre-processor that can take a protocol description with the granular model and transforms it into a syntactic theory where these equations do not need to be considered, which can directly then be analysed by ProVerif. They consider the equations:

$$g^{xy} = g^{yx}$$
$$g^{yy^{-1}} = g$$
$$x^{-1^{-1}} = x$$

Thus exponentiation is commutative, associative and has inverses, although not the identity element. Unlike Tamarin's approach, this approach only works for Horn clauses with ground exponents, which means that they cannot find certain attacks where the adversary sends products of exponents. The transformation additionally causes an increase in complexity of the model, and to the best of our knowledge, this approach has not been used in modern ProVerif analysis of protocols that use DH.

Bhargavan et al's Extension In 2015, Bhargavan, Delignat-Lavaud, and Pironti presented [31] the following model, built on top of ProVerif's standard model. Their intent is to approximate the presence of a "bad group" where small order elements exist. They introduce *gEl* and *bEl*, binary functions, which represent some base raised to exponent in either a 'good' or 'bad' group. They also introduce *gGen* a unary function which represents the generator of the 'good' group identified by its parameter. *bGen* is the corresponding generator of their 'bad' group which is the trivial group of one element.

$$\begin{aligned} gEl(gGen(id), x)^y &= gEl(gGen(id), x^y) \\ gEl(bGen, x)^y &= bEl(bGen) \\ bEl(bGen)^y &= bEl(bGen) \end{aligned}$$

The first equation defines normal exponentiation. The second and third collapse the result of a "bad" exponentiation (either in a bad group or a bad element in a good group) to a single element. In their paper, Bhargavan et al use this model for attack finding and automatically discovered unknown key share attacks on real world protocols. A similar but slightly simpler model was later used by Kobeissi and Bhargavan to show the necessity of certain validity checks in the Noise specification in [101]. However, these models are not suitable for verification or more nuanced analysis, as a considerable range of valid attacker strategies and protocol mitigations cannot be represented.

4.2.3.2 Tamarin's Model

In 2012, Schmidt et al introduced Tamarin in [144]. Tamarin is the first tool to combine verification with unbounded sessions, user defined equational theories, rich support for modular exponentiation and support for more advanced security properties[5]. Although the technique behind the tools equational support is very similar to that of Maude-NPA, significant optimisations in the prover and additional normal form conditions allow Tamarin to use a richer equational theory. In particular, handling how the adversary employs inverses requires careful attention from the designers of the tool. Their equational theory is:

[5]Security properties can be specified in a loosely restricted subset of temporal first order logic, which allows for complex properties involving dynamic compromise and other subtle notions

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$	Associativity
$x \cdot y = y \cdot x$	Commutativity
$x \cdot 1 = x$	Identity
$x \cdot x^{-1} = 1$	Inverses
$x^{-1^{-1}} = x$	Inverse Idempotence
$x^1 = 1$	Exp1
$g^{x^y} = g^{x \cdot y}$	Exp becomes multiplication

One of Tamarin's restrictions is that the protocol may not make use of multiplication of exponents directly, all operations must be described as exponentiations (which in turn typically reduce to multiplications). This means that modelling a protocol which transmits composite exponent values (e.g. xy) is not possible. Additionally, Tamarin has a number of optimisations required to make this approach feasible. Firstly, it is able to perform folding variant narrowing upon loading a protocol, this is a pre-processing step that generates the equational normal form for every protocol rule, which means that at run time, Tamarin only has to perform unification under Associative-Commutative (AC), rather than the richer theory. Secondly, a series of normal form conditions are enforced to prevent the adversary deducing the same information in multiple ways, for example, if the adversary knows the following values g^x, g^y then it can learn the term g^{xy} either by deducing g^x and then deducing g^{xy} or simply by performing $g^x y$ in a single step. The authors note that their support for equational theories is in principle the same as Maude-NPAs (indeed both tools rely upon Maude[6]). However Tamarin is significantly more likely to terminate when applied to protocols with complex equational theories.

4.2.4 CPSA

The Cryptographic Protocol Shapes Analyzer (CPSA) [73] introduced support for Diffie-Hellman equational theories in version 3.0 [115]. CPSA models the multiplicative structure of the exponents in the same style as Tamarin, omitting the additive structure. Unlike Tamarin, CPSA is able to reduce the number of cases it needs to consider by making use of the 'orthogonal' nature of randomly selected exponents. This optimisation comes at a cost however, CPSA excludes protocols which transmit composite exponents e.g. xy (like Tamarin) and also excludes protocols which produce exponents by construction from other terms (e.g. applying a hash function to an arbitrary term) unlike Tamarin. Support for this last behaviour is planned for the future.

[6]Maude is a system for term rewriting and unifying under an equational theory

4.2.5 Maude NPA

The Maude-NRL Protocol Analyzer (Maude-NPA) [79] uses a Diffie-Hellman theory of greater fidelity than ProVerif, but less granular than Tamarin or CPSA. The exponentiation operator reduces to a multiplicative operator which is both commutative and associative, but does not have an identity element or an inverse function. Additionally, Maude-NPA uses a sort system which separates Diffie-Hellman terms from other terms. This could cause some type confusion attacks to be missed.

4.2.6 Open Problems in Symbolic DH Models

Protocol analysis tools have significantly improved their support for Diffie-Hellman groups over the past two decades. However, there are two significant shortcomings which we mentioned in passing and now highlight.

4.2.6.1 Internal Group Structure

The majority of DH models do not allow for the presence of small subgroup elements or other aspects of internal group structure. The only exception is the model developed by Bhargavan, Delignat-Lavaud and Pironti in [31] as an extension to ProVerif which we discussed in Section 4.2.3.1. This models allows for attacks using small subgroup elements, but still omits a number of important behaviours such as the presence of inverses. Unfortunately, this model has not seen much uptake, with the only subsequent paper using it being [101].

In practice, there are considerable number of additional subtle behaviours in Diffie-Hellman groups such as elliptic curve point validation, equivalent group element representations and various mitigation strategies which are not captured by any of these models. Rephrased in a positive light, contemporary Diffie-Hellman models are reasonable approximations of prime order groups in protocols where elements are properly verified. However, there are many protocols and groups where these properties do not hold and consequently attacks may be missed by symbolic tools. We discuss this further and set out to remedy this in Chapter 5.

4.2.6.2 Direct Use of Group Operation

As Kapur, Narendran and Wang highlighted in [98] and we discussed in section 4.2.2 of this chapter, it has long been known that unification is undecidable if the symbolic DH model allows use of both exponentiation and the group operation itself. In all the tools we have discussed in this section, only exponentiation is supported (with varying degrees of fidelity). Representing the group operation on two elements is not possible in these models and consequently protocols which make use of the group operation cannot be described.

Interestingly, Liskov and Thayer showed in [114] that this limitation does not unduly restrict the adversary. That is, if a protocol and security property can be expressed without use of the group operation, then any attack on this protocol and property can be carried out without the use of the group operation. Consequently, we can be confident that attacks are not being missed due to the absence of this capability. However, if we were to describe a protocol which made use of the group operation, the adversary would also have to be given the capability. For example, Kaliski's attack on MQV in [97] requires the use of the group operation to succeed. We discuss this further in Chapter 6.

5

Modelling Small Subgroups and Elliptic Curves

This chapter is based on the paper:

Cas Cremers and Dennis Jackson. 'Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman'. In: *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*. 2019, pp. 78–93. DOI: 10.1109/CSF.2019.00013. URL: https://doi.org/10.1109/CSF.2019.00013

This paper won one of three 'distinguished paper' awards at CSF 2019. I was the lead researcher on the this paper, which was written in conjunction with my supervisor Cas Cremers. This chapter has been extended to include a discussion of testing subgroup leakage using hyperproperties in section 5.2.5.1 and further case studies in section 5.7.

The source code for our modifications to Tamarin, implemented attack on Scuttlebutt, case studies and models are available at [9, 65].

5.1 Introduction

In the previous chapter, we have explored existing symbolic models of Diffie-Hellman groups and their shortcomings. In this chapter, we develop a novel extension which accurately models the internal group structure of a broad class of groups used in cryptography. Furthermore, our extension also captures the additional behaviours introduced by elliptic curve groups and the mitigation available to protocol designers.

We show how to implement our model in the Tamarin prover and evaluate its effectiveness. We use our implementation to find a new attack on the Secure Scuttlebutt Gossip protocol, independently discover a recent attack on Tendermint's secure handshake and evaluate the effectiveness of the proposed mitigations for recent Bluetooth attacks.

5.2 Symbolically Modelling Group Structure

We start out by describing several behaviours of such groups, after which we present our model in detail in sections 5.2.2 to 5.2.5.

5.2.1 Behaviours of non-prime order groups

In the prime order case, by Lagrange's Theorem (§4.1.1), every element, other than the identity element, must have prime order and hence there cannot be any small subgroups. However, this is not necessarily true in the non-prime order case. A non-identity element may have an order lower than the order of the group and instead of generating the entire group, the element will generate a subgroup. For example, an element h with order 2 generates a subgroup of size 2, with $h^2 = id$, $h^3 = h$ and so on. Furthermore, *every* non-prime order group necessarily contains additional subgroups, due to Cauchy's Theorem (§4.1.1). We call elements which belong to at least one small subgroup, *small subgroup elements*. These small subgroup elements allow for a number of additional behaviours which are not found in prime order subgroups.

Firstly, *confinement*, when a small order element is exponentiated, the possible results are limited to other elements in the subgroup. Assuming the exponent was randomly sampled, this leads to a non-negligible likelihood of collisions where ${h_1}^x = {h_2}^y$ for h_i small order elements and x, y randomly sampled integers. In contrast, if the h_i are drawn from the prime order subgroup, collisions only happen with negligible probability. This can seriously impact the security of higher level protocols which rely on the 'contributory' nature of Diffie-Hellman operations as we will see in §5.6.

The second type of behaviour is that of *point mangling*, where given an element g^x in the prime order subgroup and h an element of low order, we combine them to get $h \cdot g^x$ an element in the supergroup. Although $h \cdot g^x$ is a distinct bit string from g^x, with non-negligible probability we may have $(h \cdot g^x)^z = g^{xz}$. This can impact protocols

implementing replay detection or other countermeasures, as they may expect public keys to be unique.

Finally, it is possible to exploit small order subgroups to perform *key leakage attacks*. In 1997, Lim and Lee [112] introduced this attack, permitting the recovery of secret exponents used in Diffie-Hellman operations. The crux of the technique is that the attacker submits an element outside the prime order subgroup to a protocol, which then uses the element in an exponentiation with a secret exponent. The attacker then observes the resulting behaviour of the protocol. If the attacker can deduce or confirm a guess of what the result of the exponentiation was, they can learn a limited amount of information about the secret exponent which was used. The attacker can then repeat this process and eventually combine the information in each guess using the Chinese Remainder Theorem and recover the secret exponent in its entirety. This attack has proven practical in many real world situations [123, 126, 162] including recent catastrophic key recovery attacks which impacted OpenSSL, the Exim mail server, as well as many other TLS, SSH and IKE implementations. With these behaviours in mind, we now set out to build symbolic models which can capture these behaviours.

5.2.2 Element Representation

Our first question is how to represent group elements in a more complex group. We note that because the number of operations to solve the discrete log problem in a group scales with the size of the largest prime factor of the group order, all Diffie-Hellman groups must have at least one large prime factor. Using the FTFAG (§4.1.1), which states that for any finite abelian group G of order n with a non-repeated large prime factor p, there is an isomorphism such that $G \cong H_1 \times H_2$ where $H_2 \cong \mathbb{Z}_p$ and H_1 is of order h such that $n = hp$. This means that any element in G can be expressed in the form:

$$h \cdot g^y \cong (h, g^y)$$

where on the right hand side we have (partially) decomposed the group. We will informally refer to H_1 as the cogroup of H_2. In the (cryptographically unusual) case that p is a repeated factor, the prime order subgroup may be contained in $\mathbb{Z}_{p^k}$ for some k, but this does not otherwise change our discussion.

We sketch the intuition behind this idea, noting that the underlying mathematics is well known: any finite abelian group has a finite set of generators. Any element can be written as a combination of some powers of these generators. This isomorphism is just between the 'natural' representation of the group, and the representation of the group in terms of an integer associated to each generator. Instead of performing the group operation on elements of the group, we may equivalently add the powers of each generator. Similarly, exponentiation distributes over each element in the tuple in the natural way.

This allows us to describe any element of a finite abelian group in one the following four forms:

- (id, id) is the identity element.
- (id, g^x) is a regular element of the prime order subgroup.
- (h, id) is an element in the cogroup.
- (h, g^x) is an element in the supergroup.

We will now use this decomposition to provide a general representation of group elements in the symbolic model. We define the function symbol $ele(t, s, n)$ with arity 3. The first argument t represents the identifier of the group, which will be useful when we consider protocols using multiple distinct groups later. s represents the cogroup component, n is the prime order component. We will elaborate on each of these components in the following three sections.

In the following, we differentiate between three different types of group. Firstly, in §5.2.3 we consider prime subgroups and make a modest extension to adapt traditional models to our new representation. Then in Section 5.2.4 we consider 'nearly-prime' groups where the order of the group n can be written as hp with p a large prime and h a small cofactor. Finally, in Section 5.2.5 we deal with the general case, where h may even be much larger than p.

5.2.3 Modelling Prime Order Groups

We now present a small extension to Tamarin's default model, which captures the identity element in a prime order group. Note, this is the identity element for the group operation, rather than multiplication in the exponent. The ProVerif extension from [31] provides a model suitable for the identity element but uses less accurate DH equations than Tamarin.

We can get the best of both worlds by extending Tamarin's equational theory for Diffie-Hellman groups defined in §4.2.3.2 with an additional constant gid and equation:

$$gid^x = gid$$

We combine this model with our new representation, and model a prime order group G as elements of the form:

$$ele('G', gid, X)$$

Here, ele is function symbol with arity 3, $'G'$ is a public constant representing the name of the group, s, the cogroup component is fixed as the identity element. Finally, X will be a 'traditional' Diffie-Hellman term using the exp operator discussed earlier in §4.2.3.2.

As we are working in the symbolic model, we must explicitly allow the adversary to extract information from elements. The first two parameters are public, so we need only allow the adversary to extract the third term:

```
functions: ele/3 extract/1
equations: extract_1(ele(t,s,n)) = n
```

Introducing this equation allows for non-contributive behaviour that was previously not captured in TAMARIN under the implicit assumption that protocols rejected the identity element as a valid point. Later, we will see how to restore the identity element check in situations where a protocol explicitly requests it. This model is the starting point for our work.

5.2.4 Modelling "Nearly-Prime" Order Groups

With this representation in mind, we now consider the case of 'nearly-prime' groups, before tackling the more general case in §5.2.5. We consider a group to have nearly-prime order if it can be written as $n = hp$ where h is a number such that $log_2(h) \ll log_2(p)$ and p is prime. That is, the additional factor h is very small compared to the order of the prime subgroup. This means the cogroup is of small size, leading to the confinement and point mangling behaviours from Section 5.2.1. However, due to its small size, the leakage attack of Lim and Lee is not of practical impact, leaking only a limited number of bits and we will not consider it for this type of group.

The question arises of how to model operations in the small order cogroup. Depending on the group in question this cogroup may contain further small order groups or have other internal structure. To provide a general model, we will abstract these details away by allowing the adversary to choose how this small subgroup operates. We justify this approach on two grounds. Firstly, we intend to give the adversary as much power as is reasonable to ensure our verification results are meaningful. Secondly, protocols are typically considered secure if they can be implemented with a wide class of groups, consequently, they do not and indeed cannot rely on the exact equations which hold in a specific small order subgroup. There are also various mitigation strategies that we will discuss later in §5.5.

We model this behaviour in our symbolic model by providing the adversary with a representation of these small order points and we provide a private channel by which the adversary may 'program' the result of exponentiation with a small order component. This allows the adversary to force collisions or inequalities according to their requirements, and consequently the adversary will always know the small order component of a group element term.

Consider the following example rule in TAMARIN's notation:

```
rule Operate:
  [In(X),State(y)]-->[Out(X^y)]
```

We perform the following steps:

1. Replace X^y with $ele(t, r, n^y)$.
2. Replace X with $ele(t, s, n)$.

3. Add the premise: $In(r)$.
4. Add the annotation: $Raised(t, s, r, y)$.

Thus the rule becomes:

```
rule Operate:
[In(ele(t,s,n)),State(y),In(r)]
--[Raised(t,s,r,y)]->
[Out(ele(t,r,n^y))]
```

Note that in practice, we perform this transformation automatically. This provides a private channel by which the adversary, using the message fact In, can determine the outcome of an operation in the small subgroup. In this case specifying that $s^y = r$.

However, an unrestricted private channel is too powerful. Firstly, the adversary is not required to operate the channel deterministically, meaning that repeating the same operation may lead to different results. Secondly, if the small subgroup component is the identity element gid, we know the result of exponentiating must be the identity element as $gid^x = gid$. We use the *Raised* label and Tamarin's restriction system to enforce these constraints:

$$\texttt{Consistency} : \forall t, s, r_1, r_2, y, \#i, \#j \ . \ Raised(t, s, r_1, y)@i \wedge Raised(t, s, r_2, y)@j \implies r_1 = r_2$$

$$\texttt{Identity} : \forall t, r, y, \#i \ . \ Raised(t, gid, r, y)@i \implies r = gid$$

We now explore the behaviour of this model: When the input is the identity: $ele('G', gid, gid)$, the result is necessarily also the identity. Similarly a prime order element $ele('G', gid, X)$ is also well behaved and will produce $ele('G', gid, X^y)$. Small subgroup elements are controlled by the attacker and consequently $ele('G', s, gid)$ will become $ele('G', z, gid)$ with z selected by the adversary, allowing for confinement and collisions. Finally supergroup elements $ele('G', s, X)$ are partially controlled by the adversary, as X^y is determined using Tamarin's prime order model, but control of the s term allows for point mangling.

5.2.5 Modelling Composite Groups

We now turn to more general groups, where we do not require any relationship between the prime order subgroup and the size of the supergroup. Whilst these groups are not typically considered desirable to work in from a security perspective, they have often been selected for reasons of performance. In addition to the behaviour found in 'nearly-prime' groups, these groups allow for the key leakage attack of Lim and Lee to be performed.

We build on our earlier model of nearly-prime groups. As we have already allowed the small order group in that case to grow to arbitrary size and the adversary to determine all operations in the small order group, the larger cogroup in this case requires no additional extension beyond the modelling of the information leak induced by the key leakage attack. We first consider the properties of this attack. There are a few fundamental requirements for this attack to succeed:

1. The same exponent must be reused in multiple calculations.
2. The attacker must be able to submit an element with a low order component.
3. The attacker must be able to confirm or invalidate a guess at the result. This may be an offline computation or online interaction, but even a single guess which can be confirmed or rejected suffices.
4. The supergroup must have enough small order factors to allow for the recovery of a significant fraction of the key.

Note that in the 'Nearly-Prime' setting, the low cofactor ensures this last condition is not satisfied. Three of these properties (1,2,4) are easily represented as trace properties as they are predicated on a particular sequence of events. However, the third condition is not a trace property but a hyper property as it requires reasoning about alternative possibilities. Whilst hyper properties can be explored in the symbolic model, support is considerably more limited than for trace properties.

Consequently, we first describe a trace property which captures conditions 1, 2, and 4. If this trace property holds, we can be sure that no small subgroup key leakage attack is possible on the protocol:

$$\begin{aligned}\neg\exists t_1, l_1, r_1, t_2, l_2, r_2, y, \#i, \#j.\\ \mathit{Raised}(t_1, l_1, r_1, y)@i \wedge \mathit{Raised}(t_2, l_2, r_2, y)@j \wedge\\ r_1 \neq r_2 \wedge l_1 \neq \mathit{gid} \wedge l_2 \neq \mathit{gid}\end{aligned}$$

If this property holds, then there is no trace satisfying all of conditions 1,2, and 4. The presence of two raised labels with the same exponent y ensures condition 1. Likewise, the final two clauses ensure condition 2 holds. Condition 4 is known in advance by selection of the group, if the protocol uses more than one group and only a subset of these groups are composite, then this lemma can be specialised, by further requiring that t_1, t_2 be equal to the types of the composite group. We discuss this further in Section 5.4.1.

However, if this trace property does not hold, the possibility of an attack depends on condition 3. Although a real-or-random indistinguishability test on subgroup elements is appropriate for testing this condition, and can be specified in the symbolic model, tool support for indistinguishability testing is still in its infancy. Consequently, although we developed a model which we describe in the next section, it was not tractable for real world protocols. Hence in our implementation we only test for the trace properties. However, we note all popular protocol mitigations of key leakage focus on preventing one of the first, second or fourth conditions and hence our test is unlikely to report false attacks in practice.

5.2.5.1 Testing for Information Leakage

We now describe property 3 in Tamarin as a indistinguishability game where subgroup elements are not generated by the adversary, but instead provided as tokens with hidden values from an oracle. The adversary has a number of capabilities:

- Any normal interaction with the protocol.
- Generate a random subgroup element (of unknown value).
- Submit a subgroup element to be the result the protocol performing an exponentiation.
- Reveal a subgroup element.
- Perform a real-or-random test on an unrevealed subgroup element.

Here the adversary interacts with two systems. In one system the real-or-random oracle always returns the true value of the subgroup element. In the other, a random element is returned. If these two systems are indistinguishable to the adversary, i.e., every trace that is possible in one system is possible in the other under the same conditions and vice versa, then we know that the adversary cannot confirm a correct guess of the small subgroup element and hence no small subgroup key leakage attack is possible. This is because if the two systems are equivalent, then the adversary cannot deduce any information or equations which hold for the real subgroup element, but do not hold for some random element. Alternatively, an attack trace distinguishing these systems will describe exactly how the adversary may confirm a correct guess.

Checking for the possibility of guessing or deducing low entropy values in the symbolic model requires careful finesse. We begin by hiding the value of subgroup elements from the adversary. That is, the adversary may still choose the results of computations by selecting a label, but does not know what the underlying value is that it has selected for a particular result. For any particular underlying the value, the adversary may choose either to reveal it, learning the value, or it may select that value for a real-or-random test. That is, the adversary will 'win' by being able to determine whether it was given the real value was that was used in the exponentiation, or whether it was given a random value which was unrelated. If the adversary cannot distinguish the real result from an unrelated one, we can be sure that condition 3 holds. We allow the adversary to continue interacting with the protocol after performing the real-or-random test, so the adversary may calculate some term using the revealed value and test how the protocol responds.

We alter the earlier rules concerning the production of *ele* facts to not reveal the value of the element. Instead, we provide a Fact token which the adversary can consume (once only) to either reveal the true value or a challenge value (which may be the true value or a random alternative).

More formally, we alter the production rules so that instead of revealing the representative, they create a token instead:

```
rule Make_Identity_Element:
[]-->[!Subgroup('G',gid)]

rule Make_Hidden_Element:
[Fr(~x),In($GRP)]-->
[Token(~x),!Subgroup('G',~x)]
```

Then we provide the challenge and reveal rules:

```
rule Reveal:
[Token(~x)-->[Out(~x)]

rule Challenge:
[Token(~x),Fr(~y)]--[OnceOnly('Challenge')]-->[Out(diff(~x,~y))]
```

Here, *diff* is a special kind of term. TAMARIN will generate two systems from this a rule, a left and right system, with the *diff* term being replaced by the first argument for the left system (respectively second, right). These two systems are then tested for observational equivalence. For the full explanation of how this process works and the precise definition of observational equivalence, see [24].

We illustrate our model with a simple example. Consider the following one rule protocol which uses a hash function h in the prime order case:

```
[In(x),Secret(~y)]-->[Out(h(x^~y))]
```

After transformation to the composite group model, this rule will become:

```
[In(element(t,l,n)),!Secret(~y),!Subgroup(t,r)]
--[Raised(t,l,r,~y)]-->
[Out(h(element(t,r,n^~y)))]
```

When we test this system under our model for small subgroup key leakage, we first check the trace base property. Clearly, it will not hold, as the adversary may trigger this rule twice with different subgroup elements. We then consider our indistinguishability property. The adversary can 'win' the real-or-random test with the following attack. Firstly, the adversary creates a two subgroup elements, reveals one, and submits it to the protocol. Then, submits the second, unrevealed element, to the protocol rule as the result of the exponentiation. The protocol then produces the resulting hash value.

The adversary can now perform the real-or-random test on this second value. In the left hand system, the adversary is given the true value, which allows the adversary to independently compute the hash of the element and check it for equality with the protocol's output. However, on the right hand system, the protocol learns a random value and when it is hashed, will not match the protocol's output. This violates the observational equivalence property. Similarly, if the protocol did not output a hash value, but instead expected to receive a correct hash value from the adversary before proceeding, this would also lead to an attack, as only the adversary in the left hand system would be able to trigger this behaviour from the protocol.

However, when we tested this model against real world case studies using Tamarin, it proved to be intractable for Tamarin's automatic heuristics. Unfortunately, Tamarin's support for equivalence testing is less mature than its support for trace based properties and we are hopeful that this approach will become more tractable in the future.

5.3 Symbolically Modelling Elliptic Curve Points

We now turn from our discussion of internal group structure to the representation of elliptic curve elements in the symbolic model. Whilst elliptic curves have many advantages over traditional finite field DH groups, their complex representations require more careful management.

In the finite field setting, group elements are integers in the range 1 to $p-1$ for some large prime p. Although protocols will typically intend to operate within a prime order subgroup inside this range, the group operation is well defined over the entire range of elements. However, in the elliptic curve setting, group elements consist of a pair of finite field elements, that satisfy the curve equation. This is discussed in more detail in Section 4.1.2.2. For points in the finite field that do not satisfy the curve equation, the group operation is not well defined and performing operations on these invalid points can lead to catastrophic outcomes [37, 68, 94].

In the following, we provide symbolic models for the additional behaviour that elliptic curve operations can exhibit when operating on invalid points. To aid modellers in choosing the right symbolic model for a particular curve, we provide a list of common curves and their properties in Section 5.4.

We first consider a special case, when the curve operation is implemented using a single coordinate ladder, before providing a general model suitable for more traditional implementations. Our models build on and extend those presented in the previous section.

5.3.1 Single Coordinate Ladders

Whilst elliptic curve points are traditionally represented by both an x coordinate and a y coordinate, both coordinates are not necessarily needed for cryptographic purposes. Using the curve equation (see Section 4.1.2.2), it is possible to uniquely identify a point using its x coordinate and a single bit. This is because the curve equation can be solved to give two possible solutions for y (corresponding to the positive and negative square roots) and the single bit can be used to select the correct solution. Consequently, using both the x and y coordinate only offers one additional bit of entropy over the x coordinate alone.

Furthermore, there exist special techniques for performing exponentiation (scalar multiplication) on curve elements that only use the x coordinate. Whilst these techniques do not support more general uses of elliptic curves, such as digital signature schemes, they have become popular for Diffie-Hellman operations. These techniques are known as single coordinate ladders and were first introduced by Montgomery in 1987 [131] for a special class of curves. Later Brier and Joye [50] provided a ladder suitable for any elliptic curve. In addition to requiring less bandwidth, these ladders are also easy to implement in constant time and highly performant, leading to widespread adoption.

We now consider how these ladders behave with x coordinates which are not on the correct curve. As is discussed further in Section 4.1.2.2, every elliptic curve E over a finite field is uniquely associated with another elliptic curve E', known as its quadratic twist, or twist for short. For any x coordinate in a finite field, it so happens that there exists a y value such that (x, y) either lies on the intended curve, or its twist. That is, every x coordinate can be said to lie on the curve, or its twist (or both). Furthermore, a single coordinate ladder is 'well-behaved' for both a curve and its twist. Thus for any x coordinate, the ladder will calculate the group operation either on the curve, or its twist.

This has several important consequences. Firstly, modern curves have been designed to be *twist secure* [62], where performing a discrete log on the twist is also intractable. This is intended to allow for single coordinate ladders to be used without having to ensure an x coordinate lies on the correct curve. This reduces implementation complexity and improves performance. However, older curves do not necessarily have this property. For example, calculating discrete logarithms on the twist of the NIST prime order curve P-224 takes roughly 2^{58} operations, compared to 2^{111} operations on P-224 itself [27]. Another complication is that whilst some curves may have prime order and hence no small subgroup elements, their twists may only be nearly-prime or even composite, further complicating the use of ladders on these curves.

As the properties of the twist depend on the specific curve used, we provide a model for each class of twist. Firstly, the twist may be represented by a prime order group, a nearly-prime group or a composite group as described in the previous section. We will also introduce an additional type of group, where discrete logarithms are easy, to cover curves which do not have twist security.

We will differentiate between elements on the main curve and elements on the twist using the first parameter of $ele(t, s, n)$. Previously, we left this value unconstrained and uninterpreted. Now, we will restrict to either being $'C'$, to indicate an element on the curve or $'T'$ to indicate an element on the twist.

We now specialise protocol rules that perform exponentiation into two constrained variants. One rule variant will handle elements on the curve, and the other will handle elements on the twist. We return to our earlier example from section 5.2.4 on page 63 and transform it for the single coordinate case:

```
rule Operate_Normal:
[In(ele('C',s,n)),State(y),In(r)]
--[Raised('C',s,r,y)]->
[Out(ele('C',r,n^y))]

rule Operate_Twist:
[In(ele('T',s,n)),State(y),In(r)]
--[Raised('T',s,r,y)]->
[Out(ele('T',r,n^y))]
```

We can now further restrict each rule, depending on the properties of the curve and its twist. For example, if one should be of prime order, we will set s to be *gid* on all rules for that curve. The situation is similar for testing for subgroup key leakage. If the twist

has composite order, but not the curve, we can specialise the key leakage trace property to require that the first parameter of *Raised* be equal to $'T'$, ensuring only twist attacks are captured. This duplication of rules will also come in useful later when we consider protocol level mitigations in §5.5. Note that in the special case where the curve and its twist have exactly the same properties and the protocol applies its mitigations correctly to both curves, we can omit this duplication of rules.

There is one additional behaviour that may be possible on a curve's twist: the discrete log problem may not be intractable. We capture this situation by providing the adversary with an oracle they can query to take the discrete log of a twist element. We first extend every Twist variant to also produce the token $TRes(n^y, y)$ and provide the rule:

```
rule Twist_Discrete_Log:
  [In(X^y),TRes(X^y,y)]-->[Out(y)]
```

This rule requires the adversary to have learned the resulting DH element directly before it can calculate its discrete log. In general, learning a non-invertible function of a DH element does not allow an attacker to calculate its discrete logarithm.

5.3.2 General Invalid Curve Points

We now consider the more general setting, where points are represented as both a x and a y value. This is often the case in protocols intended for widespread adoption (such as TLS) where there may be many interoperating implementations. Unlike in the single coordinate case, the attacker is not restricted to operating on the curve or its twist, meaning twist security has little relevance in this setting.

In 2000, Biehl, Meyer and Müller [36] were the first to investigate invalid curve point attacks. By carefully selecting points, the group operation can be forced to operate in a series of different curves, each with low order points. This allows the small subgroup key leakage attack of Lim and Lee to be performed (as described in §5.2.5). In 2003, Antipa et al. [8] extended this work and calculated approximate running times and the number of interactions for this style of key leakage attack. More recently, Neves and Tibouchi [132] introduced a new technique for finding useful invalid curve points on Edwards Curves. Not only does the attack find points of low order, it can also move the computation to curves where the discrete log is easy.

In summary, there are a number of general and powerful techniques for selecting invalid curve points, which allow for the behaviours we have discussed earlier: subgroup confinement, subgroup key leakage and easy discrete logs.

We now extend our earlier model of Diffie-Hellman elements to represent these special behaviours in the elliptic curve setting. We will represent the elliptic curve, according to its type, as either prime, nearly-prime or composite order. Then we provide a new construction capturing invalid points.

We represent elliptic curve points directly as the combination of two elements using TAMARIN's pairing operator: $\langle x, y \rangle$ where $x, y = ele(t, s, n)$. We then distribute exponentiation over them in the natural way. As in the earlier twist case, we will duplicate and specialise the rules which perform exponentiation.

For the first rule, we will add the constraint that $x = y$, this represents operation on a valid curve point where x and y satisfy the curve equation. For a valid point, each coordinate can be recovered from the other with only one additional bit of information, so it is reasonable to represent them as equivalent terms. Additionally, we require the first parameter of each element to be $'C'$ to reflect the well formed nature. In the second case, we will add the constraint that $x \neq y$. In this case the point is invalid and we will allow the adversary to substitute in a point of their choice.

We transform the invalid exponentiation rule by:

1. Add a Premise $In(\langle t_x, r_x, n_x, t_y, r_y, n_y \rangle)$, representing the attacker providing a substitute point
2. Write the result as $\langle ele(t_x, r_x, {n_x}^z), ele(t_y, r_y, {n_y}^z) \rangle$
3. Add the annotation $Raised(\langle t_x, t_y \rangle, \langle s_x, s_y \rangle, \langle r_x, r_y \rangle, z)$
4. Add the conclusion $IRes(ele(t_x, r_x, {n_x}^z), z)$
5. Add the conclusion $IRes(ele(t_y, r_y, {n_y}^z), z)$

And finally we have the rule for taking discrete logs on invalid elliptic curves:

```
rule Invalid_Discrete_Log:
[In(X^z),IRes(X^z,z)]-->[Out(z)]
```

This model is highly flexible and describes the behaviour of invalid points. It uses elements of the models from all previous sections. It also allows the protocol to split up x and y points according to how these distinct values might be used in practice. Note that in this case, a small subgroup key leakage attack is always potentially possible using invalid curve points, so the leakage lemma from §5.2.5 should always be added. As in the twist case, we can use pattern matching to ensure regular curve elements are not the subject of false attacks by requiring the lemma to pattern match on the type of the element ($\langle t_x, t_y \rangle$).

5.4 Choosing a Symbolic Model

In this section, we clarify how a particular model should be selected and employed. Our extensions exist along two axes:

- Internal Group Structure
  - Prime Order Groups (§5.2.3)
  - Nearly-Prime Order Groups (§5.2.4)
  - Composite Groups (§5.2.5)
- Group Elements

- Finite Field Elements (no additional behaviour)
- Elliptic Curve Elements
  - Single Coordinate Ladders (§5.3.1)
  - General Coordinates (§5.3.2)

These models do not stand in isolation, each subsequent model extends the previous model to allow for more behaviour in a particular setting. Furthermore, we allow a wide range of mitigations to be employed by the protocol, which are described in §5.5. Our models also support the use of multiple groups with distinct properties, we discuss this in §5.4.1. We now explore how to select the appropriate symbolic model for a particular real world group.

Firstly, in the finite field setting, the only relevant parameter is the group order. If it is a safe prime, e.g. $p = 2q + 1$ with q and p prime, then the group order is $2q$ and there is only a single subgroup of order 2. Consequently, the nearly-prime model is appropriate. If a DSA or Schnorr group is used, the order will have many small factors and the composite model should be used. If unsure, selecting the composite model is the most general and hence appropriate. Note there are no finite field groups with prime order, as p is prime, $p - 1$ is necessarily even and hence there is always at least the subgroup of order 2.

In the elliptic curve section, the picture is more complicated. The curve itself may have prime or nearly-prime order [1]. Then it depends on the coordinate system in use. In the case of a single coordinate ladder, the twist will necessarily be either prime, nearly-prime or composite. Additionally the twist may or may not be twist secure. Alternatively in the general coordinate system, the parameters of the twist do not matter. If the exact elliptic curve is unknown, the most general model is a nearly-prime curve with general coordinates. We provide a table of common elliptic curves and their twist's parameters in Table 5.1. Note that Curve25519 is more tightly specified than other curves, it always applies private key clamping to eliminate low order components and is always implemented using a single coordinate ladder.

5.4.1 Modelling Multiple Groups

We now briefly explain how to model combinations of different groups with different orders in one protocol. When a protocol uses multiple groups, it may behave differently when it receives an element of a particular group. For example, it may apply particular mitigation strategies or run certain sub protocols.

Elements of two different groups are distinguished in our framework by the first parameter. For example, $ele('G1', s_1, n_1)$ represents an element of G_1. A protocol can restrict a particular operation by pattern matching on this first parameter. Similarly, this allows us to capture the properties of different groups. Branching in Tamarin is typically

[1] Composite order curves are possible but not used in practice

Curve	Order	Twist Order	Twist Secure
NIST P-224	Prime	Composite	×
NIST P-256	Prime	Nearly-Prime	✓
NIST P-384	Prime	Prime	✓
Curve25519	Nearly-Prime	Nearly-Prime	✓
Ed448	Nearly-Prime	Nearly-Prime	✓
secp256k1	Prime	Nearly-Prime	✓
BN(2,254)	Prime	Composite	×
brainpoolP256t1	Prime	Composite	×
ANSSI FRP256v1	Prime	Composite	×

Table 5.1: Common elliptic curves and their properties [27, 81]

represented by having multiple rules, pattern matching or applying conditions on each rule to represent the specific conditions for that branch.

Firstly, prime order group elements should always pattern match the second parameter to *gid*, ensuring no small subgroup behaviour can apply. Secondly, when considering a composite group and a non-composite group, the trace based key leakage property can be restricted to require t_1, t_2 to match a particular group identity.

Elliptic curve groups can be managed in the same fashion, by following the transformations for individual rules given in each section, it is possible to model protocols where some entities use differing coordinate systems.

5.5 Modelling Mitigations

There are several countermeasures and mitigation strategies which can be employed by protocol designers to restrict or remove the behaviours we have discussed in this paper. Often, these countermeasures have been added to implemented protocols in an ad-hoc fashion, leading to uncertainty about their suitability and effectiveness. As well as discussing these mitigations and their corresponding model, we also consider the recommendations of various authorities as to which mitigations should be employed and under what circumstances they suffice. We begin with the more traditional checks:

5.5.1 Rejecting the Identity Element

Every group contains an identity element, which is a fixed point under exponentiation. In some implementations this point is not expressible in the chosen coordinate system; others may choose to manually catch and reject this value before operating on it.

In our model, the identity element is represented by $ele(t, gid, gid)$ in the single coordinate case and $\langle ele(t, gid, gid), ele(t, gid, gid)\rangle$ in the general model of elliptic curves.

Consequently, it suffices to check that at least one component is not *gid* to rule out the identity element.

So we provide the label: $NotID(P)$, where $P = ele(t, gid, gid)$ and the restriction:

$$\forall t, \#i. NotID(ele(t, s, n))@i \implies s \neq gid \vee n \neq gid$$

In the general model of elliptic curves, we generalise this axiom to handle both the x and y coordinate in the natural way. This check may be used in a protocol before or after exponentiation.

5.5.2 Excluding Low Order Points

In nearly-prime groups, there are a small number of elements which have low order. Excluding these points, by matching against a list, is intended to ensure 'contributory' behaviour when exponentiating points. For some elliptic curves, notably Curve25519 [28], this countermeasure has proven controversial; leading to a number of discussions between designers and implementers.

Daniel Bernstein, designer of Curve25519, recommends against excluding low order points, as this only needs to be done for 'exotic' protocols [26]. Similarly, Trevor Perrin, co-designer of Signal, recommends against the check as 'safe protocols' should not require it [136]. On the other hand, some security researchers advocate [76, 152] for including the check and the IETF has opted to allow it [77]. This has lead to a similar split in implementation behaviour with LibSodium choosing to reject low order points [111], whilst other implementations typically accept them. We will see the impact of these decisions in a case study in §5.6.

We represent these checks in our model by providing a label $NotLowOrder(P)$ where $P = ele(t, s, n)$ and the restriction:

$$\forall s, n, \#i. NotLowOrder(s, n)@i \implies n \neq gid$$

This check is not always practical, for example in composite groups there may be a huge number of low order points, which would make these exclusion checks impractical. Similarly, this check does not ensure elements belong to the intended prime order subgroup, they may still have high order and belong to the larger supergroup. Consequently, this technique will prevent confinement, but not necessarily leakage.

5.5.3 Checking Element Order

This is a more thorough, but more computationally expensive, version of the previous check. As discussed in Section 4.1, the order of an element x is the smallest natural number k such that $x^k = gid$. When the order of a group is equal to hp, p is prime and h and p are coprime, then given an element in the group, checking it belongs to the order p

subgroup can be done by ensuring $x^p = 1$. If the protocol *assumes* x belongs to the group, rather than *checking* it is a valid representative, the order check is not well defined. In prime order groups, it is common for this check to be replaced by checking that the group element is not the identity element, as this is much more efficient and all other elements have the same order.

We model this behaviour by ensuring that if an element is within the group, we correctly confine it to the prime order subgroup and otherwise do not impose a restriction. We provide the label: $OrdChk(P)$ and the restriction:

$$\forall t, s, n, \#i \ . \ OrdChk(ele(t, s, n))@i \implies s = gid$$

In the general elliptic curve setting, this becomes:

$$\begin{aligned}&\forall t_x, s_x, n_x, t_y, s_y, n_y, \#i \ . \\ &OrdChk(\langle ele(t_x, s_x, n_x), ele(t_y, s_y, n_y)\rangle)@i \implies \\ &t_x \neq t_y \vee s_x \neq s_y \vee n_x \neq n_y \vee (s_x = gid \wedge s_y = gid)\end{aligned}$$

This states that either the point is invalid, or it is an element of the prime order subgroup. Note, the identity element is allowed, as $gid^p = gid$.

5.5.4 Low Order Clearing

There are several techniques which allow the protocol or the implementation to "zero out" the low order component. Let n be the order of the supergroup, with p the order of the prime subgroup and h the cofactor such that $n = ph$. Then if p and h are coprime, for any element in $g \in G$ we have that g^h will belong to the prime order subgroup. For example, if g is a small order element, $g^h = gid$, which (also) belongs to the prime order subgroup.

This prevents information leaks, as an adversary can no longer deduce any information from the small subgroup component. However, this can exacerbate subgroup confinement problems, as all low order elements are confined to the same output value, the identity element.

This technique is known by a number of names and can be implemented in different ways. For example, in Curve25519, this is know as private key clamping and any private key is manipulated by the implementation to ensure it is a multiple of the cofactor [28]. Similarly, protocols can use this mitigation by raising elements to the power of the group cofactor before using them. This does also change the high order component of an element and consequently all implementations of a protocol must agree on whether to use this mitigation.

We model this by providing the label $ClearPoint(P)$. We substitute $P = ele(t, s, n)$ with $P' = ele(t, gid, n^{'h'})$. $'h'$ is a public constant which represents the change to the prime order component and we assume is already known to the adversary.

5.5.5 Checking the Curve Equation

When dealing with elliptic curve points, it is important to ensure they belong to the correct curve before operating on them, as discussed in §5.3. Even if using a single coordinate ladder, this is an important step when the curve is not twist secure.

In the single coordinate case, we provide the label $NoTwist(X)$ and the restriction:

$$\forall t, s, n, \#i \ . \ NoTwist(ele(t, s, n))@i \implies t \neq 'T'$$

This rejects any point on the twist. Note that if the curve check is applied immediately prior to using the point in an exponentiation, it is equivalent (and more performant in Tamarin) to simply delete the rule variant for the twist. However, this check might be made at a different point in the protocol and hence we provide this restriction.

In our general model of elliptic curves, we model this check by using the label $Eq(x, y)$ which enforces the equality (and in our model validity) of the curve point. If this check is performed in the same protocol rule as the exponentiation, we can simply omit the generation of the invalid point variant as it is trivially impossible.

5.5.6 Point Compression

Another common technique employed in elliptic curve protocol design is that of point compression. This technique is applied in the general coordinates setting. Rather than transmit the full x and y coordinate, a single bit is transmitted in place of the y coordinate. This acts as a sign bit and using the curve equation, the sign bit and the x coordinate, it is possible to recover the full y coordinate.

This technique is advantageous as it reduces bandwidth costs and ensures recipients of curve points must employ the curve equation to recover the y value, rather than mistakenly accepting any y value. However, the attacker can still choose the sign bit, which allows the attacker to select either a curve point or a point on the twist. Therefore point compression is described in our model by using the twist model for a particular curve rather than the general coordinates.

5.5.7 Ristretto

Ristretto is a recently introduced technique and draft standard [159] which provides a computationally efficient transformation from a non-prime order group to a prime order group. This provides an abstraction layer on top of the standard group operations, which if used correctly, ensures the underlying group behaves as a prime order group. Ristretto provides two functions to encode and decode points, with the intention that encode is used on all 'output' points an decode is used on all incoming points. Additionally an equality and hash-to-curve function is provided.

We model Ristretto's four functions separately. For encode, we provide a function *encode* with the associated equation:

$$encode(element(X, Y, Z)) = element('R', gid, Z)$$

This ensures the resulting element behaves as a prime order element and is in the Ristretto group. We also model the decode function. In the symbolic model, it is best captured as the identity function, which may fail. In Tamarin's description, we capture this using a action label with an associated restriction.

$$\forall XYZi.Decode(element(X, Y, Z))@i ==> X =' R' \wedge Y = gid$$

Ristretto also exposes an equality function and a hash-to-curve function. In our model the equality function is the usual equality operator in the equational theory. The hash-to-curve function can be modelled with the following equation:

$$hash(X) = element('R', gid,' g'^{h}(X))$$

where h is a function not satisfying any equations.

5.5.8 Summary

These mitigations cover the typical types of mitigation available to protocol designers and recommended by standards bodies. For example, NIST's guidance [21, §5.6.2.3.3] recommends an elaborate sequence of checks and defence in depth measures when exponentiating the point Q:

- Ensure Q is not the identity element (§5.5.1)
- Verify Q satisfies the curve equation. (§5.5.5)
- Perform a full order check (§5.5.3)
- Raising by the cofactor during exponentiation (§5.5.4)
- Rejecting the result if it is the identity element (§5.5.1)

Contrastingly, Curve25519 implementations will always raise elements by the cofactor, and may optionally reject low order points, but do not perform any other checks. This is in part because the single coordinate ladder and twist security allows for some checks to be omitted and in part because small subgroup confinement is not considered problematic by the designers.

It is important to note that some protocol designs (e.g. Bluetooth) offer a number of different mitigations which implementers can pick and choose from. This choice can be modelled directly in Tamarin by providing multiple copies of each rule with a different

mitigation present in each copy. This allows the adversary the full range of possible behaviour. More complex designs can also be captured with the axiom system, for example to specify that any particular agent may only use one particular implementation and so on. In this fashion the entire design can be verified without missing cases.

5.6 New Case Studies

Over the proceeding sections we have built a family of symbolic models which capture the behaviour of different types of DH groups. We discussed (5.4) how to select a particular model for a given group or curve, as well as advice on how to proceed if the group or curve is unknown. We now evaluate our models on three real world case studies.

5.6.1 Scuttlebutt

Secure Scuttlebutt is a peer-to-peer gossip protocol for the distributed web [150]. Users run their own Secure Scuttlebutt endpoint locally and connect to their contact's endpoints to exchange messages and synchronise state. There are several interoperating clients including Javascript, Go, Rust, and C implementations [148].

Here we focus on a specific component of Secure Scuttlebutt dubbed the *secret-handshake*, which is used to authenticate and encrypt connections between peers. This handshake is of critical importance to the higher level protocol, as application data is generally otherwise unencrypted. The Secret Handshake was described in an early whitepaper [155] and wire specification [146]. The design draws on lessons from the older Station-to-Station protocol and the more recent Noise Protocol framework and aims to ensure not only basic security requirements but also perfect forward secrecy, resistance to unknown key share attacks, and key compromise impersonation. It also aims for a less typical authorisation property: the secret handshake is intended to keep the responder's *public* key secret and only allow connections from initiators who already know the correct public key. This mechanism forms the basis of Scuttlebutt's invitation system, where peers can 'friend' each other by exchanging their public keys. The handshake was originally modelled in Tamarin in February 2018, using the traditional model of Diffie-Hellman groups [149].

The secret handshake is depicted in Figure 5.1. i, r are the private signing keys of the initiator and responder, and g^i, g^r are the corresponding public keys. There are two distinct phases. Firstly, each party exchanges an ephemeral Curve25519 public key and proves knowledge of C, which is a constant used to identify the Scuttlebutt Network. The initiator then derives a key K_1 using the ephemeral values and the responder's public key and uses it to transmit both their own public key and an Ed25519 signature 'proving' they intend to talk to the responder. The responder can decrypt this message, learn the initiator's public key g^i and then prove control of their own public key g^r. The resulting

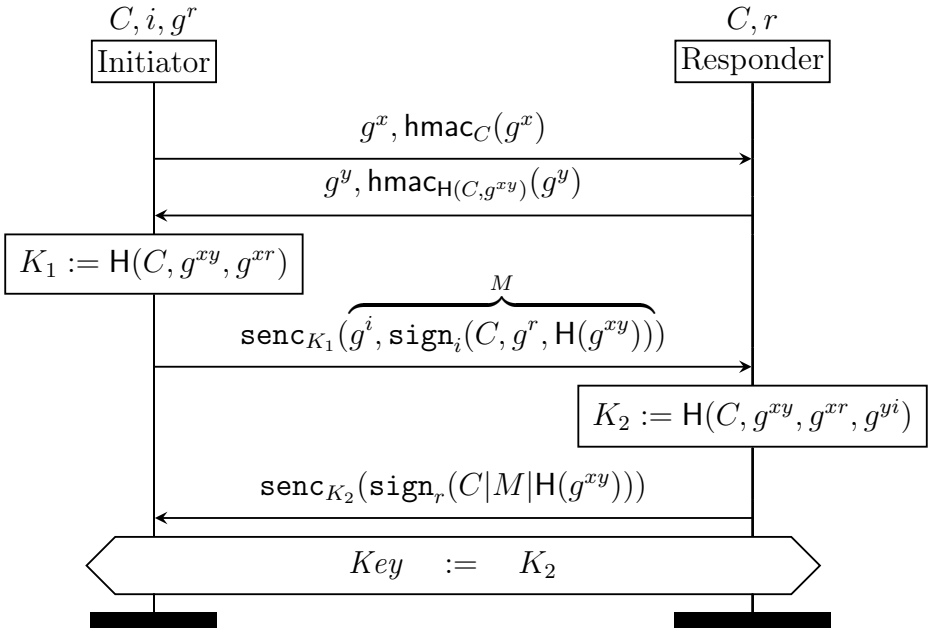


Figure 5.1: Secure Scuttlebutt's 'Secret Handshake' Protocol. x and y are freshly generated values.

key is then used to encrypt and authenticate further messages used in the higher level gossip protocols. An important authentication property is that *only an initiator who already knows the responder's public key, should be able to complete a handshake*. We modelled this protocol in Tamarin, using our 'nearly-prime single coordinate elliptic curve model with nearly-prime twist' model of Curve25519.

Tamarin automatically found a novel, previously unreported attack on this protocol which violates this authentication property. That is, an adversary with no knowledge of the responder's public key is able to complete a handshake as an initiator and knows the final key. This attack makes use of the small subgroup properties we have developed in this paper, and proceeds as follows. The attacker, as an initiator, selects their public key and ephemeral key to both be points of low order on Curve25519. The resulting shared key is constant and hence the adversary is able to derive the first ciphertext. However, the adversary must now produce a signature on the responder's public key, without knowing the responder's public key. Perhaps counter intuitively, this is possible and in fact, the method of producing such a signature is described in the original Ed25519 design paper [29, Page 7]. This additional signature property has recently been added to Tamarin [91].

As this attack makes use of two unusual properties of Curve25519 keys, one might assume that this attack is only theoretical. However, we successfully implemented the attack against the Go implementation of the 'Secret Handshake' [147], confirming the accuracy of our analysis.

Using this attack, an adversary could compromise the privacy of every Secure Scuttlebutt user, by methodically scanning the Internet for clients running Secure Scuttlebutt

and then connecting to them using our attack. These clients would then gossip their private state to the attacker.

Consequently, we have disclosed our findings to the Secure Scuttlebutt team and have recommended two different mitigations. Firstly, Secure Scuttlebutt clients could be configured to reject low order points. This ensures the attacker cannot derive the symmetric key without knowing the responder's public key and does not require a protocol change, as honest clients will never send such malformed public keys. Our alternative suggestion is to include the initiator's and responder's public key when deriving K_1 and K_2. Whilst this second suggestion requires a new version of the protocol and risks incompatibility with older clients, it ensures implementations will not accidentally (and silently) forget the low order point check. These two suggestions mirror the recommendations of protocol and primitive designers in §5.5.2 - it is possible to fix contributivity problems at either the primitive or the protocol level. We also provide models of our two suggested fixes, which Tamarin automatically proves correct.

The Scuttlebutt team responded promptly and acknowledged the security issue. Interestingly, it transpired that even though all Scuttebutt implementations use the same NaCl API to handle their Curve25519 operations, not all Scuttlebutt implementations were vulnerable to our attack.

Through a joint investigation with the Scuttlebutt team, we discovered that libraries offering NaCl support have unknowingly diverged and handle low order points differently. LibSodium [110], one of the most popular libraries, rejects low order points before they are operated on. However, HACL [169] - a formally verified library advertising itself as a "drop in" replacement for LibSodium - omits this check, as does the NaCl implementation in Go's standard library [85] and Cloudflare's CIRCL library [58]. Consequently, a protocol relying on contributivity, implemented using the NaCl API, may or may not be vulnerable depending on the underlying library.

Our work has lead to several changes. The Scuttlebutt team opted for our first proposed fix, as backwards compatibility is an important real world concern, and they have updated the vulnerable Scuttlebutt clients to ensure low order points are rejected. We also raised the omitted check with the maintainers of Go's NaCl API , Cloudflare's CIRCL and the HACL library, and they have now added this check or committed to adding it in future releases.

5.6.2 Bluetooth Handshakes

The Bluetooth standard [88] has a long and convoluted history of overlapping standards, design flaws and vulnerable implementations. One of its core security goals is to establish a secure channel between two devices, without relying on complex configuration or prior exchange of secret information.

We focus on two handshakes defined in the Bluetooth specification: Secure Simple Pairing (SSP), introduced in 2007 and Low Energy Secure Connections (LESC), introduced in 2014. Whilst there are considerable differences in how these two handshakes behave at a low level, accounting for changes in Bluetooth packet formats and advances in hardware manufacturing, these two protocols are equivalent from a design perspective. Unlike earlier Bluetooth handshakes, these two protocols were widely considered to be secure and underwent multiple formal analyses using ProVerif [12, 52, 57].

Two devices perform a handshake by exchanging Diffie-Hellman public keys on the NIST P-224 Elliptic Curve, along with some metadata. Next, each device displays a short numeric code which is intended to be a short confirmation code of the earlier exchange. The user checks that both codes match and then the two devices each prove knowledge of the shared secret key. We depict this in Figure 5.2. The importance of public key validation was documented in the standard and implementers of the protocol were recommended to either:

- use an ephemeral Diffie-Hellman key, or
- verify the points are on the correct curve.

Perhaps unsurprisingly, implementers almost universally opted for the first mitigation, as it requires very little additional code. However, as was discovered in 2018 by Biham and Neumann [37], the first mitigation is not sufficient to secure the handshake. The issue impacted all major Bluetooth vendors, including Qualcomm, Broadcom, Intel and Google.

Biham and Neumann observed that the Bluetooth standard requires both the x and y coordinate of the Diffie-Hellman key be transmitted, however only the x coordinate is included in the short numeric code that the user checks. This allows an attacker to silently change the y coordinate, leading to an invalid curve attack. Biham and Neumann perform this attack, setting the y coordinate to 0 which in turn ensures the point $(x, 0)$ will have order 2. This means each device will compute one of two possible keys, either $(x, 0)$ or $\mathcal{O}$, the point at infinity, which are both known to the attacker.

Using our improved symbolic model for elliptic curve elements, Tamarin automatically finds this attack. We model P-224 as a prime order elliptic curve with general coordinates. We systematically investigate the proposed mitigations and claims by Biham and Neumann, as well as proposing one of our own. We also explore our model of subgroup key leakage and evaluate how different mitigations allow or prevent this type of attack. Our full results are documented in Table 5.2.

Our results show, as expected, that if the implementation used static keys rather than ephemeral keys, a key leakage recovery attack could have been performed. Additionally, we propose an alternative protocol level mitigation of authenticating both the x and y coordinates and show this also mitigates the attack.

Surprisingly, we find a new attack that contradicts one of the claims of Biham and Neumann. They state in their disclosure that as long as at least one device is patched

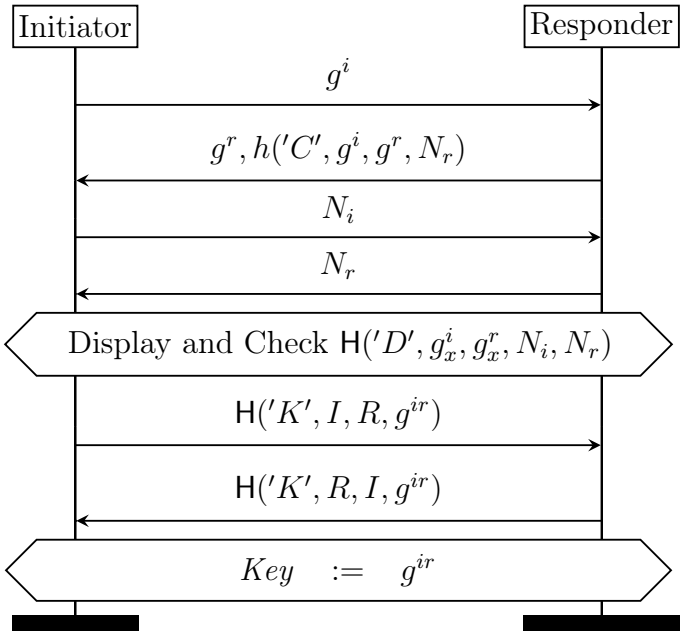


Figure 5.2: Bluetooth's Simple Secure Pairing Protocol. Note that x subscripts on a EC element indicate only the x-coordinate is used

against their attack, the resulting handshake is secure. We show that is not true: it is possible for an attacker to forge the packets such that whilst the patched device will believe the handshake failed due to a network error, the unpatched device will accept the handshake and believe pairing has succeeded. Consequently, whilst an attacker may not learn confidential information exchanged over the Bluetooth link, as only one honest device is on the link, they may still use the link to compromise resources (data, sensors) on the unpatched device.

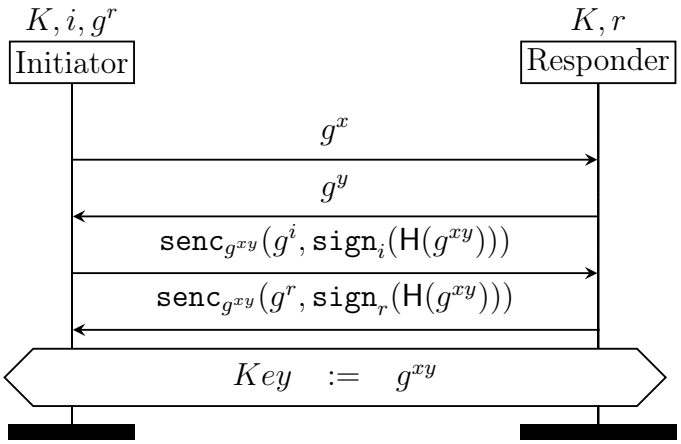


Figure 5.3: Tendermint's Protocol. x and y are freshly generated values.

5.6.3 Tendermint

Tendermint [158] is a 'Blockchain API' which can be used to implement byzantine fault tolerance for arbitrary state machines. One of its features is a secure peer-to-peer layer which authenticates and encrypts connections between clients in a private network. They use a custom handshake [157] which they describe as being based on the Station-to-Station protocol. The protocol, depicted in Figure 5.3, performs an ephemeral Diffie-Hellman handshake over Curve25519 and derives a key and a challenge value from the shared secret. Each party then signs the challenge value and transmits it to the other. Verifying these signatures over the challenge value is intended to prove both parties are using the same secure channel.

We model this protocol in Tamarin with our enhanced DH models and evaluate its security properties. Specifically, we focus on the secrecy of the derived keys and the authenticity of completed handshakes. If an honest party concludes a handshake with another party whose key is not compromised, the resulting key should be secret from the adversary.

Tamarin automatically discovers an attack using small subgroups, which allows an attacker to steal the identity of any party willing to complete a handshake with them. That is, if the attacker completes a handshake with A, the attacker can now connect to B using A's identity. This violates both the secrecy and the authenticity properties mentioned earlier. The attack is relatively simple, if an attacker sends a small order point as their ephemeral public key, the resulting challenge is constant[2], regardless of the other party's contribution. When the handshake completes, the attacker obtains a signature on this constant challenge from the honest participant, which can then be substituted as their own signature in future sessions.

This attack was independently discovered by another security researcher and disclosed to the Tendermint team [139]. The Tendermint team proposed two possible mitigations, the first and easiest being to reject low order points and the second to include the identities of each party in the challenge derivation. We model both mitigations and verify they prevent the attack.

5.7 Automatic Translation of Existing Case Studies

In order to further evaluate the performance of our models, we decided to leverage Tamarin's existing body of protocol analyses which use the traditional DH-model. In this section we apply our enhanced DH models to these existing case studies, through an automated tool we developed to translate between the traditional model and our enhanced one.

[2]Curve25519 always raises by the cofactor as discussed in §5.5.4

Protocol	Variant	Secure?	Leakage?	T (min)
Bluetooth P-224	Original	●	✓	5
	Using static "ephemerals"	●	●	6
	With curve order check	✓	✓	<1
	With static "ephemerals" and curve order check	✓	✓	<1
	Authenticating both coordinates	✓	✓	<1
	Patching one device	●	✓	5
Scuttlebutt Curve25519	Original	●	✓	131
	With exclusion of low order points	✓	✓	88
	Including identities in the KDF	✓	✓	<1
Tendermint Curve25519	Original	●	✓	<1
	With exclusion of low order points	✓	✓	<1
	Including identities in the transcript	✓	✓	<1

Table 5.2: Verification results when applying our various TAMARIN models to each of our case studies. Each case study terminated automatically without any user input or choice of heuristic.
Secure means the protocol achieves its stated security objectives
Leakage means the protocol is susceptible to a key recovery attack
✓ indicates that TAMARIN successfully verified the property
● indicates that TAMARIN found an attack
P-224 is modelled using a prime order elliptic curve with general coordinates; Curve25519 is modelled using a nearly-prime order elliptic curve with a single coordinate ladder, a nearly-prime order twist and low order components cleared after exponentiation.

5.7.1 Choice of Models and Transformations

We developed a tool which can ingest a traditional model and automatically transform it to one of the following models:

- Prime Order Groups
- Nearly Prime Order Groups
- Composite Groups

However, we do not introduce either of our elliptic curve models automatically. This is because the original models do not contain any information about the handling of elliptic curve points or any associated mitigations, as this cannot be expressed in the original models. Consequently, any automatically transformed protocol would be either secure or insecure according to our translation, rather than any information specific to the protocol. On the other hand, the intended group structure is available to us and consequently extending the original protocol models is of considerable interest.

Our automatic transform proceeds much as described in section 5.2. For a given protocol model, we identify a term representing a Diffie-Hellman group element through its use in an exponentiation operation. We must then distinguish between exponentiations carried out on constants, e.g. the construction of g^x from the constant g and the fresh

term x or the protocol operating on a group element of unknown provenance which may have been manipulated by the adversary.

Thankfully, Tamarin encodes this information through its sort system. We consider the construction of a group element from fresh or public terms to be a constant, whereas the use of a variable of unknown sort in the term indicates it could have been produced by the adversary and we invoke our normal machinery. This step is necessary because traditionally protocol models have used g^x to represent some DH element where x is produced by the adversary, as opposed to $g^{\sim y}$ which indicates y is an atomic fresh term produced by the protocol.

We implemented our transformation as an operation on Tamarin's input syntax. As well as being useful in this work, our tooling can also be leveraged to support future protocol modelers by reducing the labour required to use our enhanced models.

5.7.2 Choice of Protocols

Our starting point is list of official Tamarin case studies found at [156]. We filter this list to remove any case studies which do not use Diffie-Hellman groups. We then remove any case studies which use conditional compilation as we cannot automatically generate the correct protocol theory without manual inspection. Finally, we remove a small number of experimental or example protocols, which do not correspond to any protocol published in peer reviewed literature.

This process results in 14 candidate protocols and associated variants. All 14 protocols are 2-party authenticated key exchanges. Across these protocols there are 81 distinct security claims (lemmas) of which 46 hold in Tamarin's traditional DH model. The remainder are false and demonstrate attacks on the protocols.

5.7.3 Results

In the interests of space, we reproduce here only the results which change between DH models. The remainder of the security claims do not vary between the distinct DH models we consider here.

Recall that our prime model allows for an identity element, which is not found in the traditional model. The nearly-prime model also allows for small subgroup elements which under the control over the adversary, as well as equivalent elements. Consequently, each model allows for strictly more adversary behaviour than the preceding one and as might be expected, increasing the adversary's capabilities can only increase the number of possible attacks.

For the five protocols upon which we discover attacks, each was originally described as operating in a prime order group. Thus, attacks using Nearly Prime groups are outside the protocols threat model. However, the protocol descriptions are ambiguous with respect

Protocol	Original Paper	Property	Traditional	Prime	Nearly Prime
NAXOS	[108]	eCK Key Secrecy	✓	✓	•
UM One Pass	[53]	CK Secure	✓	✓	•
KEA+ Fixed	[49]	Initiator Secure	✓	✓	•
		Responder Secure	✓	✓	•
KEA+	[109]	Initiator Secure	✓	✓	•
		Responder Secure	✓	✓	•
		Initiator hwPFS	✓	✓	•
		Initiator KCI hwPFS	✓	✓	•
		Responder KCI hwPFS	✓	✓	•
		Initiator KI KCI	✓	✓	•
		Responder KI KCI	✓	✓	•
MTI C0	[106]	Initiator Secure	✓	•	•
		Responder Secure	✓	•	•

Table 5.3: Verification results when applying our various Tamarin models to automatically transformed protocols.
We omit the composite column as any attack valid in the 'Nearly Prime' model is also valid in this model.
✓ indicates that Tamarin successfully verified the original properties in the specified model
• indicates that Tamarin found an attack in the specified model.

to prime order groups. In particular, no protocol description is explicit about how to handle the identity element, and some give conflicting guidance over how to implement the protocol using a prime order subgroup of a composite DH group.

As it transpires, nearly all the protocols we consider are secure regardless of whether the identity element is accepted. However, unlike the other protocols, an attack is discovered on MTI C0 in the Prime model. On inspection, this is because the adversary can send the identity element to an honest agent which forces the eventual key to be the identity element and thus breaks the protocol. Interestingly, the authoritative description of the protocol [127, §12.53] does not specify that the identity element must be rejected. However, the paper [106] proving the security of MTI C0 is explicitly rejects the identity element, thus our automatically found attack highlights the importance of this requirement - it is not simply an assumption made to simplify the analysis. We verified that when we add the mitigation described in 5.5.1, rejecting the identity element, that the protocol does indeed verify as secure.

When we consider the composite model, it transpires that none of the protocols are vulnerable to small subgroup key leakage attacks . This is surprising, however we believe it is due to a combination of three factors:

- All the protocols in our sample only exchange ephemeral DH values over the wire and expect pre-provisioned static keys. Ephemeral keys are typically not vulnerable to leakage attacks.

- The protocol models do not typically include full range of adversary capabilities with respect to static keys. For example
  - Updating the public key associated with a (compromised) identity.
  - Registering a malicious public key (as opposed to compromising an honest public key)
- The protocol models do not typically include any higher level application operations. For example, most authenticated key exchanges will be followed with a message exchanged under agreed key (or some other function of the key). Access to such a message is often required for a subgroup leakage attack to be carried out.

This highlights a shortcoming with automatically repurposing protocol models designed for different threat models. It also highlights the importance of developing complete models, which do not omit protocol behaviour simply because it does not appear to be 'interesting'.

Statistic	**Traditional**	**Prime**	**Nearly Prime**
Lemmas Proven Secure	46	44	33
Lemmas Violated	35	37	48
Total Running Time (s)	486	733	732
Average Running Time (s)	6	9	9
Longest Single Running Time (s)	86	144	142

Table 5.4: Summary of performance results for our automatically transformed case studies. No manual intervention was required and the default heuristic was used in each study.

We also report a summary of running times for all our case studies in Table 5.4. Extending the nearly prime model to the composite model results in identical running times for the existing lemmas and a further 126 seconds in total to prove the leakage lemma for each case study. These results, achieved on automatically transformed models developed before our DH models were conceived of, highlight the efficiency of our approach. We did not have to alter any existing protocol descriptions or provide any heuristics in order to achieve termination. Additionally, the additional time required was relatively minimal and appears to be linearly proportional to the existing running time.

Our results in this section showcase the effectiveness of our symbolic models, across 14 candidate protocols, 81 distinct security claims and 3 different DH models, we are able to achieve automatic termination without manual intervention in all of them. Our additional models result in approximately a 50% increase in running time, which whilst not negligible, is perfectly tractable. With our new models we able to to provide the first analysis (computational or symbolic) of these protocols in a non-prime order group and demonstrate concrete attacks if the protocols are used without due care.

5.8 Future Work

As is typical for automated analysis performed in the symbolic model, our verification results are not proven to be computationally sound. That is, a successful verification does not necessarily imply the existence of a reduction from an attack on the protocol, to some underlying hard problem. However, we view our work as the first step towards realising a computationally sound model of real world Diffie-Hellman groups. We have shown our model to be tractable and effective at capturing these new behaviours and look forward to investigating computational soundness results in future work.

With symbolic models becoming increasingly granular, automatically generating models from reference implementations looks increasingly attractive. It would be interesting to see whether recent work on automatically generating implementations can be integrated with our granular model of Diffie-Hellman groups.

5.9 Conclusions

In this work, we have identified and addressed several shortcomings of the traditional symbolic models for Diffie-Hellman groups. We have introduced models to capture rich group structure such as small subgroups and key leakage attacks, and explored the additional attacks and capabilities of an attacker who exploits invalid elliptic curve points. We have provided a family of symbolic models which are suitable for both verification and attack finding and implemented them in Tamarin. Through a series of case studies, both bespoke and automatically generated, we have discovered novel attacks on real world protocols, provided new analysis results on protocols in alternative settings and verified the practicality of our models.

5.9.1 Real World Impact

New Attack on Secure Scuttlebutt Using our new family of symbolic models, we were able to accurately represent the behaviour of Curve25519, including mitigations such as its ladder implementation and raising by the cofactor. This accurate model allowed us to discover a new attack on the Secure Scuttlebutt secret handshake. As the attack relies on a subtle combination of Diffie-Hellman properties we implemented our attack on the Go port of Secure Scuttlebutt and confirmed it works. We then contacted the Secure Scuttlebutt developers and worked with them to patch the vulnerability in their design.

Validation Missing in NaCl Compatible Libraries As part of our work on Secure Scuttlebutt, we reviewed a number of implementations in different languages. Each implementation used the same NaCl API for cryptography, which is one of the most popular cryptographic APIs available and is used on all modern platforms. We discovered

that the original NaCl library had a cryptographically consequential check added, which was omitted from many alternative libraries which advertised themselves as 'drop in' replacements. In particular, a check for small subgroup elements on X25519 was present in NaCl, but missing in HaCl - the formally verified library from Project Everest, CirCl - from Cloudflare and the Golang standard library NaCl implementation. We contacted the developers of each library and notified them of the missing check, which all but HaCl have been swift to add.

First Automated Analysis Results for Non-Prime Order Groups Our symbolic models and Tamarin implementation represent the first automated method for analysing protocols in this setting. Although manual computational analysis can principle consider any group structure, it is most often carried out in prime order groups, meaning many proofs of security do not apply to (or even discuss) the non-prime order case. Despite the high popularity of non-prime order groups such as Curve25519. We leverage our models to consider existing protocols in this new setting and show concrete attacks in the event they are misused by implementers.

6

Modelling the Field Structure of DH Exponents

This chapter consists of unpublished work which is under active development. The work is my own, in conjunction with my supervisor Cas Cremers. It is an extension and generalisation of earlier work by Dougherty and Guttman which is discussed in Section 6.2.

6.1 Introduction

In this chapter, we revisit the problem we discussed in Section 4.2.6.2: the lack of support for direct use of the group operation in current state of the art Diffie-Hellman symbolic models. Such models cannot represent the group operation due to undecidability results in unification theory. Consequently no existing automated tool (e.g. Tamarin, ProVerif, Maude-NPA or CPSA) can currently model a protocol which uses the field structure of DH exponents. Additionally, due to Kaliski's attack on MQV [97], we know that allowing the adversary access to this group operation is indeed necessary in order to avoid missing attacks on such protocols.

Our primary contribution in this chapter is a new set of constraint solving rules, extending Tamarin's existing framework, which enables the verification of protocols which make direct use of the group operation. Our work builds on and extends that of Dougherty and Guttman, who provided a new equational theory for capturing the behaviour of a finite field in [74, 75] and showed how it could be used to prove security of a selected class of protocols. We target a wider class of protocols and aim to integrate our constraint solving rules with the Tamarin prover.

We stress that our work in this chapter is still in progress. In particular, we have not yet implemented our constraint solving rules in the Tamarin Prover. Whilst we provide proofs or proof sketches for many of our theorems, they will undergo further refinement prior to publication. Additionally we leave some statements open as conjectures which we are still investigating. Finally, there are several directions we would like to extend this work in before seeking publication, notably support for attack finding, which we discuss further in Section 6.10.

We structure this chapter as follows:

- In 6.2, we discuss previous work, including results that rule out straightforward extensions of existing symbolic models to capture group operations directly. We also discuss the results of Dougherty and Guttman that our work is founded on.
- In 6.3, we give an overview of our new constraint solving rules and present them in detail in sections 6.4 to 6.7.
- In 6.9, we show the effectiveness of our constraint solving rules on some hand worked examples.
- Finally, we identify a number of areas of future work in 6.10 and conclude in 6.11.

6.2 Related Work

In this section, we discuss previous work in this area. Firstly, we recap why straightforward extensions of the existing approaches to Diffie-Hellman models, rooted in unification, are believed to be impossible to extend to capture direct use of the group operation. Secondly,

we discuss a recent line of work of Dougherty and Guttman, which offers an alternative approach.

6.2.1 'No Go' Results Regarding Unification

Having discussed the history of equational theories in symbolic models in Chapter 4, we will now examine why the current approach cannot be directly extended. We recall the field structure of Diffie-Hellman exponents which we are trying to model:

Definition 40. A **field** is a set with the signature $\Sigma_F = \{\cdot^{(2)}, +^{(2)}, \div^{(1)}, -^{(1)}, 1^{(0)}, 0^{(0)}\}$ satisfying the following axioms:

$(a + b) + c = a + (b + c)$	Associativity of Addition
$(a \cdot b) \cdot c = a \cdot (b \cdot c)$	Associativity of Multiplication
$a + b = b + a$	Commutativity of Addition
$a \cdot b = b \cdot a$	Commutativity of Multiplication
$a + 0 = a = 0 + a$	Additive Identity
$a \cdot 1 = a = 1 \cdot a$	Multiplicative Identity
$a + (-a) = 0 = (-a) + a$	Additive Inverse
$a \cdot (\div a) = 0 = (\div a) \cdot a$ if $a \neq 0$	Multiplicative Inverse
$a \cdot (b + c) = a \cdot b + a \cdot c$	Left Distributivity

Contained in a finite field is a slightly simpler structure: a commutative ring.

Definition 41. A Commutative Ring (CR) is a set with the signature:

$$\Sigma_{CR} = \{\cdot^{(2)}, +^{(2)}, -^{(1)}, 1^{(0)}, 0^{(0)}\}$$

and the field axioms, excepting the axiom for multiplicative inverse.

All fields are also commutative rings, but so are structures like $(\mathbb{Z}, +, \times, -, 0, 1)$. We now look at the computational aspects.

Theorem 4. *Unification in the equational theory of CR is undecidable.*

Proof. (Sketch) It is known that the initial algebra[1] of CR is the ring of integers. Consider a term built over the signature of CR and containing n different variables, this can be interpreted directly as a polynomial over the integers in n indeterminates. Furthermore, for every such polynomial there exists such a term. Substituting ground terms for the

[1] An initial algebra is an interpretation of an equational theory which can be found in any model of that theory

variables of a term is equivalent to evaluating the corresponding polynomial with the integers corresponding to these ground terms. This mapping exists as $\mathbb{Z}$ is the initial algebra. It is easy to see that a CR-Unification problem has a solution if and only if it has a substitution that substitutes only ground terms. Consequently, there is a reduction from solving the CR-Unification problem to solving polynomial equations over the ring of integers, which is Hilbert's 10th problem. Consequently, by Matiyasevich's theorem [119], which answered Hilbert's 10 problem in the negative, CR-Unification is undecidable. □

This sketch is taken from [83]. There is a more formal proof in [98]. Consequently, there can be no effective procedure[2] for unification in commutative rings.

Theorem 5. *CR-Unification is not finitary*

Proof. Consider the instance $x^2 - 3y^2 - 1 = 0$ of Pell's equation[3]. It is known that (x, y) is a solution of this equation in $\mathbb{Z}$ if and only if $|x| + \sqrt{3}|y| = (2 + \sqrt{3})^n$ for some integer n. Thus $x^2 - 3y^2 - 1 = 0$ has an infinite number of constant polynomial solutions. Since no constant polynomial can be obtained from another constant polynomial by substitution, it follows that all solutions of the above equation are most general and they are pairwise incomparable. □

This result is from [51]. Furthermore it turns out that CR-unification is of type zero [82], which is to say there are CR-unification problems in which no minimal unifier exists, there is only an infinite chain of unifiers, each one more general than the last. Consequently, even if a we had a CR-Unification oracle which terminated in practice, some unification queries would not have a finite answer and thus would not be tractable for mechanical proofs.

This means unification in commutative rings is unworkable for our purposes. Worse, these results on the undecidable and non-finite nature of CR-Unification also extend directly to finite fields. Perhaps there is there a theory weaker than CR but still stronger than the equational theory in Tamarin? Unfortunately, it is the interaction between distributivity and associativity that gives rise to the undecidability of CR-Unification. The following discussion is taken from [83]:

$$
\begin{aligned}
D_L &= \{f(g(x,y),z) = g(f(x,z),f(y,z))\} \\
D_R &= \{f(z,g(x,y)) = g(f(z,x),f(z,y))\} \\
D &= D_L \cup D_R \\
DA &= D \cup \{g(g(x,y),z) = g(x,g(y,z))\}
\end{aligned}
$$

[2] It is natural to consider if we can regain decidability by imposing restrictions on the polynomials considered. Unfortunately it has been shown that any Diophantine equation can be rewritten into one of degree 4 or less, with no greater than 9 variables. Within this set, general solutions are only known for a very limited class.

[3] $x^2 - dy^2 - 1 = 0$

Consider the above equational theories, which correspond respectively to left (D_L) and right (D_R) sided distributivity, full distributivity (D) and distributivity and associativity (DA). D_L-Unification and D_R-Unification are decidable. DA-Unification is undecidable and furthermore unification is undecidable for every equational theory over the signature CR that contains DA and is a subset of CR. The decidability of D is still an open question, although there is a proposed algorithm without proof of correctness in [145]. This rules out any useful extension of the existing model as distributivity is an essential property.

In summary, known results appear to rule out any straightforward extension of existing approaches which rely on unification. In addition to undecidability results which apply to even quite restrictive classes of equations, there are simple equations which do not have a finite set of most general unifiers. As existing automated symbolic model tools are based heavily on unification for their mechanical reasoning, this poses a significant problem.

6.2.2 Previous work on group operations

We now discuss a promising recent line of work on sidestepping the results in the previous section. In 2012, Dougherty and Guttman published a paper establishing several new and important results in the symbolic modelling of modular exponentiation [74]. In particular they presented a novel term rewriting system and proved it decides equality for a structure representing generic finite fields. Furthermore they showed with this term rewriting system, they could derive a symbolic analogue for the discrete log assumption (a computational assumption that deducing e from g^e is not possible in sufficiently large groups of prime order). By hand, they sketched out how these new results can prove key exchange protocols such as MQV and Unified Model (UM) satisfy reasonable security properties.

In 2014, they published [75] which extends their earlier work and provides a new decidability result for a class protocols which use modular exponentiation, with substantial restrictions. In particular, they were able to show that for the considered class of protocols they could decide a limited set of security properties. Although they have not implemented their approach, they show the effectiveness of their approach with hand worked examples. As our own work is directly based on their methodology, we now discuss these two papers in detail.

6.2.2.1 An Algebra for Symbolic Diffie-Hellman Protocol Analysis

Dougherty and Guttman's first contribution is an order-sorted term rewriting system to represent equations which hold in a generic finite field. Their signature consists of three sorts. The first sort, G, represents terms which are group elements e.g. g^x and a sort E represents exponents such as x. A further sort NZE, a subsort of E, represents terms

which are exponents and also known to be non-zero. They define the following order sorted signature:

$$\begin{array}{lll} \cdot : G \times G \to G & + : E \times E \to E & ** : NZE \times NZE \to NZE \\ id. :\to G & id_+ :\to E & id_{**} :\to NZE \\ {}^{-1} : G \to G & inv_+ : E \to E & inv_{**} : NZE \to NZE \end{array}$$

$$\begin{array}{lll} exp : G \times E \to G & \times : E \times E \to E & \mu : G \to E \end{array}$$

Note that later we typically write $exp(g, x)$ as g^x. This is the expected signature for group multiplication, exponent addition and exponent multiplication. There is also an exponentiation operator, a operator μ to coerce group elements into exponents and an operator $\times$ for multiplication of "possibly-zero" exponents.

We will see the details shortly, but the intent is that $**$ and $\times$ perform the same function, but only the multiplicative inverse of NZE elements is defined, so we must differentiate between them. The equational theory EQT_{DH} is:

1. $(G, \cdot, ^{-1}, id.)$ is an abelian group
2. $(E, +, id_+, inv_+, \times)$ is a commutative ring with identity
3. Exponentiation makes G a right E-module with identity, which is to say:

$$\begin{aligned} (a^x)^y &= a^{x \times y} \\ (a \cdot b)^x &= a^x \cdot b^x \\ a^{id_{**}} &= a \\ (id.)^x &= id. \\ a^{(x+y)} &= a^x \cdot a^y \end{aligned}$$

4. Multiplicative inverse, closure at sort NZE

$$\begin{aligned} u ** v &= u \times v \\ inv_{**}(u ** v) &= inv_{**}(u) ** inv_{**}(v) \\ u ** inv_{**}(u) &= id_{**} \\ inv_{**}(id_{**}) &= id_{**} \\ inv_{**}(inv_+(u)) &= inv_+(inv_{**}(u)) \\ inv_{**}(inv_{**}(v)) &= v \end{aligned}$$

This defines $\cdot$, $+$, and $\times$ quite naturally in 1 and 2. 3 defines the relationship between group elements and exponents. Finally in 4, the relationship between E and NZE. Let R be the set of rewrite rules given by the natural (left to right) orientation of these equations,

other those equations corresponding associativity and commutativity. Furthermore with the additional[4] rules below, then the resulting rewrite relation $\rightarrow_{EQT_{DH}}$ is defined to be rewriting with R modulo the associativity and commutativity equations. The additional rules are:

At sort G:	At sort E:
$inv.(a \cdot b) \rightarrow inv.(a) \cdot inv.(b)$	$inv_+(id_+) \rightarrow id_+$
$inv.(inv.(b)) \rightarrow b$	$inv_+(x + y) \rightarrow inv_+(x) + inv_+(y)$
$(inv.(a))^x \rightarrow inv.(a^x)$	$inv_+(inv_+(x)) \rightarrow x$
$a^{id_+} \rightarrow id.$	$id_+ \times x \rightarrow id_+$
$a^{inv_+(x)} \rightarrow inv.(a^x)$	$inv_+(x) \times y \rightarrow inv_+(x \times y)$
$inv.(id.) \rightarrow id.$	

Dougherty and Guttman now present some of the properties of this equational theory and rewrite system:

Theorem 3, [74]. *The reduction $\rightarrow_{EQT_{DH}}$ is terminating and confluent modulo AC*

Consequently we can use rewriting modulo AC to decide equality between terms. We now give an informal statement of one of their main results:

Theorem 9, [74]. *$\forall s, t \in G$ then s reduces to s', t reduces to t' with s' and t' irreducible, then s' and t' are identically modulo associativity and commutativity of $\cdot$, $+$ and $\times$ $\iff$ there are infinitely many finite fields in which $s = t$*

This result, originally stated in the language of model theory, shows that this equational theory captures the structure we are interested in. Importantly it does not describe any particular finite field, rather it describes equations which are true in all but finitely many finite fields. This bypasses impossibility results in category theory which show that a fixed finite field cannot be described in an equational theory [118].

Furthermore, Dougherty and Guttman characterise the normal forms of this term algebra:

Lemma 4, Part 1 [74]. *If $e : E$ is a normal form then e is a sum $m_1 + \ldots + m_n$ where:*

1. *Each m_i is of the form $\pm(e_1 \times \ldots \times e_k)$ where $k \geq 0$*
2. *No e_i is of the form $inv_{**}(e_j)$*
3. *Each e_i is one of: $x : NZE$, $inv_{**}(x : NZE)$, $\mu(t : G)$, $inv_{**}(\mu(t : G))$*
4. *When $n = 0$ then e is the element id_+*
5. *When $k = 0$ then m_i is the ring element id_{**}*

[4] Required to join critical pairs which ensures confluence

We call terms of the form $\pm m_i$ ***irreducible monomials****.*

Lemma 4, Part 2 [74]. *If $t : G$ is a normal form then t is a product $t_1 \cdot \ldots \cdot t_n$ with $t \geq 0$ where:*

1. *No t_i is of the form $inv.(t_j)$*
2. *Each t_i is one of: v, $inv.(v)$, v^e, $inv.(v^e)$ with $e : E$ an irreducible monomial and $v : G$*
3. *When $n = 0$, then $t = id$.*

Next, they introduce some useful definitions for reasoning about these normal forms.

Definition 10, [74]. *Say that a monomial m is a maximal-monomial of t if t has a subterm of the form b^m. Let $N = (v_1, \ldots, v_d)$ be a vector of NZE-variables.*

If m is an irreducible monomial, the N-vector for m is $(z_1, \ldots, z_d)$ where z_i is the multiplicity of v_i in m, counting occurrences $inv_{**}(v_i)$ negatively.

An E-term $e = m_1 + \ldots m_k$ is N-free if each m_i has N-vector $(0, \ldots, 0)$.

If t is irreducible, then $Ind_N(t)$ is the set of all vectors z_m such that z_m is the N-vector of m where m is a maximal-monomial subterm of t.

Let $T = \{t_1, \ldots, t_k\}$ be a set of terms. Then $Gen(T)$ is the set generated by T: the least set of terms which includes T and is closed under the application of function symbols.

These definitions allow a term to be described as a vector in terms of its N components. Some examples follow:

Example 1. Let $N = (x, y)$. Let $m = inv_{**}(x) \times y \times y \times z$ be an irreducible monomial. The N-vector for m is $(-1, 2)$.

Example 2. Using the same N as before, we have that:

$$x + z + w \text{ is not N-free}$$
$$z + w \text{ is N-free}$$
$$inv_{**}(y) \text{ is not N-free}$$

and

$$Ind_N\left(g^{x \times inv_\times(y)} \cdot g^{x \times y \times y} \cdot g^{x \times x}\right) = \{(1, -1), (1, 1), (2, 0)\}$$

This leads to the primary theorem of Dougherty and Guttman:

Indicator Theorem, [74]. *Let N be a vector of NZE-variables and let T be a set of terms each $e \in T$ such that e is of sort E is N-free. Then*

1. *Every $e \in Gen(T)$ of sort E is N-free, and:*
2. *If $u \in Gen(T)$ is of sort G and $z \in Ind_N(u)$, then for some $t \in T$, $z \in Ind_N(t)$.*

This theorem is an analogue of the computational discrete log assumption in the symbolic model. Namely, if you don't know an exponent e, you cannot manipulate g^e to calculate g^{e^2} or $g^{inv_\times(e)}$. As a corollary, if N is defined to be the vector of exponent values used by the protocol, which the adversary does not know, then if the adversary constructs a term with z a non-zero N-vector, there must have been a message sent with that N-vector z.

Dougherty and Guttman then goes on to apply these results in a strand space formalism, which is an alternative operational semantics for the symbolic model. The precise details of strand spaces are not important to us, but in short roles are defined by "strands" which are made up of nodes, a node is defined to be either message transmission, message reception or a neutral node (in which local state is changed). They then go on to adapt their general theorem to the specifics of strand spaces, by defining when exponents are considered to be "non-originating" which in turn defines the set of known terms of the adversary and consequently the set of terms it is possible for the adversary to generate.

Finally, the authors go on to hand prove various security properties of some key agreement protocols, namely: MQV [124], UM [7] and a weakened variant of Cremers-Feltz (CF) [63].

6.2.2.2 Decidability for Lightweight Diffie-Hellman Protocols

In their follow up paper [75], the Dougherty and Guttman extend the results of their earlier work by considering a specific class of protocols and show a decision algorithm for some security properties in that protocol class.

Firstly, the authors define a restricted class of protocols to consider which they term *Lightweight Diffie-Hellman Protocols*. They require these protocols to be well typed, which is to say they only send or receive values which are simple group elements in the form g^x or other primitive values such as names, nonces, or are recursively built out of the prior with pairing and digital signatures. The restriction that only simple group elements of the form g^x may be transmitted is limiting, as many protocols that make use of the full field structure transmit composite elements. Although the protocols are allowed to compute composite group elements in their local state, e.g. keys, they cannot transmit such elements to the adversary. Another requirement is that whether or not a specific an exponent is deemed to be compromised is encoded as an assumption rather than a security goal to be proven.

The paper goes on to define a new adversary with power to employ the full algebraic structure of finite fields and a goal language for security properties which can encode properties such as key secrecy, forward secrecy and so on. The authors then prove that the resulting combination of protocols, adversary and goals has the small witness property: which is to say if there exists a protocol run that violates a security property, then some run smaller than a computable bound also violates it.

Finally, the authors give an algorithm for testing every protocol run within the bound and verifying whether or not the security goal holds. Their process is to build a constraint system, which in turn reduces to a system of linear equations which can be solved by Gaussian elimination. Consequently, they have a decision procedure for this class of protocols. They go on to demonstrate this procedure and derive both key secrecy for MQV and Kaliski's attack on MQV. This is a significant result, previously no symbolic model could express these protocols, proofs or attacks.

6.3 Overview of our Presentation

In the main body of this chapter we develop a new formulation and extension of Dougherty and Guttman's work. Rather than following the style of [75], we will investigate a flexible constraint solving system in the style of Tamarin.

Our approach is to embrace the underlying term algebra and its associated invariants of [74]. However, we extend it with a number of generalisations and combine it with a different high level approach for modelling protocols. Due to our alternative construction, we lift a number of restrictions present in [75]. This expands not only the class of protocols which can be modelled, but also the security properties which can be considered.

There are three key insights that form the bulk of work. Firstly, we discuss how to combine Dougherty and Guttman's equational theory with an arbitrary user defined equational theory, as opposed to the fixed equational theory of hash functions, symmetric encryption and digital signatures used in [74].

Secondly, we use a novel encoding of partially fixed term structure in our constraint system. This insight allows us to perform an effective backwards search, rather than the forwards search described in [75] and is crucial to our approach. We also rework the results of [74] to fit in our new representations.

Finally, we show that equality constraints in our new extended equational theory can be partially solved using unification in the traditional symbolic Diffie-Hellman model. This is vital in order to lift the restrictions on [74] that limit the class of considered protocols to those that only output fixed group elements. We can instead consider protocols which receive group elements, operate on them and output the results to the adversary.

We develop our formulation in the following sections:

- In 6.4, we show how we adapt Tamarin's existing term rewriting system to the one developed by Dougherty and Guttman. We will show how these distinct systems can be composed.
- In 6.5, we will generalise the notion of indicators introduced by Dougherty and Guttman and provide an analogue as a set of constraint solving rules.
- In 6.6, we show how unification can applied on a simplified equational theory in order to derive new constraints on indicator sources.

- In 6.7, we provide a new set of constraint solving rules which allow us to reason about indicators in order to find additional equalities and constraints.
- In 6.8, we summarise our constraint solving rules and discuss their composition.
- Finally, in 6.9, we show how our design can be used to automatically discover proofs of the absence of attacks in a set of protocols that use the group operation directly.

6.4 Adapting Tamarin's Equational Theory

In this section, we show how Tamarin can be adapted to support the equational theory introduced in [74]. We first describe how the new sort structure can be implemented in Tamarin. This is easier than might be expected as Tamarin's existing sorts are only used as a termination aid rather than as carrying any additional expressive power. Finally, we show how unification in the simpler traditional *DH* equational theory can be used to derive constraints for the new equational theory.

6.4.1 Integrating Equational Theories

As discussed in 2.3.1, Tamarin has a single top sort M with two subsorts representing fresh and public terms. Fortunately, these subsorts are not strictly necessary and do not influence Tamarin's operational semantics except as an optimisation. As we are introducing an orthogonal sort system, we will remove these existing sorts whilst preserving Tamarin's operational semantics and existing user interface.

We take the current sort M and add the additional sorts G and E with NZE as a subsort of E.

Definition 42. When we say *DH*-term, we mean a term of one of the sorts: G, NZE or E.

Note there is no sort relationship between M and G or E. This means there is no single top sort, rather there are three. Additionally we provide the following two functions which satisfy no additional equations:

$$\mathtt{box}_G : G \to M$$
$$\mathtt{box}_E : E \to M$$

We use these functions to separate the two term rewriting systems, representing the act of converting the group elements or integers to a bitstring. We do not provide a function for unwrapping the box functions, instead we will use pattern matching in rules to allow the protocols and the adversary to unwrap such terms. We show this structure in Figure 6.1.

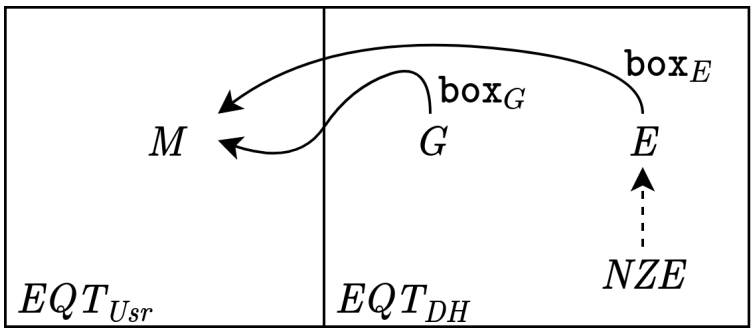


Figure 6.1: This diagram shows the relationship between the four sorts. Notice there are is only a subsort relationship between *NZE* and *E*, indicated by the dashed line. The user defined equational theory is seperate from the *DH* equational theory with a two uninterpreted functions allowing *DH* terms to be contained in *M* terms.

Tamarin uses a pre-computation procedure based on unification to automatically derive a set of rules which allow the adversary to deconstruct terms in order to learn new information and construct new terms from the adversary's knowledge set. Of course, this approach will not work for our *DH* equational theory as there is no unification algorithm. Instead, we will only generate construction and deconstruction rules for the EQT_{Usr} equational theory. We discuss how we handle adversarial construction and deconstruction of EQT_{DH} terms in 6.6.1.

6.4.2 Unification in Disjoint Equational Theories

As previously discussed in 2.4.4, Tamarin uses unification to resolve equality constraints between terms. Where terms do not contain any `box` functions, we can use Tamarin's normal approach to unification. Similarly, when terms are of *DH* sort, we will not in general resolve these equality constraints directly. However, this leaves the case of a mixed term, of *M* sort but using `box` functions. Thankfully, due to the disjoint equational theories, we can compute the set of unifiers for the outermost *M* terms and simply introduce additional equality constraints on any resulting *DH* subterms.

We first define a function which converts a term into its 'outer' and 'boxed' components. Let $\mathcal{V}$ be the class of variables and $\mathcal{S}$ be the class of substitutions over EQT_P. For a set B, we use $\mathcal{P}(B)$ to refer the power set of B.

$$\texttt{Cl} : M \times \mathcal{P}(\mathcal{V}) \to M \times \mathcal{P}(\mathcal{S})$$

`Cl` is given a set of fresh variables (not currently used in constraint system) and a *M*-term. It provides a *cleaned* *M*-term and a set of substitutions which restores the cleaned term to original state. In a *cleaned* term, the internal structure of any subterm of *G* or *E* sort has been replaced with a variable. This transformation is reversible and the cleaning function returns the substitutions which would reverse the transformation.

Definition 43.

$$\mathtt{Cl}(X,V) = \begin{cases} \mathtt{box}_G(\theta : G), \{\theta \to \alpha\} & X = \mathtt{box}_G(\alpha), \theta \in V \\ \mathtt{box}_E(\theta : E), \{\theta \to \alpha\} & X = \mathtt{box}_E(\alpha), \theta \in V \\ \mathtt{Cl}_1(X_1, V_1) \circ \ldots \circ \mathtt{Cl}_1(X_n, V_n) & \\ \mathtt{Cl}_2(X_1, V_1) \cup \ldots \cup \mathtt{Cl}_2(X_n, V_n) & X = X_1 \circ \ldots \circ X_n \\ X & \text{otherwise} \end{cases}$$

where $\circ \neq \mathtt{box}_G, \mathtt{box}_E$ and $V_1, \ldots, V_n$ are a non-finite partition of V. Note that $\mathtt{Cl}$ is outputs a 2-tuple and $\mathtt{Cl}_1$ refers to the first element of the returned tuple, $\mathtt{Cl}_2$ refers to the second element.

We will typically omit the second parameter as we will always consider V to be an countably infinite set of fresh variables (i.e. $\mathcal{V}$). We now formally define the restriction of a substitution to a particular sort.

Definition 44. For a substitution σ and sort S, we write $\sigma|_S$ to be the substitution resulting from the restriction of σ to the domain of variables in S.

We define the function $\mathtt{ex}$ to be a mapping from a substitution to a set of constraints, wherein renamed variables are lifted to equality constraints:

Definition 45. We define the function $\mathtt{ex} : \mathcal{S} \to \mathcal{C}$ where $\mathtt{ex}(\{\alpha \to \beta\} := \{Eq(\alpha, \beta)\}$ for $\alpha, \beta \in \mathcal{V}$. Where the substitution contains more than one mapping between variables, we lift $\mathtt{ex}$ to the union of the set of constraints in the natural way.

We now introduce our constraint solving rule. From a constraint system containing an Equality fact, we produce a set of constraint systems (one new constraint system for each most general unifier). Notice that whilst this rule solves the existing Equality constraint, it will introduce a new equality constraint for each DH-subterm. These new equality constraints cannot be solved by unification directly.

$$\frac{Eq(A,B), \Gamma}{\{\beta\sigma|_M(\Gamma), \beta\mathtt{ex}(\sigma|_{DH})\}}\,\mathcal{C}_{\approx} \qquad \begin{array}{r} \beta = \alpha_1 \ldots \alpha_n, \alpha_i \in \mathtt{Cl}_2(A) \cup \mathtt{Cl}_2(B) \\ \sigma \in MGU(\mathtt{Unify}_{EQT_{Usr}}(\mathtt{Cl}_1(A), \mathtt{Cl}_1(B))) \end{array}$$

Figure 6.2: Rule $\mathcal{C}_{\approx}$

Notice this is a set of constraint systems over the most general unifiers of the cleaned term. Each constraint system is the result of applying the M-unifier, reversing the cleaning and adding additional equality constraints for each DH-subterm.

Theorem 6. *$\mathcal{C}_{\approx}$ is complete*

Proof. **(Sketch)**

Assume the constraint system has some fixed solution and $\mathcal{C}_{\approx}$ can be applied. Then there must be some total substitution ψ such that $\psi(A) =_E \psi(B)$ which is compatible with the fixed solution.

As the equational theories are disjoint, ψ can be decomposed such that $\psi = \psi_{DH}\psi_M$ where each substitution only acts on terms of that sort and uses only function symbols from that equational theory.

We now show that applying $\mathcal{C}_{\approx}$ will preserve this solution. Firstly, we apply the cleaning function cl and unify the resulting terms, returning a set of most general unifiers. By the definition of most general unifiers (Def 14) and the reversibility of cl, there must be a MGU σ and a compatible substitution ϕ such that $\psi_M = cl^{-1}\phi\sigma$.

$$\begin{array}{ccccc} A & \xrightarrow{\psi_M} & & \xrightarrow{\psi_{DH}} & \psi(A) \\ \downarrow{\scriptstyle cl} & & \overset{cl^{-1}\phi}{\uparrow} & & \\ & \xrightarrow[\sigma]{} & & & \end{array}$$

Furthermore, the substitution ψ_{DH} only operates on DH terms and due to the fixed structure of the 'outer' M term, the individual DH terms in each box position must be equal. Consequently, ψ_{DH} is compatible with the `ex` equality constraints and the solution is preserved.

□

Theorem 7. *Rule $\mathcal{C}_{\approx}$ is sound*

Proof. **(Sketch)**

We show that if after applying $\mathcal{C}_{\approx}$, the constraint system has a solution, then that solution was also a valid solution prior to the rule application.

Our argument proceeds in much the same way as before. Assuming ψ is a total substitution providing the solution. It can necessarily be expressed in terms of an MGU of $Eq(cl(A), cl(B))$ and consequently as ψ must be compatible with the induced equality constraints from `ex` it must be also be compatible with this MGU and hence a solution to the original $Eq(A, B)$ constraint.

□

We now see how this rule works in practice with a short example:

Example 3. We consider a simple constraint system containing only one constraint, equality between two terms. In the following, the function f is of arity 2 and is uninterpreted (satisfies no equations). "n" is a constant.

$$Eq(f(m, \mathtt{box}_G(g^{xy})), f(\text{“}n\text{”}, \mathtt{box}_G(g^z)))$$

As these terms are of sort M we can apply the above constraint solving rule. When we 'clean' these terms, we are left with:

$$Eq(f(m, \mathtt{box}_G(\alpha)), f(\text{“}n\text{”}, \mathtt{box}_G(\beta)))$$

This has a single most general M-unifier:

$$\{m \to \text{“}n\text{”}\}, \{\alpha \to \beta\}$$

And this second substitution is mapped into an equality constraint, leaving us with:

$$\{m \to \text{“}n\text{”}\}, Eq(g^{xy}, g^z)$$

6.4.3 Indelible Terms

We will also introduce a new type of constraint which constrains the structure of a term. In this section, we introduce ‘Indelible’ constraints which describe when a subterm cannot be cancelled or otherwise destroyed by any future substitution. We will use these constraints to reason about invariants in non-ground terms.

We introduce some definitions for describing the subterms of a DH-term.

Definition 46. We say a term x is an ingredient of a normalised DH-term Y, written $x \sqsubseteq_I Y$, if there is a position p such that $Y|_p = x$ and this position is not enclosed within a exp or μ function.

Definition 47. For a general DH term X and an associative and commutative operator $\circ$ with $X = x_1 \circ \ldots \circ x_n$, we speak of the **root terms** of X under $\circ$ as the multiset, written $\mathtt{rt}_\circ(X)$ as $\{x_1, \ldots, x_n\}$. We say x_1 is a root term of X. We define a **root term** under an operation $\circ$ to be a term X which is normalised under the term rewriting system such that $|\mathtt{rt}_\circ(X)| = 1$. Where the operator is clear from context, we sometimes omit it.

Example 4.

$$\begin{aligned} \mathtt{rt}_\cdot(g^x \cdot g^y) &= \{g^x, g^y\} \\ \mathtt{rt}_\cdot(g^x) &= \{g^x\} \\ \mathtt{rt}_+(x + y) &= \{x, y\} \end{aligned}$$

We now introduce the important definition for this section.

Definition 48. We say a root term x is **indelible** from the term Y under the equational theory T and invertible function $\circ$ if:

$$\forall \overrightarrow{\theta} \in \mathcal{S}\ \mathtt{rt}_\circ(\overrightarrow{\theta}(inv_\circ(x))\downarrow) \cap \mathtt{rt}_\circ(\overrightarrow{\theta}(Y)\downarrow) = \emptyset \wedge \overrightarrow{\theta}(x) \neq id_\circ$$

For brevity we will write this as a predicate: $\mathtt{Indelible}(x, Y)$.

This definition captures the notion that a particular root term cannot be cancelled out by other root terms. Cancellation only occurs when (after a substitution) a root term could be mapped to the identity or the inverse of another root term in the term. This

definition captures both of these possibilities. In particular, note that a root term in a ground term is always indelible.

We now need to introduce a special type of constraint to represent this property. This constraint might be something that we discover from directly inspecting a term, or it might be a constraint we impose on new variables that we introduce to the system.

We write $\mathit{Indelible}(x, Y)$ as the constraint analogue of the predicate $\texttt{Indelible}(x, Y)$. As we typically use this constraint after defining a term e.g. $x + Y$ or $x \cdot Y$ or $x \times Y$, we leave the operator implicit as it is obvious from context. Wherein we are matching a term against a *Indelible* condition, if the Y term is not present, we will consider it to be $id_\circ$ and $\mathit{Indelible}_\circ(x, id_\circ)$ always holds when x cannot unify with $id_\circ$.

We also introduce a constraint to represent a weaker condition, that the term cannot be cancelled, without requiring that it is not zero. We write this fact symbols as $\mathit{NoCancel}(x, Y)$. As $\mathit{Indelible}(x, Y) \implies \mathit{NoCancel}(x, Y)$, where a premise requires *NoCancel* we treat it as filled if the terms are *Indelible*.

Unfortunately we cannot automatically discover when these constraints hold in general due to the undecidable nature of unification in EQT_{DH}. Later, in section 6.7.5 we will see how we can make use of these constraints through rules that apply in particular special cases, as well as how these constraints are useful when introducing new variables. However, we can automatically check for the violation of these constraints:

$$\frac{\mathit{Indelible}_\circ(x, Y), \Gamma}{\bot}\,\mathcal{C}_{Sep,\bot} \qquad\qquad x \not\sqsubseteq \texttt{rt}_\circ((x \circ Y)\downarrow)$$

Figure 6.3: Rule $\mathcal{C}_{Sep,\bot}$

The soundness and completeness of $\mathcal{C}_{Sep,\bot}$ follows directly from the definition.

6.5 Introducing Indicators

In this section, we show how Dougherty and Guttman's definition of an indicator can be described purely as a function mapping terms to terms. We also introduce a number of definitions that will be helpful for reasoning about indicators in composite terms. We introduce our key constraint solving rules that allow Tamarin to deduce the basis set for a particular constraint system, as well as which indicators the adversary must have learned from the protocol.

6.5.1 Preliminary Constraints

We first define a number of constraints we will use frequently in this work. Firstly, a secrecy constraint:

Definition 49. We introduce the constraint $Secret(X)$ to mark when a term X must be secret from the adversary.

$$Secret(X) := \forall x, \#i.K(x)@i \implies \bot$$

We also introduce a special rule for reasoning about E-terms which we will need to represent a basis. This is a stronger secrecy requirement, the adversary must not be able to learn any E-term containing the secret term in question.

Definition 50. We introduce the constraint $SecretE(X)$ to mark when the family of all E-terms containing a particular ingredient should remain secret from the adversary forever.

$$\frac{SecretE(X), K(Y), Indelible(X, Y')}{\bot}\,\mathcal{C}_{\neg K_I, \bot} \qquad \begin{aligned} Y &: E \\ X &\sqsubseteq_I Y \\ Y &= X \cdot Y' \end{aligned}$$

Figure 6.4: Rule $\mathcal{C}_{\neg K_I, \bot}$

Finally, we introduce its negation, where the adversary must learn such a term.

Definition 51. We introduce the constraint $K_I(X)$ which represents the adversary learning an E-term containing X.

$$\frac{K_I(X), \Gamma}{K(X \times \theta_1 + \theta_2), Indelible(X, \theta_1), Indelible(X \times \theta_1, \theta_2), \Gamma}\,\mathcal{C}_{K_I}$$

Figure 6.5: Rule $\mathcal{C}_{K_I}$

Soundness and Completeness of $\mathcal{C}_{K_I}$ follows immediately from the definition.

6.5.2 Basis Introduction

In [74], Dougherty and Guttman introduce the notion of a basis set. A basis set contains exponent values created by the protocol which are secret from the adversary. In their formulation, the basis set is input by the protocol modeller. We show how we can derive the basis set from the protocol description using new constraint reduction rules.

In principle, any fresh NZE-term could be in the basis. However, we are only interested in NZE-terms which are useful to the adversary. We will see later in 6.5.3 how to determine whether a particular term is useful. For now, we simply state that such terms will be marked by the following constraint $PiEq(X, Y)$ which we introduce properly later in

section 6.5.3.2. Typically, Y will be a variable of sort NZE and X will be a fresh constant appearing in a $\mathsf{Fr}()$ fact.

Our basis introduction rule is naive and simply case splits on whether or not a particular term is in the basis.

$$\frac{\mathsf{Fr}(X : NZE)@i \wedge PiEq(\theta, X)@j, \Gamma}{\{K_I(X), NB(X, j), \Gamma\}, \{SecretE(X), B(X, j), \Gamma\}} \mathcal{C}_B$$

Figure 6.6: Rule $\mathcal{C}_B$

This rule is trivially sound and complete, as each case split is a negation of the other. In practice, this introduces a case split for every fresh value used as an exponent in the protocol. However, we can reasonably expect exponent values to either be secret or trivially discoverable through the adversary compromising an agent. Consequently, only one case will 'survive'.

These rules specify how basis sets behave with respect to time:

$$\frac{B(X, i) \wedge j < i}{B(X, j)} \mathcal{C}_{B,\leq} \qquad \frac{NB(X, i) \wedge j > i}{NB(X, j)} \mathcal{C}_{B,\geq} \qquad \frac{NB(X, i) \wedge B(X, i)}{\bot} \mathcal{C}_{B,\bot}$$

Figure 6.7: Rules $\mathcal{C}_{B,\leq}$, $\mathcal{C}_{B,\geq}$ and $\mathcal{C}_{B,\bot}$

That is: if a value is secret at timepoint i it is secret at all earlier timepoints. Likewise if it is known at timepoint i then it is known at all later timepoints. Furthermore, the same value cannot simultaneously be in and not in the basis.

6.5.2.1 Properties of Basis Sets

We refer to the set of elements in B at timepoint j as $B(j)$. This set consists of exponent elements which the adversary has not learned by timepoint j. We now show this set matches up with the definition of a basis set in [74].

Proposition 1. *Let B be a set of NZE ground terms at timepoint j as produced by the above process. The set of terms used by the adversary at the protocol by timepoint j is B-free.*

Proof. Assume not, then for some $b \in B(j)$ the adversary learnt some term α before j such that $\alpha = \beta \times b + \gamma$ with β, γ constants and indelible from b. But this contradicts the $SecretE(b)$ constraint. □

6.5.3 Indicator Conservation

In this section we generalise the definition of indicator given in [74]. We proceed in three parts. Firstly, we extend the definition to report the underlying indicator term, rather than an integer tuple describing the relationship to the basis set. Secondly, we allow for an arbitrary user defined equational theory, rather than a hardcoded combination of signatures, symmetric encryption and hash functions. Finally, we lift our new definition to support the partial evaluation of the indicator of a term, returning a set of constraints that must be solved in order to fully evaluate the term.

Definition 52. For X a root term and B, NB which are sets of *NZE* ground terms and disjoint we define $\Pi_{B,NB}(X) : DH \to DH$ as follows:

$$\Pi(X') = \begin{cases} exp(\Pi(X), \Pi(y)) & X' = exp(X, y) \\ \Pi(X) & X' = inv.(X) \\ \Pi(x) \times \Pi(y) & X' = x \times y \\ inv_{**}(\Pi(x)) & X' = inv_{**}(x) \\ id_{\times} & X' = h(X) \\ id_{\times} & X' : E \wedge X' \in NB \\ X' & X' : E \wedge X' \in B \\ X' & X' : G \wedge X' \text{is ground} \end{cases}$$

For brevity we omit B, NB where it is clear from context, typically this will be the timepoint at which the adversary wishes to learn some term. Where X is ground and all E-terms appear in $B \cup NB$, then $\Pi(X)$ is fully defined. In our constraint solving system, it will often be the case that X will not be ground, or $B \cup NB$ will not be complete. We could simply refuse to evaluate the function until we fully know X, but thanks to our recursive definition, it is possible to partially evaluate $\Pi(X)$. We discuss this shortly.

The motivation behind this definition is that it describes the "indicator" of a term. I.E., where elements in B are secret from the adversary, the result of $\Pi(X)$ is the minimum 'core' that the adversary must have received, in order to construct the term X. Of course, the adversary could have received the whole term X and not needed to construct anything, and this possibility is preserved.

Proposition 2. *Let B, NB be some complete basis set and let X, Y be normalised root terms with the same base G element. Then:*

$$\Pi_B(X) = \Pi_B(Y) \iff Ind_B(X) = Ind_B(Y)$$

Proof. Let g be the common base G element. Then $\Pi(X) = g^{\Pi(\alpha_i)}$ where $\alpha_i \in B \vee inv_{**}(\alpha_i) \in B$. Consequently both directions hold as they are clearly equivalent representations. □

The function symbol Π is parameterised by two sets B and NB. We identify these sets as the set of all $B(x,i)$ constraints and all $NB(y,i)$ constraints and write Π_i. Let X be a G root term and Y a G term.

$$\frac{K(X \cdot Y)@i, |\mathtt{rt}(X)| = 1, \mathit{Indelible}(X,Y), \Pi_i(X) \neq \mathit{id}.}{K(\theta_1 \cdot \Pi_i(X)^{\theta_2})@j, \mathit{PiEq}(\theta_2, \mathit{id}_\times), \mathit{Indelible}(\Pi_i(X)^{\theta_2}, \theta_1), j < i, \theta_1 : G, \theta_2 : \mathit{NZE}} \mathcal{C}_\Pi$$

Figure 6.8: Rule $\mathcal{C}_\Pi$

This rule states that where the adversary must learn a G root term X and this term is indelible from any other root terms, then the adversary must have first learned its indicator value from the protocol.

6.5.3.1 Correctness

Firstly, we require a generalisation of Dougherty and Guttman's Indicator Theorem discussed in section 6.2.2.1. In [74], this theorem is proven for a fixed equational theory which consists of hashing, symmetric encryption and digital signatures in the operational semantics of strand spaces. We note the proof of the Indicator Theorem does not make any use of the fixed equational theory beyond its disjointness from EQT_{DH}. Additionally, the requirements of the proof with respect to the operational semantics are simpler in our case as we moved the basis set from a fixed assumption to part of our constraint system. We conjecture this theorem holds for any equational theory which is disjoint from EQT_{DH} and for Tamarin's operational semantics and proceed on the assumption it does.

Theorem 8. *The rule $\mathcal{C}_\Pi$ is sound and complete.*

Proof. As the rule only adds a constraint, it is clear to see that if $\mathcal{C}_\Pi$ reduces Γ_1 to Γ_2 and Γ_1 has no solutions then Γ_2 has no solutions. Now we show that if Γ_1 has a solution, so does Γ_2. As Γ_1 has a solution,there is a sequence of protocol rules and message deduction rules such that the overall constraint system is satisfied.

Recalling our operational semantics, we have an important definition of the adversary knowledge. Let tr be a trace and i be a timepoint in that trace. If $K(x)@i$ for some term x, then it must be that $x \in \mathit{Gen}(AF \cup P \cup tr_i^{out})$, where AF is the set of fresh terms that the adversary generated, P is the set of public terms and tr_i^{out} is the set of terms appearing in Out facts up till timepoint i in the trace. This property holds from the definition of the adversary message deduction rules in [142].

Consequently by our generalised Indicator Conservation Theorem, if B is the set of E-terms which the adversary does not learn, then AK is a set where each $e : E \in T$ is B-free and hence we can apply part 2 of the theorem:

$$x \cdot Y \in \mathit{Gen}(AK) \text{ of sort } G \text{ and z } \in \mathit{Ind}_B(x \cdot Y) \implies \exists t \in AK \text{ s.t. } z \in \mathit{Ind}_B(t)$$

We know that x is not cancelled out by Y so consequently we can pick z_x which is the indicator corresponding to x. Therefore, $\exists t \in AK$ s.t. $z_x \in Ind_B(t)$. We now show that this t must satisfy the additional constraints we introduced with $\mathcal{C}_\Pi$.

Firstly, we notice that $z_x \in Ind_B(\theta_1 \cdot \Pi(x)^{\theta_2})$ as due to $Indelible(\Pi(x)^{\theta_2}, \theta_1)$ we have that $Ind_B(\Pi(x)^{\theta_2}) \subseteq Ind_B(\theta_1 \cdot \Pi(x)^{\theta_2})$ and that $Ind_B(\Pi(x)^{\theta_2}) = Ind_B(\Pi(x)) = Ind_B(x)$. The first step follows from the restriction that θ_2 not contain any basis values and the second step from that Π preserves indicator values.

Consequently, we can see that there is a substitution on θ_1, θ_2 such that

$$t = \theta_1 \cdot \Pi(x)^{\theta_2}$$

As t must contain $\Pi(x)^{\theta_2}$ as a root term in order to have the same indicator value, and any other root terms that t contains are satisfied by θ_1. The two restrictions on θ_1, θ_2 are simply that the latter root term is not cancelled out and that the indicator value is not changed. Consequently, if Γ_1 has a solution, then these additional constraints can also be satisfied.

□

6.5.3.2 Partial Evaluation

Often, we will want to apply this rule to a non-ground term or a fresh value which is not yet in B or NB. However, Π is not defined for these values. We resolve this by supporting partial evaluation of Π. Where $\Pi(x)$ is undefined, we introduce a new variable α and define $\Pi(x) := \alpha$. We then add $PiEq(\alpha, x)$ as a constraint. When x is a fresh term, this will trigger the basis introduction rule. We use the following rule for the case that x is a variable:

$$\frac{PiEq(\gamma, \alpha), \Gamma, \Pi(\alpha) \text{ is computable}}{\{\gamma \to \Pi(\alpha)\}\Gamma} \mathcal{C}_{\Pi, P.E.}$$

Figure 6.9: Rule $\mathcal{C}_{\Pi, P.E.}$

This rule simple updates the constraint system by substituting the correct value and dismissing the constraint.

6.6 Sourcing Indicators

The constraint solving rules in the previous section allow us to move from reasoning about terms the adversary must construct, to indicators the adversary must be able to extract from the protocol. In this section we provide constraint solving rules for reasoning about which DH-terms the adversary can extract from the protocol and which indicators are produced by the protocol.

6.6.1 Indicator Origination

Our indicator conservation rule $\mathcal{C}_{\Pi}$ will introduce goals of the form $K(\theta_1 \cdot X^{\theta_2})$ where X is some indicator value. As previously mentioned, we cannot use Tamarin's existing rules for handling *DH* terms, as they are based on unification using special construction and deconstruction rules for manipulating *DH* terms (See Section 2.3.3). However, due to our formulation of the indelible constraint we do not need to.

Instead, we provide two rules allowing the adversary to coerce a bitstring back into a group element or exponent. These rules will be the only possible origin points of *DH* terms.

$$[\ K(\mathtt{box}_G(X))\ ]\!\!\rightarrowtail\!\![\ K(X)\ ]$$
$$[\ K(\mathtt{box}_E(X))\ ]\!\!\rightarrowtail\!\![\ K(X)\ ]$$

Figure 6.10: Rules for deconstruction of *DH* terms

These boxed terms are in turn extracted from received message terms using deconstruction rules. Recall that we can perform unification on terms that do not make use of EQT_{DH}. Eventually, this will lead back to protocol rules. Where protocol rules do not use *DH* operators, we can exploit unification. However, where they make use of an exponentiation operator, we will need an alternative approach.

6.6.2 Indicator Construction

With our Indicator Origination rules, Tamarin can apply a sequence of standard intruder deconstruction rules to protocol output in order to learn the boxed term. This induces an equality constraint between the indicator the adversary wishes to learn and boxed term which was constructed by the protocol. In this section we introduce a rule which can partially solve this equality constraint through use of unification in a simpler equational theory.

Our rule works by considering the *DH* term constructed by the protocol and reducing it to a simplified form and attempting to unify it with the indicator under a simpler *DH*-theory which excludes $\cdot, +$ and $^{-1}$. This process cannot fully solve the equality, however, it does allow Tamarin to deduce further information about the inputs the protocol must of used to construct the indicator. This rule can then be applied again to the inputs. Eventually, an input must lead back to a Fresh or Public fact which contains an atomic constant and we can solve an equality between an atomic constant and a term with indelible constraints.

We now define two functions on our term algebra that we will use to move terms from our extended DH theory to a simplified one and back again.

Definition 53. We define the function $\texttt{simplify}() : G \times \mathcal{P}(NZE) \to G$ i.e. from normalised G-terms to normalised G-terms such that:

$$\texttt{simplify}(X)_{NB} = \begin{cases} \texttt{simplify}(Y) & X = inv.(Y) \\ \exp(Y, \texttt{simplify}(Z)) & X = Y^Z \\ \texttt{simplify}(Y) \times \texttt{simplify}(Z) & X = Y \times Z \\ id_\times & X = \mu(Y) \vee (X \text{ is ground } \wedge X \notin B) \\ X & \text{otherwise} \end{cases}$$

As with Π, we sometimes reference the basis set B by its timepoint i.e. $\texttt{simplify}()_i$.

Definition 54. We define the function $\texttt{generalise}() : \mathcal{S} \to \mathcal{S} \times \mathcal{P}(\mathcal{C})$ by applying the following mapping to each variable in the input substitution:

$$\texttt{generalise}(X) = \begin{cases} exp(X, \beta) \cdot \alpha, Indelible(\exp(X, \beta), \alpha), PiEq(\beta, id_\times)) & X : G \\ X \times \beta + \gamma, Indelible(X \times \beta, \gamma), PiEq(\beta, id_\times) & X : E \\ X & \text{otherwise} \end{cases}$$

where $\alpha : G$, $\beta : NZE$ and $\gamma : E$. We lift this to substitutions with multiple mappings in the usual way: composition of substitutions and the union of constraints.

Definition 55. We denote by T the order sorted signature and equational theory produced by restricting EQT_{DH} in the following ways:

- Removal of $\cdot$ and associated equations
- Removal of $+$ and associated equations
- Removal of μ and associated equations
- Removal of $^{-1}$ and associated equations

We can now introduce the rule:

$$\frac{\overbrace{\{Eq(\alpha, \theta_1 \cdot A^{\theta_2}), \Gamma\}}^{\Gamma'}, Indelible(A^{\theta_2}, \theta_1), PiEq(\theta_2, id_\times), \Pi(A) = A}{\{\texttt{generalise}_1(\sigma)\Gamma' \cup \texttt{generalise}_2(\sigma) | \sigma \in \texttt{Unify}_T(\texttt{simplify}(\alpha_i), A), \alpha_i \in \texttt{rt}(\alpha)\}} \mathcal{C}_{\mathcal{U}_G}$$

Figure 6.11: Rule $\mathcal{C}_{\mathcal{U}_G}$

Here we apply the substitution returned by $\texttt{generalise}()$ and append the constraints. This rule introduces a case split for every root term in the term constructed by the protocol and for every possible way that said root term could produce the target indicator.

Theorem 9. *Rule $\mathcal{C}_{\mathcal{U}_G}$ is complete.*

Proof. **(Sketch)**

Let us assume the constraint system has a solution prior to the application of $\mathcal{C}_{\mathcal{U}_G}$. We show that this solution is preserved. Let ψ be the substitution of a solution then to satisfy the equality constraint we must have that:

$$\psi(\alpha) = \psi(\theta_1 A^{\theta_2})$$

We first observe that $\exists \alpha_i$ such that $A \sqsubseteq_I \psi(\alpha_i)$ and $\alpha_i \in \mathtt{rt}(\alpha)$. This follows from the fact that A is a root term and $\mathtt{rt}(\psi(\alpha)) \subseteq \bigcup_i \mathtt{rt}(\psi(\alpha_i))$ (we show this in 6.7.5.2).

We now construct ψ' such that $\psi'(\mathtt{simplify}(\alpha_i)) = A$. We simply iterate through the variables that ψ' operates on where they are substituted for a term containing $\cdot$ or $+$ we select the correct argument which lead to the indicator value A. The other argument will produce a distinct root term and we replace it with the appropriate identity element. Additionally we replace any ground terms not in the basis with the appropriate identity element.

The result is that $\psi'(\mathtt{simplify}(\alpha_i)) = E$ and uses function symbols that are only present in our simplified equational theory T. Consequently, there must be a most-general-unifier σ such that $\sigma \in \mathtt{unify}_T(\mathtt{simplify}(\alpha_i), A)$ and σ is compatible with ψ'. I.E. $\exists \phi \in \mathcal{S}$ such that $\psi' = \phi\sigma$.

Furthermore, $\exists \phi'$ such that $\phi'(\mathtt{generalise}_)(\sigma) = \phi$. We construct ϕ' by the reverse of the winnowing process to construct ψ'.

□

Theorem 10. *Rule $\mathcal{C}_{\mathcal{U}_G}$ is sound.*

Proof. As the rule only introduces constraints, any resulting solutions also satisfy the initial constraint system. □

We demonstrate the intuition behind this rule with an example.

Example 5.

$$Eq(\alpha^x \beta^y, \theta_1 \cdot g^{ab^{\theta_2}})$$

where α, β, x, y are variables and a, b are ground *NZE*-terms in the basis.

Assume σ is some total substitution such that g^{ab} is an indicator contained in $\sigma(\alpha^x \beta^y)$. It follows that g^{ab} is contained within either $\sigma(\alpha^x)$ or $\sigma(\beta^y)$. Indicators cannot be produced by interacting root terms, only cancelled out and hence destroyed.

Consequently, we can use unification in a simplified theory to consider all possible ways an indicator can be produced. We then use $\mathtt{generalise}()$ to account for any additional terms that may also be present. This ensures we do not miss substitutions.

In our example, one possible outcome is that α 'provided' g^a, x 'provided' b and the other root term was not required (as it does not interact so we learn no information about it). Our final substitution is then $\{\alpha \to \alpha_1 \cdot g^{a\alpha_2}\} \cup \{x \to x_1 \times b + x_2\}$.

Consequently, we learn that b must be contained in x. This is significant, because whilst G terms typically contain complex structure, many protocols only use E terms in simple or even atomic forms. Commonly, E is a fixed atomic constant sourced from a random number generator, rather than a complex function of other protocol components.

If we go on to discover that x is drawn from a $\mathsf{Fr}()$ fact, we can safely throw away the variables introduced by `generalise`() and conclude this $\mathsf{Fr}()$ fact must have contained only x. In many cases this will lead to a contradiction.

In short, we use this rule to move from considering difficult unification questions around our *DH*-terms, to considering control flow questions.

We can also extend this rule to handle E-terms in a straightforward fashion:

$$\frac{Eq(\alpha, \theta_1 + A \times \theta_2), Indelible(A \times \theta_2, \theta_1), PiTerm(A), PiEq(\theta_2, id_\times)}{Eq(g^\alpha, g^{\theta_1} \cdot g^{A \times \theta_2}), Indelible(g^{A \times \theta_2}, \theta_1), PiEq(\theta_2, id_\times)} \mathcal{C}_{\mathcal{U}_E}$$

Figure 6.12: Rule $\mathcal{C}_{\mathcal{U}_E}$

We now show the correctness of $\mathcal{C}_{\mathcal{U}_E}$ by showing it can be expressed as an equivalent problem for $\mathcal{C}_{\mathcal{U}_G}$:

Theorem 11. *Let θ_1 and θ_2 be two ground E-terms. Then*

$$\theta_1 = \theta_2 \iff g^{\theta_1} = g^{\theta_2}$$

Where g is a fresh G-term not appearing in θ_1 or θ_2.

Proof. $\implies$ is trivial. To see the reverse, we consider θ_1 and θ_2 in normal form such that $\theta_1 \neq \theta_2$ but $g^{\theta_1} = g^{\theta_2}$. This implies that there must be some reduction which can be applied to either g^{θ_1} or g^{θ_2} but not θ_1 or θ_2. There are only four such rules:

$$g^{id(\times)} \to G$$
$$g^{id_+} \to id(\cdot)$$
$$g^{inv_+(\alpha)} \to inv.(g^\alpha)$$
$$g^{\alpha+\beta} \to g^\alpha \cdot g^\beta$$

These rules can also be seen as identities and consequently there must be a series of applications of them such that g^{θ_1} is syntactically equivalent to g^{θ_2}. The first two reductions cannot be of use as they immediately imply $\theta_1 = \theta_2$ so we consider the last two.

We can apply the fourth rule until the term θ_1 no longer contains any addition symbols and then apply the third rule until no subterm contains any inv_+ function symbols. However, these reductions unlock no further transformations on θ_1. □

6.7 Equalities and Contradictions

In this section we introduce a set of rules for reasoning about indicators and equality constraints. These rules compose together to either make progress on equality constraints by providing substitutions or else derive contradictions which close open constraint systems.

6.7.1 Adversarial Knowledge of Indicators

Let x be an E-term.

$$\frac{PiEq(\theta_1, x), K_I(x), \Gamma}{\{\theta_1 \to id_\times\}\Gamma}\,\mathcal{C}_{\Pi,K}$$

Figure 6.13: Rule $\mathcal{C}_{\Pi,K}$

When evaluating the indicator of a term, we may come across a value provided by the adversary in the exponent. This value will be a variable and hence cannot be introduced to the B or NB sets, however as it is known to the adversary, it cannot have any non-identity indicator value. This means we do not need to fully learn a term in order to finalise its indicator value. Correctness follows immediately from the definition of an indicator.

6.7.2 Indicator Inequalities

Let A and B be G-terms.

$$\frac{Eq(A, B) \wedge \Pi(A) \neq \Pi(B)}{\bot}\,\mathcal{C}_{Eq,\bot}$$

Figure 6.14: Rule $\mathcal{C}_{Eq,\bot}$

Correctness follows from the observation that: $A = B \implies \Pi(A) = \Pi(B)$. Taking the contrapositive we have that $\Pi(A) \neq \Pi(B) \implies A \neq B$.

6.7.3 Instantiation

$$\frac{Eq(\theta, x), \Gamma \wedge \theta \in \mathcal{V} \wedge \theta \notin Vars(x)}{\{\theta \to x\}\Gamma}\,\mathcal{C}_I$$

Figure 6.15: Rule $\mathcal{C}_I$

As before, this rule lets us solve equalities in very particular circumstances, typically when a premise and a conclusion are being matched. This rule is correct as any alternative unifier ψ of θ and x has $\psi(\theta) = \psi(x)$. Let ϕ be the substitution $\{\theta \to x\}$. Then $\psi(\phi(\theta)) = \psi(x)$ and consequently $\psi\phi$ is also a unifier of x and θ.

6.7.4 Atomic Unification

We introduce two rules to handle unification with atomic ground terms.

$$\frac{(\mathsf{Fr}(x) \vee \mathsf{Pub}(x)), Eq(x, \theta_1 \times a + \theta_2), Indelible(\theta_1 \times a, \theta_2), Indelible(a, \theta_1), \Gamma}{\{a \to x\}\{\theta_1 \to id_\times\}\{\theta_2 \to id_+\}\Gamma} \mathcal{C}_{A,G}$$

$$\frac{(\mathsf{Fr}(x) \vee \mathsf{Pub}(x)), Eq(x, a^{\theta_1} \cdot \theta_2), Indelible(a^{\theta_1}, \theta_2), \Gamma}{\{a \to x\}\{\theta_1 \to id_\times\}\{\theta_2 \to id.\}\Gamma} \mathcal{C}_{A,E}$$

Figure 6.16: Rules $\mathcal{C}_{A,E}$ and $\mathcal{C}_{A,G}$

These rules prune excess variables, typically introduced by the `generalise`() function. This rules follow immediately from the definition of fresh and public terms as atomic names and the definition of indelible.

6.7.5 Separable Introduction

In certain circumstances we can deduce when a root term is indelible by inspecting the structure of the term. In particular, the use of μ to hash group elements to exponents induces some indelible constraints on the resulting root terms which we capture here.

Definition 56. For x, Y G-terms, the predicate `manualSep`(x, Y) holds if:

$$x = g^{a\mu(\alpha)} \qquad\qquad Y = y_1 \cdot \ldots \cdot y_n$$

and $\forall i \in 1, \ldots, n$:

$$y_i = g^{b_i\mu(\alpha)} \vee y_i = \alpha^{c_i}$$

where α is a variable., g, a, b_i, c_i are constants or functions of constants such that $g \neq a \neq b_i \neq c_i$ and none use the function symbol μ.

We now apply this predicate in a rule:

$$\frac{K(x \cdot Y), \Gamma}{K(x \cdot Y), Indelible(x, Y), \Gamma} \mathcal{C}_{Sep,I} \qquad \text{if } \texttt{manualSep}(x, Y)$$

Figure 6.17: Rule $\mathcal{C}_{Sep,I}$

To prove $\mathcal{C}_{Sep,I}$ sound and complete, we must establish a number of preliminary results first. Firstly, we show how indelible constraints can be described as a unification problem. Next, we show how root terms are produced by substitutions. Finally, we show how indelible constraints can be reduced to pairwise unification between root terms. This leads to our final result - correctness of the manual predicate for introducing indelible constraints.

6.7.5.1 Relating Indelible Constraints to Unification

We first translate indelible into a unification problem with additional constraints.

Lemma 1. *Indelible*$(x)(Y) \iff \forall \phi \in \texttt{Unify}(\theta \cdot inv(x), Y)$ *it holds that* $\phi(x) \in \texttt{rt}(\phi(\theta))$ *and additionally that* $\texttt{Unify}(x, id.) = \emptyset$

Proof. We show the implication first. Invert the conclusion and assume some ϕ does exist. Then we have by indelible constraints that

$$\forall \omega \ \texttt{rt}(\omega(inv(x))) \nsubseteq \texttt{rt}(\omega(Y))$$

However we have that for some ϕ, $\phi(\theta \cdot inv(x)) = \phi(Y)$ which means that

$$\forall a \in \texttt{rt}(\phi(\theta \cdot inv(x))) \exists ! b \in \texttt{rt}(\phi(Y)) \text{ s.t } a = b$$

But as $\texttt{rt}(\phi(inv(x))) \nsubseteq \texttt{rt}(\phi(Y))$, we must have that $\texttt{rt}(\phi(inv(x))) \nsubseteq \texttt{rt}(\phi(\theta \cdot inv(x)))$. But then $\texttt{rt}(\phi(x)) \subseteq \texttt{rt}(\phi(\theta))$ and we have a contradiction.

Now we show the reverse. Again we assume the premise holds but the conclusion does not, i.e. the terms are not indelible. Consequently $\exists \phi$ s.t. $\texttt{rt}(\phi(inv(x))) \subseteq \texttt{rt}(\phi(Y))$. Consequently we can extend ϕ to act on the free variable θ such that $\phi'(\theta \cdot inv(x)) = \phi'(Y)$ by simply setting θ to be the left over terms of $\phi(Y)$. And hence we have a contradiction. □

This equivalent formulation is not mechanically tractable as we have no efficient or effective unification algorithm for the equational theory EQT_{DH}.

6.7.5.2 Producing Root Terms

Next, we consider how root terms are produced under substitution. In particular, we show a relationship between the images of individual root terms under a substitution and the image of the composition of those root terms under a substitution.

Lemma 2.

$$\forall \phi, \texttt{rt}(\phi(y_1 \cdot y_2)) \subseteq \texttt{rt}(\phi(y_1)) \cup \texttt{rt}(\phi(y_2)) \cup \{id.\}$$

Proof. Assume not, then for some ϕ there is some $a \in \texttt{rt}(\phi(y_1 \cdot y_2))$ but $a \notin \texttt{rt}(\phi(y_1))$, $\texttt{rt}(\phi(y_1))$, $\{id\}$. However, when we normalise the term $y_1 \cdot y_2$ we can consider it as a tree with $\cdot$ as the root and y_1, y_2 as two sub trees. If we consider the substitution ϕ on the subtree y_1 we can again normalise it. Furthermore, if we normalise $\phi(y_1)$ and $\phi(y_2)$, then there is only one further reduction possible when we consider the term $\phi(y_1) \cdot \phi(y_2)$ which is when $\phi(y_1) = inv.(\phi(y_2))$ in which case the term is equal to $id.$. □

Corollary 1.

$$\forall \phi, \texttt{rt}(\phi(y_1 \cdot \ldots \cdot y_n)) \subseteq \texttt{rt}(\phi(y_1)) \cup \ldots \cup \texttt{rt}(\phi(y_n)) \cup \{id\}$$

6.7.5.3 Pairwise Unification

We now apply our insights regarding the construction of root terms to move from a unification problem over a composite term to a pairwise unification problem over each root term.

Definition 57. We say a root term x is a **singleton**, if, for all substitutions ϕ, $|\mathtt{rt}(\phi(x))| = 1$ and $\mathtt{unify}(x, id.) = \emptyset$

This is true when x is ground, or when any variable in x is contained by a μ function symbol.

Lemma 3. *When x is a singleton, then $\forall i \in 1, \ldots, n$ if $\forall \phi \in \mathtt{unify}(\theta \cdot inv(x), y_i)$ it holds that $x \in \mathtt{rt}(\phi(\theta)) \implies \forall \phi \in \mathtt{unify}(\theta \cdot inv(x), y_1 \cdot \ldots \cdot y_n)$ it holds that $x \in \mathtt{rt}(\phi(\theta))$*

Proof. We use the contrapositive and hence have a substitution ϕ between $\theta \cdot inv(x)$ and Y. As x is a singleton, we know that $|\mathtt{rt}(\phi(x))| = 1$ and that by our earlier lemma $\mathtt{rt}(\phi(x) \subseteq \mathtt{rt}(\phi(Y)) \subseteq \mathtt{rt}(\phi(y_1)) \cup \ldots \cup \mathtt{rt}(\phi(y_n))$ Note: This is why we need to exclude the identity! As $\mathtt{rt}(\phi(x))$ only has one element it must belong to $\mathtt{rt}(\phi(y_i))$ for some i. Also as θ is not in x or Y, we can take the mapping on other variables such that $\mathtt{rt}(\phi(x)) \subseteq \mathtt{rt}(\phi(y_i))$ and use θ to equal any remaining terms. □

Definition 58.

$$\forall \phi \in \mathtt{unify}(\theta \cdot inv(x), y_i) \text{ it holds that } x \in \mathtt{rt}(\phi(\theta))$$

If the above condition holds, we say that $\mathtt{unify}(\theta \cdot inv(x), y_i)$ only contains **trivial unifiers**.

6.7.5.4 Coerced Exponents and Trivial Unifiers

Finally, we use our new sufficient condition for indelible constraints to establish some cases in which it holds by checking there are only trivial unifiers.

Lemma 4. *For θ a G-variable, x a E-variable a, b ground terms not involving μ with $a \neq b$: $\mathtt{unify}(\theta \cdot g^{a\mu(x)}, g^{b\mu(x)})$ contains only trivial unifiers*

Proof. By our earlier lemma, this is the same as showing that there is no substitution ϕ such that $\mathtt{rt}(\phi(g^{a\mu(x)})) \subseteq \mathtt{rt}(\phi(g^{b\mu(x)}))$. Considering the two terms as tree, we know that the terms are equal if and only if they have the same normal form. However, no matter the substitution on x, that $\mu(x)$ cannot cancel or even alter the term a or b because neither a nor b include a μ term and hence no reduction can occur. Consequently, no ϕ can exist without using θ and is hence a trivial unifier. □

Lemma 5. *For θ a G-variable, x a E-variable a, b ground terms not involving μ with $a \neq b$: $\mathtt{unify}(\theta \cdot g^{a\mu(x)}, x^b)$ contains only trivial unifiers.*

Proof. As before, we note that there must be ϕ such that $\mathtt{rt}(\phi(x^b)) \subseteq \mathtt{rt}(\phi(g^{a\mu(x)}))$. However, as a does not contain μ it cannot cancel out. Equally neither does b so it cannot contribute a μ. Hence y must contain $h(inv(y))$ as a subterm which cannot happen in a finite term. □

6.7.5.5 Correctness of $\mathcal{C}_{Sep,I}$

Theorem 12. *$\mathcal{C}_{Sep,I}$ is correct*

Proof. As $\mathcal{C}_{Sep,I}$ only introduces indelible constraints, it suffices to show the introduced constraints are correct. This follows directly from the results we have proven in this section. In particular, $\mathcal{C}_{Sep,I}$ establishes that for each root term, one of the two cases of trivial unifiers applies and then the indelible property follows by our results on pairwise unification. □

Our rule applies the observation that variables contained within μ functions cannot 'escape' and secondly that $\mu(\alpha)$ exponent can be use to prevent cancellation from a α root term. Any protocol making use of the group operation must have some rationale for why the combined terms cannot cancel out (otherwise the adversary would be able to mount an attack). From this perspective, our rule is capturing the 'purpose' of μ in a generic fashion.

6.8 Summary of Constraint Solving Rules

In the preceding section we have introduced a number of constraint solving rules. Our rules fall in to three broad categories. Firstly, we have rules like $\mathcal{C}_{\Pi}$, $\mathcal{C}_{Sep,I}$ and $\mathcal{C}_B$ which correspond to high level invariants. Secondly we have rules such as $\mathcal{C}_{\mathcal{U}_G}$, $\mathcal{C}_{A,G}$ and $\mathcal{C}_I$ which solve equality constraints in special cases. Finally, we have bookkeeping rules such as $\mathcal{C}_{B,\leq}$, $\mathcal{C}_{\Pi,P.E.}$ and $\mathcal{C}_{Eq,\perp}$.

Typically, solving a constraint system proceeds as follows: We first identify some *DH*-term the adversary must learn. Using our special case rules for solving equality constraints, we construct some part of the term. We then apply an indelible constraint through analysis of the term structure. This is typically feasible because a secure protocol must constrain how *DH* terms are constructed.

As we now want to apply the insights of Dougherty and Guttman and use the indicator constraints, we need to discover a basis. This typically involves a number of case splits (for each exponent in the term). With the basis discovered for each case we can apply the indicator theorem through our constraints and discover G and E terms that the adversary must have learned.

We now use our equality rules to reason about what terms the protocol must have produced. Usually, we can follow back each target *DH* term to some originating rule in the

Rule	Description	Section
$\mathcal{C}_{\approx}$	Solves equality constraints between M-terms which contain DH subterms.	6.4.2
$\mathcal{C}_{Sep,\bot}$	Checks for violated indelible constraints	6.4.3
$\mathcal{C}_{\neg K_I,\bot}$	Checks for violated basis constraints	6.5.1
$\mathcal{C}_{K_I}$	Expresses a K_I constraint in terms of adversary knowledge and separability.	6.5.1
$\mathcal{C}_{B}$	Case splits on whether a fresh E-term is in the basis	6.5.2
$\mathcal{C}_{B,\leq}$	Allows Basis Sets to be translated backwards in time.	6.5.2
$\mathcal{C}_{B,\geq}$	Allows Non-Basis Sets to be translated forwards in time.	6.5.2
$\mathcal{C}_{B,\bot}$	Checks for elements which both in the Basis and Non-Basis sets at the same timepoint.	6.5.2
$\mathcal{C}_{\Pi}$	Expresses a G-term the adversary must learn in terms of its indicator value.	6.5.3
$\mathcal{C}_{\Pi,P.E.}$	Evaluates Π for a term as part of a partial evaluation.	6.5.3.2
$\mathcal{C}_{\mathcal{U}_G}$	Partially solves an equality constraint for G-terms.	6.6.2
$\mathcal{C}_{\mathcal{U}_E}$	Partially solves an equality constraint for E-terms.	6.6.2
$\mathcal{C}_{\Pi,K}$	Evaluates Π for a variable known to the adversary.	6.7.1
$\mathcal{C}_{Eq,\bot}$	Checks that Equality and Π-Equality constraint agree.	6.7.2
$\mathcal{C}_{I}$	Solves Equality Constraints where one side is a variable.	6.7.3
$\mathcal{C}_{A,G}$	Solves Equality Constraints where one side is a G-constant.	6.7.4
$\mathcal{C}_{A,E}$	Solves Equality Constraints where one side is a E-constant.	6.7.4
$\mathcal{C}_{Sep,I}$	Introduces Indelible Constraints where known to hold.	6.7.5

Table 6.1: Summary of our constraint solving rules, their purpose and the section they were introduced

protocol and discover a contradiction, i.e. the protocol would not construct that particular G term or reveal a E term (at least, not without invoking some other relevant constraint). This ultimately leads to a contradiction and if every case results in a contradiction we arrive at a proof.

6.9 Case Studies

In this section we demonstrate our constraint solving rules. We prove, through hand-worked application of our constraint solving rules, that the original MQV protocol enjoys key secrecy and weak perfect forward secrecy. We further conjecture that that our rules are sufficient to also prove the absence of unknown key share attacks in HMQV and security after session key compromise in MQV, but do not show this here.

6.9.1 Well Formedness Conditions

We discuss three general recommendations on protocol specification and justify each one. Note that these recommendations are not mandatory and deviations from them do not

necessarily lead to an incorrect result.

Use of Sorts The protocol should not use the sort *NZE* unless applied to a variable appearing inside a *Fr* fact in the same rule. This is because *NZE* is a sort for terms which can never be zero and hence can never contain the symbol +. The correct way to enforce a particular value is non-zero in valid traces is to add a Inequality action fact in the usual manner. This expresses that the term might contain the symbol + but in a ground instantiation is not equal to id_+. This does have the consequence that if a protocol is going to invert a term, it must do so "early". For example $inv_{**}(x + y + z)$ is not a valid term because $x + y + z$ is necessarily of sort E whereas $inv_{**} : NZE \rightarrow NZE$. However the protocol can instead invert each component term and add them together. Future work could revisit how to handle the case where the protocol wishes to invert a received term. Typically there must be an 'abort' case for handling the situation where the terms cancel to the identity.

Premise Terms Where a rule contains a Premise with a *DH*-term, the term should be represented as a single variable. This is a reasonable restriction as in practice you cannot "look inside" a received group element or integer, you can only test for equality with another term. Secondly, it prevents the protocol from breaking the discrete log assumption and extracting indicator values it should not have access to.

6.9.2 MQV

MQV was originally published in 1995 was standardised by NIST, as well as being included in the NSA's Cipher Suite B recommendations. The original paper [124] did not provide security proofs but did argue that MQV was an authenticated key exchange protocol with weak forward secrecy and resistance to both unknown key share attacks and key compromise impersonation attacks. For a full discussion of these security properties, [66] is an excellent resource. We consider two properties, firstly that honest parties who engage in a session with another uncompromised participant derive a shared key which is secret from the adversary. Secondly, that if two honest parties agree a key without the adversary interfering in the session, the resulting session key is secure even if the participant's long term secrets are later compromised.

Recall that μ is a function that maps group elements to exponents. The key derivation is symmetric, Alice computes a secret value:

$$f_A = e_A + \mu(g^{e_A}) \times s_a$$

and a public group element for Bob:

$$F_B = E_B \cdot (S_B)^{\mu(E_B)}$$

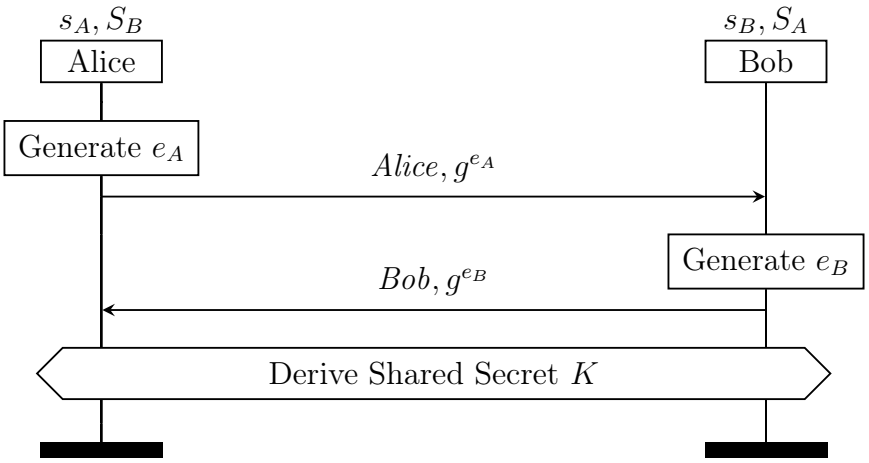


Figure 6.18: Protocol diagram for MQV

This gives the overall key:

$$K_{AB} = {F_B}^{f_A} = (E_B \cdot (S_B^{\mu(E_B)}))^{e_A + \mu(g^{e_A}) \times s_A}$$

In the event that both parties acted honestly and the adversary did not interfere, the resulting key is then:

$$K_{AB} = g^{e_A e_B} \cdot g^{s_A e_B \mu(g^{e_A})} \cdot g^{e_A s_B \mu(g^{e_B})} \cdot g^{s_A s_B \mu(g^{e_B}) \mu(g^{e_A})}$$

MQV employs a four way binding between the ephemeral and static public keys of each party, and rather than use computationally expensive signatures to pair static-ephemeral values, MQV uses “implicit signatures" $\mu(g^e)$ which are public keys which have been coerced into exponents.

In 2001, Kaliski published an unknown key share attack on MQV [97]. An unknown key share occurs when two honest parties agree upon a key, that key is secret from the adversary, but one of the parties is mistaken about the identity of the other. For example, Alice correctly believes she has agreed a key with Bob, whereas Bob believes he has agreed a key with Eve (an identity chosen by the attacker), although he has in fact agreed it with Alice. Although the attacker controls the identity of the party, the attacker doesn't know the key and consequently cannot otherwise manipulate the session beyond these identities. This type of attack is often problematic for higher level protocols with state.

6.9.2.1 Formal Model

We now desribe a model of this protocol in Tamarin, using our new extension.

As MQV is a stateless 2-message protocol its corresponding formal model is straightforward. We have two rules for the registration of identities which allow honest parties to be created and the adversary to register maliciously controlled parties with their choice of key.

$$\frac{\mathsf{Fr}(id), \mathsf{Fr}(s)}{!\mathsf{Ltk}(id, s), !\mathsf{PubKey}(id, g^s), \mathsf{Out}(id, g^s)}\ \texttt{RegisterKey}, \mathsf{Honest}(id)$$

$$\frac{\mathsf{In}(id), \mathsf{In}(s)}{!\mathsf{Ltk}(id, s), !\mathsf{PubKey}(id, g^s)}\ \texttt{MaliciousKey}$$

$$\frac{!\mathsf{Ltk}(i, s_i), !\mathsf{PubKey}(r, S_r), \mathsf{Fr}(e_i)}{\mathsf{Out}(i, g^{e_i}), \mathsf{Init}(i, r, e_i)}\ \texttt{Initiator1}$$

$$\frac{!\mathsf{Ltk}(r, s_r), !\mathsf{PubKey}(i, S_i), \mathsf{Fr}(e_r), \mathsf{In}(i, E_i)}{\mathsf{Out}(r, g^{e_r})}\ \texttt{Responder1}, \mathsf{AgreeKey}(r, i, k_r)$$

$$\frac{!\mathsf{Ltk}(i, s_i), !\mathsf{PubKey}(r, S_r), \mathsf{In}(r, E_r), \mathsf{Init}(i, r, e_i)}{}\ \texttt{Initiator2}, \mathsf{AgreeKey}(i, r, k_i)$$

$$k_r := {E_i}^{e_r} \cdot {S_i}^{e_r \mu(E_i)} \cdot {E_i}^{s_r \mu(g^{e_r})} \cdot {S_i}^{s_r \mu(g^{e_r}) \mu(E_i)}$$
$$k_i := {E_r}^{e_i} \cdot {S_r}^{e_i \mu(E_r)} \cdot {E_r}^{s_i \mu(g^{e_i})} \cdot {S_r}^{s_i \mu(g^{e_i}) \mu(E_r)}$$

Figure 6.19: TAMARIN model of MQV with suppressed $\texttt{box}_E, \texttt{box}_G$ functions for clarity.

The remaining three rules model the actions of the initiator and the responder. We expose an action AgreeKey which denotes when an agent marks a handshake as concluded and who the agent believes the session is shared with and the corresponding key.

6.9.2.2 Key Secrecy

$$\mathsf{AgreeKey}(A, B, k) \wedge \mathsf{Honest}(A) \wedge \mathsf{Honest}(B) \implies \mathit{Secret}(k)$$

This lemma states that key secrecy holds, for any honest party talking to another honest party. We now show how to derive a proof of this using our new constraint solving rules.

Step 1 We form the initial constraint system using TAMARIN's existing constraint solving rules which transform the lemma into a satisfaction problem. TAMARIN looks for a trace in which the premise of the lemma holds, but the conclusion does not. We have suppressed the timepoints, as they are not relevant to our discussion. In the figure below, we show the four open unsolved constraints that form our initial constraint system.

$\mathsf{Honest}(a)$ $\quad$ $\mathsf{Honest}(b)$

$\mathsf{AgreeKey}(a, b, k)$ $\quad$ $\mathsf{K}(k)$

Figure 6.20: MQV - Key Secrecy: Constraint System after Step 1

Step 2 We consider the source of the $\mathsf{AgreeKey}(a, b, k)$ fact. There are two possible sources, we now discuss the case it arises from the `initiatior2` rule, leaving the `responder1` case open[5]. We have also solved the equality induced between k and k_i with the instantiation rule $\mathcal{C}_I$. The a and b variables have been renamed for clarity.

In the figure below, we show the $\mathsf{AgreeKey}(a, b, k)$ from Step 1 has been satisfied by the `initiatior2` rule. This premises of that rule are now unfilled constraints themselves.

$$\mathsf{Honest}(i) \qquad \mathsf{Honest}(r)$$

$$\mathsf{K}(k_i)$$

$$\frac{\mathsf{!Ltk}(i, s_i), \mathsf{!PubKey}(r, S_r), \mathsf{In}(r, E_r), \mathsf{Init}(i, r, e_i)}{\mathsf{AgreeKey}(i, r, k_i)}$$

Figure 6.21: MQV - Key Secrecy: Constraint System after Step 2. k_i is defined in Figure 6.19

Step 3 We now fill the open premises and in the process satisfy the $\mathsf{Honest}(i)$ and $\mathsf{Honest}(r)$ constraints. All satisfied actions have been suppressed. In each case we are able to either unify the M-terms or solve the DH equality using $\mathcal{C}_I$. No additional cases are introduced in this step. Notice the only remaining variable in the 'key' term is E_r.

In the figure below, we have introduced rules which have conclusion facts that satisfy our open premise facts. We highlight which rule instance conclusion is filling which rule instance premise with a single line. Notice that the $\mathsf{Fr}()$ facts are not expanded as they can only be filled with a unique atomic constant and hence it would be redundant to unfold them further.

[5]We will not discuss it further here, but it proceeds much as the initiator case

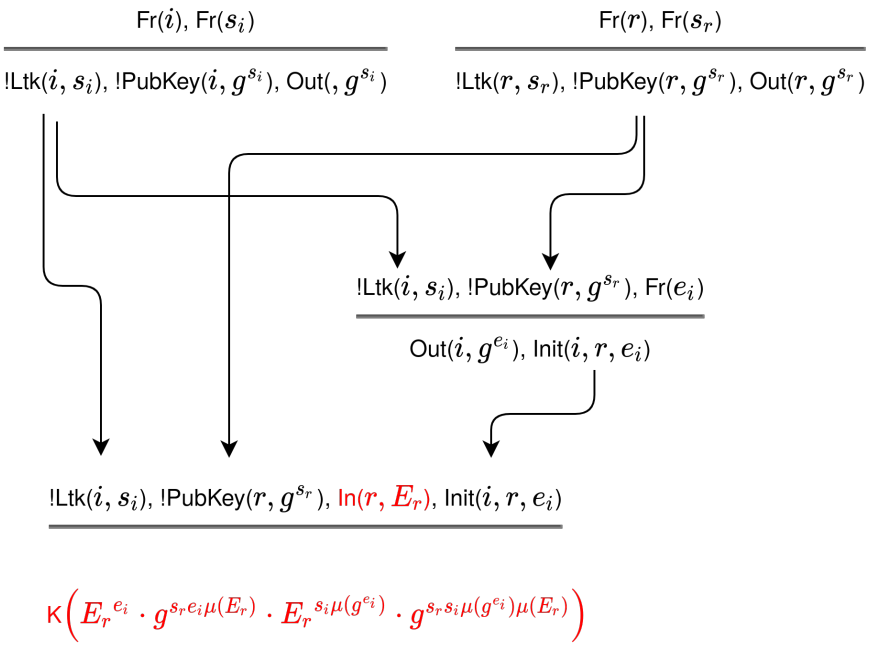


Figure 6.22: MQV - Key Secrecy: Constraint System after Step 3. The remaining unsatisfied constraints are highlighted in red. Tfhe 'key' term has been written out in its normal form.

Step 4 We apply $\mathcal{C}_{Sep,I}$ to the 'key' term to introduce the constraint:

$$Indelible(g^{s_r s_i \mu(g^{e_i})\mu(E_r)}, {E_r}^{e_i} \cdot g^{s_r e_i \mu(E_r)} \cdot {E_r}^{s_i \mu(g^{e_i})})$$

This allows $\mathcal{C}_\Pi$ to trigger, evaluating $\Pi_i(g^{s_r s_i \mu(g^{e_i})\mu(E_r)})$. At this point, no elements have been added to the basis, so this triggers a partial evaluation and the introduction of $PiEq(,)$ constraints for s_i, s_r. The $\mathcal{C}_B$ rule resolves these constraints, but produces four cases depending on whether these two values are in the basis or not. All the cases involving s_i or s_r not being in the basis collapse immediately as there are no valid sources for these terms. In the remaining case, the partial evaluation concludes and $\mathcal{C}_\Pi$ introduces the following constraints:

$$K(\theta_1 \cdot g^{s_r s_i \theta_2}), PiEq(\theta_2, id_\times), Indelible(\Pi_i(x)^{\theta_2}, \theta_1)$$

Step 5 The backwards search for this term leads to three possible sources of G-terms. It was either produced from `Initiator1` or `Responder1` in the position of an ephemeral key, or it was produced from `RegisterKey` as a static key. In each of these three cases, an equality fact is introduced between the G-term output by the rule and the indicator term. This is the first non-trivial equality that we need to solve and we cannot resolve it with $\mathcal{C}_I$ as neither term is a variable.

However, we can apply $\mathcal{C}_{\mathcal{U}_G}$ to make progress on this equality. Each unification is of the same form for some θ:

$$\texttt{Unify}_{DH}(g^{s_i s_r}, g^{\theta})$$

$$\frac{!Ltk(id, s)}{Out(s)}\,\mathsf{Compromise}(id)$$

Figure 6.23: Additional Compromise Rule for MQV Protocol

This has the most general unifier $\{\theta \to s_i s_r\}$. Generalisation is performed on this substitution and consequently we obtain the constraints:

$$Eq(\theta, \alpha_1 \times s_i \times s_r + \alpha), Indelible(\alpha_1 \times s_i \times s_r, \alpha)$$

Step 6 Finally, we step back to the sources of θ which is $\mathsf{Fr}(\theta)$ in each case. We solve this equality with the $\mathcal{C}_{A,E}$ rule which has no valid solutions. Consequently the constraint system has no open cases and thus no possible solution, resulting in a proof of the lemma. □

6.9.2.3 Weak Perfect Forward Secrecy

wPFS means that as long as the adversary did not interfere in a session, later compromise of the participants does not allow the adversary to recover the key material for the session. We extend the label AgreeKey to capture the ephemerals used in the exchange. We add the additional rule:

This allows the adversary to compromise the long term key of a participant. The weak perfect forward secrecy lemma is then:

$$\begin{aligned}\mathsf{AgreeKey}(i, r, E_i, E_r, k)@i \wedge \mathsf{AgreeKey}(r, i, E_r, E_i, k)@j \wedge \mathsf{Honest}(i) \wedge \mathsf{Honest}(r) \wedge \\ (\forall m.\mathsf{Compromise}(i)@m \implies i < m) \wedge \\ (\forall n.\mathsf{Compromise}(r)@n \implies j < n) \\ \implies Secret(k)\end{aligned}$$

Here, we encode the weak perfect forward secrecy requirement by requiring that the two parties agree on the exchanged ephemerals for the session in question. The adversary is permitted to compromise the participants after the session concludes.

Step 1 We create this system in the usual fashion and solve sources with $\mathcal{C}_I$ where possible. This brings us to the following constraint system:

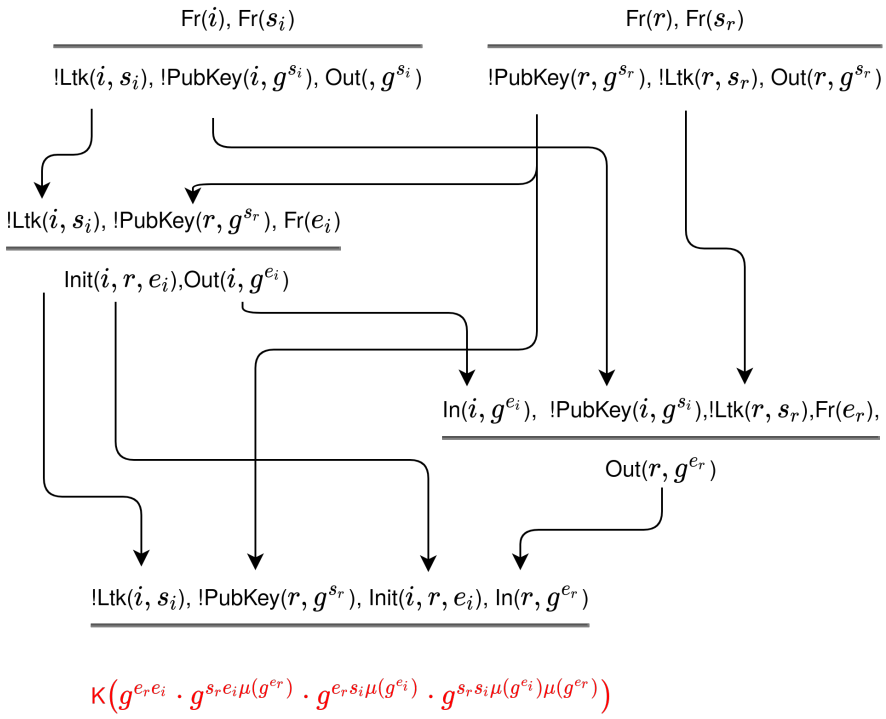


Figure 6.24: MQV - wPFS: Constraint System after Step 1

Step 2 The next rule that then applies is $\mathcal{C}_{\Pi}$. As before, this results in partial evaluation and now we have a case split over which of the terms s_i, s_r, e_i, e_r are in the basis (B). The only surviving case is that e_i, e_r are in the basis and s_i, s_r are not. Each constituent root term is separable from the others as the term is ground. However we focus on $g^{e_i e_r}$ and introduce the following constraints:

$$K^{\downarrow}(\theta_1 \cdot g^{e_r e_i \theta_2}), PiEq(\theta_2, id_{\times}), Indelible(\Pi_i(x)^{\theta_2}, \theta_1)$$

Step 3 To finish the case, we must now check that this indicator cannot be learned from the protocol. This happens in much the same fashion as the previous example, where we use $\mathcal{C}_{\mathcal{U}_G}$, $\mathcal{C}_{\mathcal{U}_E}$ and backtrack until we discover that no rule can produce such a term using $\mathcal{C}_{A,G}$, $\mathcal{C}_{A,E}$. □

6.9.2.4 Summary

In this section we have discussed two security properties of the simple MQV protocol and show how they can be proven using our new constraint solving rules. We abbreviated the application of Tamarin's existing rules. Additionally we did not discuss the many cases that arise when detailing with the backwards search using our unification results. Nevertheless, we feel our presentation illustrates the utility and flexibility of our approach and motivates its implementation in order to further verify its practicality.

6.10 Future Work

We now consider various areas of future work. Our discussion ranges from immediate items such as the refinement of our proofs and the implementation of our constraint solving rules in Tamarin to more open ended extensions regarding equality constraints, attack finding and non-prime order groups.

6.10.1 Proofs of Correctness

For each of our constraint solving rules, we have provided proof sketches of their soundness and completeness in our appendices. However, these sketches will require significant effort to develop into formal proofs suitable for peer review.

6.10.2 Implementation and Evaluation

Another area of continuing and future work is to develop an implementation of our constraint solving rules in order to test the real world performance of our design. We anticipate a straight forward integration with the Tamarin Prover. However, as our analysis is necessarily undecidable we may need to consider additional heuristics and further invariants to ensure reasonable performance in practice. Without a functional implementation, developing heuristics is a nigh-impossible task.

It would also be interesting to see if these constraint solving rules can aid analysis of protocols which do not use the group operation directly. Whilst Tamarin's existing approach can analyse such scenarios, our additional constraint solving rules may improve performance further and avoid unnecessary computation. In principle, it should be possible to achieve the best of both approaches, our new indicator invariants combined with the traditional equational unification.

6.10.3 Equality Constraints

Our constraint solving rules are targeted towards discovering whether a particular value is secret from the adversary. This is sufficient to prove many properties, but it does not suffice for all. For example, when considering an unknown key share attack, we might be interested in determining under what circumstances can two values be equal. This is tractable in the restricted Diffie-Hellman theory using unification, however it is not tractable in our current approach as our invariants consider questions of adversary knowledge. An extension targeting this property would be of interest, although it is unclear whether manual computational analysis is capable of exploring this property.

6.10.4 Attack Finding

We have not discussed support for attack finding in this chapter. It would be useful if in addition to performing proofs, we could also automatically recover attacks on protocols which are not secure. The constraint solving rules we described in the earlier sections are not sufficient for this purpose. For example, whilst a proof might fail because the adversary can recover all necessary indicator values, we do not correctly check whether they can be correctly combined to learn the exact term the adversary is interested in.

Recently Barthe et al. introduced a decision procedure for solving deducibility in a Diffie-Hellman exponentiation theory [23]. Deducibility is the problem of determining whether a given term can be constructed from an input set consisting of fresh values (representing exponents) and group elements with some polynomial in the exponent. This result is not useful for finding proofs in our framework because if the decision procedure returns 'false' it is only a verdict with respect to adversary's current knowledge set. It does not tell us anything about which terms the adversary would have to learn in order to deduce the target value.

However, deducability would be of considerable utility for attack finding. If we reach a state where the adversary has learned all necessary non-basis elements and indicator values, we can check to see if the adversary can currently deduce the target value. If so, we can instantiate the appropriate term and resolve that constraint. This approach could be extended with additional heuristics to improve the chance we successfully find an attack. We conjecture that even the limited extension described above would be sufficient to recover Kaliski's Unknown Key Share Attack on MQV.

6.10.5 Non-Prime Order Groups

In the preceding chapter we extended the traditional symbolic model to capture the additional behaviours of non-prime order groups. Our work in this chapter is orthogonal and not directly compatible with these extensions. It remains to be seen if our two extensions could be combined to achieve the best of both worlds. However, we are hopeful this combination would be straightforward.

6.11 Conclusion

In this chapter we have presented a set of constraint solving rules which extend the Tamarin Prover to support the field structure of Diffie-Hellman group exponents. We have sketched proofs that each rule is sound and complete and shown how they can be composed to together to prove simple security properties of the protocol MQV. Finally, we identified a number of avenues of future work which we plan to investigate in the future before submitting this work for publication.

Our work in the chapter forms the first proposed automated method for analysing a large class of Diffie-Hellman protocols that make direct use of the group element. Although we have not implemented our method and verified its tractability or generality, we believe our method is flexible and that many protocols and properties will become amenable to automated analysis as a result.

As we have discussed, our work in this chapter is ongoing and the subject of continuing refinement. In particular, we have highlighted further work that is required on the proofs of correctness and the implementation before it would be suitable for publication. Furthermore, we wish to explore additional extensions such as automated attack finding and non-prime order groups in the fullness of time.

7

Conclusions

In this thesis we set out to strengthen the relationship between automated protocol verification tools and real world cryptographic primitives. We designed novel symbolic models for digital signatures and Diffie-Hellman groups, implemented them in Tamarin and used them to analyse a number of well-known protocols. Finally, we developed a new approach for automatically analysing protocols which make use of the full field structure of Diffie-Hellman exponents and sketched how it could be applied to a key agreement protocol.

As a result of this work, we discovered new attacks on deployed protocols such as WS-Security, Secure Scuttlebutt, Tendermint, and DRKey. We also documented a number of previous analyses that had missed attacks on protocols such as Bluetooth's Secure Pairing, the Station-to-Station key exchange and Let's Encrypt's ACME. Our results also highlighted mismatches in cryptographic libraries claiming drop-in compatibility with LibSodium, with popular implementations such as Golang's NaCl module, Project Everest's HACL and Cloudflare's CIRCL treating small subgroup elements in X25519 and Ed25519 inconsistently. As a result of our outreach, both Golang and Cloudflare made changes to their API, protecting their users from an attacks leveraging this subtle behaviour.

Looking back at the cryptographic models we developed in this thesis, we diverged significantly from previous approaches. A traditional symbolic model is just an equational theory, which is a set of function symbols and identities which the functions satisfy. In our approach, we have replaced many of the identities by modelling functions as adversarially controlled oracles where the adversary is allowed to freely choose the result of the function evaluation, subject to additional constraints which reflect the security definition of the primitive. These constraints, which can be predicated over the entire trace, are much more expressive than universally quantified identities, which can only refer to information contained within the term.

Given the increased expressive power of our approach and the special purpose algorithms available for analysing case distinctions with equational theories, it is natural to expect there would be severe performance penalties as a consequence of our approach. Surprisingly, this turns out not to be the case, as the our evaluation showed. Instead,

we see only a minor performance impact even for some of the most complex properties. We speculate the additional cases that arise from our more precise models are mitigated by the Tamarin delaying case distinctions which it previously would have had to handle explicitly with every backwards search step.

Our new models are not entirely without disadvantages though. Whilst they offer much stronger guarantees than traditional models, they can be harder to interpret. For example, in our attack-finding signature models which use a traditional approach, inspecting a trace reveals the precise type of manipulation used by the attacker and it is straightforward to convert the attack into a precise algorithm which can be executed. Contrastingly, in the verification model, we simply see the attacker perform a manipulation which is not forbidden by the security definition but is not necessarily possible in practice.

We also saw that using a constraint-based model allowed us to model the full field structure of Diffie-Hellman exponents, in stark contrast to the impossibility results described for the approaches based on equational theories. This suggests a promising future for constraint-based models and motivates the development of a new generation of analysis tools designed from the ground-up to support modular definitions of constraint solving steps.

7.1 Contributions

A brief summary of our contributions follows:

- In Chapter 3:
  1. We developed a new family of symbolic models for digital signatures which captured attacks and behaviours unknowingly omitted from traditional models for the past two decades.
  2. We implemented our models in the Tamarin Prover, providing the first automated method for finding, or proving the absence of, attacks on security protocols that exploit subtle behaviours of provably secure signature schemes.
  3. We used our models to find known and new attacks on protocols that were previously proven secure in coarser symbolic models. Specifically, we showed new attacks on WS-Security Mutual Auth and the DRKey protocol. We also automatically find and verify the fix for the known key substitution attack on Let's Encrypt Draft 4. These protocols were all previously proven secure in the traditional symbolic signature model.
- In Chapter 5:
  1. We developed a new family of symbolic models for Diffie-Hellman groups that captures the internal structure of non-prime order groups in the real world. This model makes it possible to discover real world attacks, such as those based on small subgroup confinement, small subgroup key leakage and invalid elliptic curve

points. Additionally our models support the accurate representations of the various mitigations available to protocol designers such as excluding low order points, curve equation checks and single coordinate ladders. This allows for the effective symbolic verification of a protocol despite the use of a non-prime order group.

2. Using the Tamarin Prover, we develop the first automated method for finding or proving the absence of small subgroup and invalid curve point attacks on security protocols that use Diffie-Hellman groups.
3. We evaluate our models effectiveness on real world protocols and discover a new attack on the Secure Scuttlebutt Gossip protocol, violating a core authentication property, despite it having previously being verified as secure in a coarser symbolic model. We independently discover a recently disclosed attack on Tendermint's secure handshake protocol which allows for the impersonation of legitimate users.

- In Chapter 6:
  1. We develop a set of constraint solving rules, integrated with the Tamarin prover, to analyse protocols that make full use of the field structure of Diffie-Hellman group exponents. We sketch the correctness of our system.
  2. We illustrate the effectiveness of our rules on a hand worked example, showing how they can be used to succinctly analyse a real world protocol.

7.2 Future Work

At the end of Chapters 3, 5 and 6, we outlined specific areas of future work. Here we sketch some exciting research directions motivated by our results.

Tackling further primitives In this thesis we restricted our examination of symbolic models to digital signatures and Diffie-Hellman groups. However, the symbolic model contains many other primitives including hash functions, symmetric encryption and public key encryption. Exploring the match between these models and the corresponding behaviour in the real world is likely to be fruitful future work.

In particular, the symbolic model of symmetric encryption offers many opportunities for further refinement. It currently models a strong form of authenticated encryption and omits many properties found in practice. For example, the consequences of nonce-reuse attacks (and thus misuse-resistant schemes) as well as 'colliding' ciphertexts which can be decrypted (and authenticated) under more than one key. This latter property in particular is missing from symbolic models and could be of considerable use to an attacker attempting to 'confuse' a protocol which expects such ciphertexts to be a commitment to a particular plaintext.

Connections to the Computational Model Traditionally, researchers have searched for "computational soundness" results; theorems which specify under which conditions a symbolic proof implies a proof in the computational model. These theorems have typically had to rule out broad classes of protocols which are in practice widely used, or else require the use of strong primitives which are often impractical. Revisiting these results with our new symbolic models may prove fruitful, either we will found computational soundness results for broader classes of primitives and protocols or else we will discover new issues which we do not currently capture and hence can target for inclusion in our models. There are very few computational soundness results for Diffie-Hellman Groups and searching for connections with the algebraic group model may also be of interest.

Bridging the Gap Between Design and Implementation Our enhanced symbolic models for digital signatures and Diffie-Hellman groups capture each primitive in significantly more detail than has been possible before. However, this comes at a cost. Researchers or designers using our symbolic tools must put more detail into the models, for example specifying when particular checks occur on received Diffie-Hellman values. Previously these checks could not be represented because the relevant behaviour was not captured. This increased granularity motivates closing the gap between the symbolic model and the implementation. There are many active areas of research looking at this connection, either through code generation from the symbolic model or through some form of program verification to prove a hand written implementation corresponds to the model. As symbolic models increase in complexity, bridging the design-implemention gap becomes even more important.

Acronyms

AC Associative-Commutative
ACME Automatic Certificate Management Environment
CA Certificate Authority
CEO Conservative Exclusive Ownership
CF Cremers-Feltz
CR Commutative Ring
DEO Destructive Exclusive Ownership
DH Diffie-Hellman
DSKS Duplicate Signature Key Selection
ECDSA Elliptic Curve Digital Signature Algorithm
EUF-CMA Existential Unforgeability under an Adaptive Chosen Message Attack
FTFAG Fundamental Theorem of Finite Abelian Groups
IETF Internet Engineering Task Force
JPAKE Password Authenticated Key Exchange by Juggling
KDF Key Derivation Function
LE Let's Encrypt
MAC Message Authentication Code
MGU Most General Unifier
MQV Menezes–Qu–Vanstone
SEF-CMA Strong Existential Unforgeability under an Adaptive Chosen Message Attack
STS Station-to-Station
TLS Transport Layer Security
UEO Universal Exclusive Ownership
UKS Unknown Key Share
UM Unified Model
wPFS Weak Perfect Forward Secrecy

Bibliography

[1] *ACME Draft Barnes v1*. 2015. URL: https://datatracker.ietf.org/doc/draft-barnes-acme/01/ (visited on 02/2019).

[2] *ACME Draft Barnes v3*. 2015. URL: https://datatracker.ietf.org/doc/draft-barnes-acme/03/ (visited on 02/2019).

[3] *ACME Draft Barnes v4*. 2015. URL: https://datatracker.ietf.org/doc/draft-barnes-acme/04/ (visited on 02/2019).

[4] *ACME signature misuse vulnerability in draft-barnes-acme-04*. 2015. URL: https://www.ietf.org/mail-archive/web/acme/current/msg00484.html (visited on 11/2018).

[5] Roberto M Amadio and Denis Lugiez. 'On the reachability problem in cryptographic protocols'. In: *Concur*. Vol. 1877. Springer. 2000, pp. 380–394.

[6] Jee Hea An, Yevgeniy Dodis and Tal Rabin. 'On the Security of Joint Signature and Encryption'. In: *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*. 2002, pp. 83–107.

[7] R Ankney, D Johnson and M Matyas. 'The unified model. contribution to ansi x9f1'. In: *Standards Projects (Financial Crypto Tools), ANSI X* 42 (1995).

[8] Adrian Antipa et al. 'Validation of Elliptic Curve Public Keys'. In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*. 2003, pp. 211–223.

[9] *ANTLR SPHTY Repository*. 2019. URL: https://github.com/galadran/antlr-spthy (visited on 12/2019).

[10] *Apache CXF: An Open-Source Services Framework*. 2019. URL: https://cxf.apache.org/ (visited on 02/02/2019).

[11] *Apache CXF: WS-Security*. 2018. URL: https://cxf.apache.org/docs/ws-security.html (visited on 02/02/2019).

[12] Kenichi Arai and Toshinobu Kaneko. 'Formal Verification of Improved Numeric Comparison Protocol for Secure Simple Paring in Bluetooth Using ProVerif'. In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer ... 2014, p. 1.

[13] Alessandro Armando et al. 'The AVISPA tool for the automated validation of internet security protocols and applications'. In: *International conference on computer aided verification*. Springer. 2005, pp. 281–285.

[14] *Automatic Certificate Management Environment (ACME)*. 2015. URL: https://tools.ietf.org/html/draft-ietf-acme-acme-00 (visited on 02/2019).

[15] *Automatic Certificate Management Environment (ACME)*. 2016. URL: https://tools.ietf.org/html/draft-ietf-acme-acme-02 (visited on 02/2019).

[16] *Automatic Certificate Management Environment (ACME)*. 2018. URL: https://tools.ietf.org/html/draft-ietf-acme-acme-16 (visited on 02/2019).

[17] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge university press, 1999.

[18] Michael Backes et al. 'Symbolic and Cryptographic Analysis of the Secure WS-ReliableMessaging Scenario'. In: *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006, Proceedings*. 2006, pp. 428–445.

[19] Joonsang Baek and Kwangjo Kim. 'Remarks on the unknown key share attacks'. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 83.12 (2000), pp. 2766–2769.

[20] Manuel Barbosa et al. 'SoK: Computer-Aided Cryptography'. In: ().

[21] Elaine Barker, Don Johnson and Miles Smid. 'NIST special publication 800-56A: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised)'. In: *Comput. Secur. Natl. Inst. Stand. Technol.(NIST), Publ. by NIST* (2007).

[22] Gilles Barthe et al. 'Easycrypt: A tutorial'. In: *Foundations of security analysis and design vii*. Springer, 2013, pp. 146–166.

[23] Gilles Barthe et al. 'Symbolic Proofs for Lattice-Based Cryptography'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 2018, pp. 538–555.

[24] David Basin, Jannik Dreier and Ralf Sasse. 'Automated symbolic proofs of observational equivalence'. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 1144–1155.

[25] David Basin et al. 'A formal analysis of 5G authentication'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2018, pp. 1383–1396.

[26] Daniel Bernstein. *Curve25519: How do I validate Curve25519 public keys?* 2006. URL: https://cr.yp.to/ecdh.html (visited on 02/02/2019).

[27] Daniel Bernstein. *Twist Security*. 2013. URL: https://safecurves.cr.yp.to/twist.html (visited on 02/02/2019).

[28] Daniel J. Bernstein. 'Curve25519: New Diffie-Hellman Speed Records'. In: *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*. 2006, pp. 207–228.

[29] Daniel J. Bernstein et al. 'High-speed high-security signatures'. In: *J. Cryptographic Engineering* 2.2 (2012), pp. 77–89.

[30] Karthikeyan Bhargavan, Antoine Delignat-Lavaud and Nadim Kobeissi. 'Formal Modeling and Verification for Domain Validation and ACME'. In: *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*. Vol. 10322. Lecture Notes in Computer Science. Springer, 2017, pp. 561–578.

[31] Karthikeyan Bhargavan, Antoine Delignat-Lavaud and Alfredo Pironti. 'Verified Contributive Channel Bindings for Compound Authentication'. In: *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*. 2015.

[32] Karthikeyan Bhargavan, Cédric Fournet and Andrew D. Gordon. 'Verified Reference Implementations of WS-Security Protocols'. In: *Web Services and Formal Methods, Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006, Proceedings*. 2006, pp. 88–106.

[33] Karthikeyan Bhargavan et al. 'Secure sessions for web services'. In: *Proceedings of the 1st ACM Workshop On Secure Web Services, SWS 2004, Fairfax, VA, USA, October 29, 2004*. 2004, pp. 56–66.

[34] Karthikeyan Bhargavan et al. 'TulaFale: A Security Tool for Web Services'. In: *Formal Methods for Components and Objects, Second International Symposium, FMCO 2003, Leiden, The Netherlands, November 4-7, 2003, Revised Lectures*. 2003, pp. 197–222.

[35] Karthikeyan Bhargavan et al. 'Verified interoperable implementations of security protocols'. In: *ACM Trans. Program. Lang. Syst.* 31.1 (2008), 5:1–5:61.

[36] Ingrid Biehl, Bernd Meyer and Volker Müller. 'Differential Fault Attacks on Elliptic Curve Cryptosystems'. In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. 2000, pp. 131–146.

[37] Eli Biham and Lior Neumann. 'Breaking the Bluetooth Pairing–The Fixed Coordinate Invalid Curve Attack'. In: *International Conference on Selected Areas in Cryptography*. Springer. 2019, pp. 250–273.

[38] 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*. Vol. 1560. Lecture Notes in Computer Science. Springer, 1999, pp. 154–170.

[39] Bruno Blanchet. 'An Efficient Cryptographic Protocol Verifier Based on Prolog Rules'. In: *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*. IEEE Computer Society, 2001, pp. 82–96.

[40] Bruno Blanchet. 'CryptoVerif: Computationally sound mechanized prover for cryptographic protocols'. In: *Dagstuhl seminar "Formal Protocol Verification Applied*. Vol. 117. 2007, p. 156.

[41] Bruno Blanchet. 'Security protocol verification: Symbolic and computational models'. In: *Proceedings of the First international conference on Principles of Security and Trust*. Springer-Verlag. 2012, pp. 3–29.

[42] Bruno Blanchet et al. 'ProVerif 2.00: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial'. In: (2018). Version from 2018-05-16.

[43] Oleg Vladimirovič Bogopolskij. *Introduction to group theory*. Vol. 6. European Mathematical Society, 2008.

[44] Jens-Matthias Bohli, Stefan Röhrich and Rainer Steinwandt. 'Key substitution attacks revisited: Taking into account malicious signers'. In: *Int. J. Inf. Sec.* 5.1 (2006), pp. 30–36.

[45] Dan Boneh, Emily Shen and Brent Waters. 'Strongly Unforgeable Signatures Based on Computational Diffie-Hellman'. In: *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*. Vol. 3958. Lecture Notes in Computer Science. Springer, 2006, pp. 229–240.

[46] Michele Boreale. 'Symbolic trace analysis of cryptographic protocols'. In: *ICALP*. Vol. 2076. Springer. 2001, pp. 667–681.

[47] Michele Boreale and Maria Grazia Buscemi. 'A framework for the analysis of security protocols'. In: *CONCUR*. Vol. 2421. Springer. 2002, pp. 483–498.

[48] Michele Boreale and Maria Grazia Buscemi. 'Symbolic analysis of crypto-protocols based on modular exponentiation'. In: *MFCS*. Vol. 2747. Springer. 2003, pp. 269–278.

[49] Colin Boyd et al. 'ASICS: authenticated key exchange security incorporating certification systems'. In: *Int. J. Inf. Sec.* 16.2 (2017), pp. 151–171.

[50] Eric Brier and Marc Joye. 'Weierstraß Elliptic Curves and Side-Channel Attacks'. In: *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*. 2002, pp. 335–345.

[51] Stanley Burris and John Lawrence. 'Unification in commutative rings is not finitary'. In: *Information Processing Letters* 36.1 (1990), pp. 37–38.

[52] Richard Chang and Vitaly Shmatikov. 'Formal analysis of authentication in bluetooth device pairing'. In: *Fcs-arspa07* (2007), p. 45.

[53] Sanjit Chatterjee, Alfred Menezes and Berkant Ustaoglu. 'Combined Security Analysis of the One- and Three-Pass Unified Model Key Agreement Protocols'. In: *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*. 2010, pp. 49–68.

[54] Yannick Chevalier and Mounira Kourjieh. 'Key Substitution in the Symbolic Analysis of Cryptographic Protocols'. In: *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*. Vol. 4855. Lecture Notes in Computer Science. Springer, 2007, pp. 121–132.

[55] Yannick Chevalier et al. 'Deciding the security of protocols with Diffie-Hellman exponentiation and products in exponents'. In: *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer. 2003, pp. 124–135.

[56] Kim-Kwang Raymond Choo, Colin Boyd and Yvonne Hitchcock. 'Errors in computational complexity proofs for protocols'. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2005, pp. 624–643.

[57] Tom Chothia, Ben Smyth and Christopher Staite. 'Automatically Checking Commitment Protocols in ProVerif without False Attacks'. In: *Principles of Security and Trust - 4th International Conference, POST 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings*. 2015, pp. 137–155.

[58] *CIRCL (Cloudflare Interoperable, Reusable Cryptographic Library)*. 2019. URL: https://github.com/cloudflare/circl (visited on 02/02/2019).

[59] Edmund M Clarke, Somesh Jha and Will Marrero. 'Using state space exploration and a natural deduction style message derivation engine to verify security protocols'. In: *Programming Concepts and Methods PROCOMET98*. Springer, 1998, pp. 87–106.

[60] *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2005. ISBN: 978-1-58488-518-4.

[61] Hubert Comon-Lundh and Stéphanie Delaune. 'The finite variant property: How to get rid of some algebraic properties'. In: *RTA*. Vol. 3467. Springer. 2005, pp. 294–307.

[62] Craig Costello, Patrick Longa and Michael Naehrig. 'A brief discussion on selecting new elliptic curves'. In: *Microsoft Research. Microsoft* 8 (2015).

[63] Cas Cremers and Michele Feltz. 'Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal'. In: *European Symposium on Research in Computer Security*. Springer. 2012, pp. 734–751.

[64] Cas Cremers and Dennis Jackson. 'Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman'. In: *32nd IEEE Computer Security Foundations Symposium, CSF 2019, Hoboken, NJ, USA, June 25-28, 2019*. 2019, pp. 78–93.

[65] Cas Cremers and Dennis Jackson. *Supplementary Materials and Tamarin models*. 2019. URL: https://people.cispa.io/cas.cremers/tools/tamarin/primeorder/index.html (visited on 02/02/2019).

[66] Cas Cremers and Sjouke Mauw. *Operational semantics and verification of security protocols*. Springer Science & Business Media, 2012.

[67] Cas Cremers et al. 'Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication'. In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 470–485.

[68] *Critical vulnerability in JSON Web Encryption (JWE)*. 2017. URL: http://blog.intothesymmetry.com/2017/03/critical-vulnerability-in-json-web.html (visited on 02/02/2019).

[69] Martin Dehnel-Wild. 'Component-based security under partial compromise'. PhD thesis. University of Oxford, 2018.

[70] W. Diffie and M. Hellman. 'New Directions in Cryptography'. In: *IEEE Trans. Inf. Theor.* 22.6 (Sept. 2006), pp. 644–654.

[71] Whitfield Diffie and Martin Hellman. 'New directions in cryptography'. In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[72] Whitfield Diffie, Paul C. van Oorschot and Michael J. Wiener. 'Authentication and Authenticated Key Exchanges'. In: *Des. Codes Cryptography* 2.2 (1992), pp. 107–125.

[73] Shaddin F Doghmi, Joshua D Guttman and F Javier Thayer. 'Searching for shapes in cryptographic protocols'. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2007, pp. 523–537.

[74] Daniel J Dougherty and Joshua D Guttman. 'An algebra for symbolic Diffie-Hellman protocol analysis'. In: *International Symposium on Trustworthy Global Computing*. Springer. 2012, pp. 164–181.

[75] Daniel J Dougherty and Joshua D Guttman. 'Decidability for lightweight Diffie-Hellman protocols'. In: *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*. IEEE. 2014, pp. 217–231.

[76] Thai Duon. *Why not validate Curve25519 public keys could be harmful*. 2015. URL: https://vnhacker.blogspot.com/2015/09/why-not-validating-curve25519-public.html (visited on 02/02/2019).

[77] *Elliptic Curves for Security*. 2016. URL: https://tools.ietf.org/html/rfc7748 (visited on 02/02/2019).

[78] Santiago Escobar, Catherine Meadows and José Meseguer. 'Maude-NPA: Cryptographic protocol analysis modulo equational properties'. In: *Foundations of Security Analysis and Design V*. Springer, 2009, pp. 1–50.

[79] Santiago Escobar, Catherine A. Meadows and Jose Meseguer. 'Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties'. In: *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*. 2007, pp. 1–50.

[80] Santiago Escobar et al. 'Diffie-Hellman cryptographic reasoning in the Maude-NRL protocol analyzer'. In: *Proc. of SecRet* 2007 (2007).

[81] Pierre-Alain Fouque et al. 'Fault attack on elliptic curve Montgomery ladder implementation'. In: *2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*. Ieee. 2008, pp. 92–98.

[82] Mark Franzen. 'Hilbert's tenth problem is of unification type zero'. In: *Journal of Automated Reasoning* 9.2 (1992), pp. 169–178.

[83] Dov M Gabbay, CJ Hogger and JA Robinson. 'Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 2'. In: ().

[84] *Getting Started with gSOAP*. 2019. URL: https://www.genivia.com/dev.html (visited on 02/02/2019).

[85] *Go Documentation for the Standard Library: x/crypto/nacl*. 2019. URL: https://godoc.org/golang.org/x/crypto/nacl (visited on 02/02/2019).

[86] Shafi Goldwasser, Silvio Micali and Ronald L. Rivest. 'A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks'. In: *SIAM J. Comput.* 17.2 (1988), pp. 281–308.

[87] Jean Goubault-Larrecq, Muriel Roger and Kumar Neeraj Verma. 'Abstraction and resolution modulo AC: How to verify Diffie–Hellman-like protocols automatically'. In: *The Journal of Logic and Algebraic Programming* 64.2 (2005), pp. 219–251.

[88] Bluetooth Special Interest Group. *Bluetooth Core Specification v5.1*. 2019. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (visited on 02/02/2019).

[89] Felix Günther and Bertram Poettering. 'Linkable Message Tagging: Solving the Key Distribution Problem of Signature Schemes'. In: *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*. Vol. 9144. Lecture Notes in Computer Science. Springer, 2015, pp. 195–212.

[90] ISO Central Secretary. *Information technology security techniques - key management - Part 3: Mechanisms using asymmetric techniques*. en. Standard Iso/iec 11770-3. Version Final Text for Publication. Geneva, CH: International Organization for Standardization, 1999.

[91] Dennis Jackson et al. 'Seems Legit: Automated Analysis of Subtle Attacks on Protocols that Use Signatures'. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. 2019, pp. 2165–2180.

[92] Dennis Jackson et al. *Supplementary Materials and Models*. 2019. URL: https://people.cispa.io/cas.cremers/downloads/archives/Tamarin%5Fbetter%5Fsignatures.zip.

[93] Tibor Jager, Saqib A. Kakvi and Alexander May. 'On the Security of the PKCS#1 v1.5 Signature Scheme'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Acm, 2018, pp. 1195–1208.

[94] Tibor Jager, Jörg Schwenk and Juraj Somorovsky. 'Practical Invalid Curve Attacks on TLS-ECDH'. In: *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. 2015, pp. 407–425.

[95] jedisct1. *It's possible to forge messages that cryptosignopen verifies if the public key is zero.* Issue 112 jedisct1/libsodium. 2016. URL: https://github.com/jedisct1/libsodium/issues/112 (visited on 11/2018).

[96] Simon Josefsson and Ilari Liusvaara. 'Edwards-Curve Digital Signature Algorithm (EdDSA)'. In: *Rfc* 8032 (2017), pp. 1–60.

[97] Burton S Kaliski Jr. 'An unknown key-share attack on the MQV key agreement protocol'. In: *ACM Transactions on Information and System Security (TISSEC)* 4.3 (2001), pp. 275–288.

[98] Deepak Kapur, Paliath Narendran and Lida Wang. 'An E-unification algorithm for analyzing protocols that use modular exponentiation'. In: *RTA*. Vol. 2706. Springer. 2003, pp. 165–179.

[99] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. ISBN: 9781466570269.

[100] Tiffany Hyun-Jin Kim et al. 'Lightweight source authentication and path validation'. In: *ACM SIGCOMM Computer Communication Review*. Vol. 44. 4. Acm. 2014, pp. 271–282.

[101] Nadim Kobeissi and Karthikeyan Bhargavan. 'Noise Explorer: Fully Automated Modeling and Verification for Arbitrary Noise Protocols'. In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019.

[102] Nadim Kobeissi, Karthikeyan Bhargavan and Bruno Blanchet. 'Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach'. In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2017, pp. 435–450.

[103] Neal Koblitz and Alfred Menezes. 'Critical perspectives on provable security: Fifteen years of" another look" papers'. In: *Advances in Mathematics of Communications* 13.4 (2019), pp. 517–558.

[104] Hugo Krawczyk. 'HMQV: A High-Performance Secure Diffie-Hellman Protocol (Extended Abstract)'. In: *CRYPTO 2005*. Springer. 2005, p. 546.

[105] Sébastien Kunz-Jacques and David Pointcheval. 'About the Security of MTI/C0 and MQV'. In: *Security and Cryptography for Networks* (2006), pp. 156–172.

[106] Sébastien Kunz-Jacques and David Pointcheval. 'About the Security of MTI/C0 and MQV'. In: *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*. 2006, pp. 156–172.

[107] Ralf Küsters and Tomasz Truderung. 'Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation'. In: *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*. 2009, pp. 157–171.

[108] Brian LaMacchia, Kristin Lauter and Anton Mityagin. 'Stronger security of authenticated key exchange'. In: *International conference on provable security*. Springer. 2007, pp. 1–16.

[109] Kristin E. Lauter and Anton Mityagin. 'Security Analysis of KEA Authenticated Key Exchange Protocol'. In: *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*. 2006, pp. 378–394.

[110] *LibSodium: a modern, easy-to-use cryptographic library*. 2019. URL: https://download.libsodium.org/doc/ (visited on 02/02/2019).

[111] *LibSodium: reject low order points before the multiplication*. 2017. URL: https://github.com/jedisct1/libsodium/commit/c190574cee382ace1ee0f8fdacc136f1797287e6 (visited on 02/02/2019).

[112] Chae Hoon Lim and Pil Joong Lee. 'A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroupp'. In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. 1997, pp. 249–263.

[113] Christina Lindenberg, Kai Wirt and Johannes A. Buchmann. 'Formal Proof for the Correctness of RSA-PSS'. In: *IACR Cryptology ePrint Archive* 2006 (2006), p. 11.

[114] Moses Liskov and F Javier Thayer. 'Modeling Diffie-Hellman Derivability for Automated Analysis'. In: *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*. IEEE. 2014, pp. 232–243.

[115] Moses D. Liskov et al. 'Enrich-by-Need Protocol Analysis for Diffie-Hellman'. In: *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*. 2019, pp. 135–155.

[116] Gavin Lowe. 'Breaking and fixing the Needham-Schroeder public-key protocol using FDR'. In: *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 1996, pp. 147–166.

[117] Christopher Lynch and Catherine Meadows. 'Sound approximations to Diffie-Hellman using rewrite rules'. In: *ICICS*. Vol. 3269. Springer. 2004, pp. 262–277.

[118] David Marker. *Model theory: an introduction*. Vol. 217. Springer Science & Business Media, 2006.

[119] Yuri Matiyasevich. 'Computation paradigms in light of Hilbert's tenth problem'. In: *New computational paradigms*. Springer, 2008, pp. 59–85.

[120] Catherine Meadows. 'The NRL protocol analyzer: An overview'. In: *The Journal of Logic Programming* 26.2 (1996), pp. 113–131.

[121] Simon Meier. 'Advancing automated security protocol verification'. Doctoral thesis, ETH Zurich, Switzerland. PhD thesis. ETH Zürich, 2013. (Visited on 04/2018).

[122] Simon Meier et al. 'The TAMARIN prover for the symbolic analysis of security protocols'. In: *International Conference on Computer Aided Verification*. Springer. 2013, pp. 696–701.

[123] Alfred Menezes. 'Another look at HMQV'. In: *J. Mathematical Cryptology* 1.1 (2007), pp. 47–64.

[124] Alfred Menezes. 'Some new key agreement protocols providing implicit authentication'. In: *Selected Areas in Cryptography (SAC'95)* (1995).

[125] Alfred Menezes and Nigel P. Smart. 'Security of Signature Schemes in a Multi-User Setting'. In: *Des. Codes Cryptography* 33.3 (2004), pp. 261–274.

[126] Alfred Menezes and Berkant Ustaoglu. 'On reusing ephemeral keys in Diffie-Hellman key agreement protocols'. In: *Ijact* 2.2 (2010), pp. 154–158.

[127] Alfred J. Menezes, Scott A. Vanstone and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 1996. ISBN: 849385237.

[128] *Message Security in WCF*. 2019. URL: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/message-security-in-wcf (visited on 02/02/2019).

[129] Jonathan Millen and Vitaly Shmatikov. 'Symbolic protocol analysis with products and Diffie-Hellman exponentiation'. In: *Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*. IEEE. 2003, pp. 47–61.

[130] Kevin Milner. 'Detecting the misuse of secrets: foundations, protocols, and verification'. PhD thesis. University of Oxford, 2018.

[131] Peter L Montgomery. 'Speeding the Pollard and elliptic curve methods of factorization'. In: *Mathematics of computation* 48.177 (1987), pp. 243–264.

[132] Samuel Neves and Mehdi Tibouchi. 'Degenerate Curve Attacks - Extending Invalid Curve Attacks to Edwards Curves and Other Models'. In: *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part II*. 2016, pp. 19–35.

[133] *OASIS Web Services Security (WSS) TC*. 2006. (Visited on 02/02/2019).

[134] *Oracle Fusion Middleware: WS-Policy Reference*. 2014. URL: https://docs.oracle.com/cd/E55956%5F01/doc.11123/user%5Fguide/content/ws%5Fpolicies.html (visited on 02/02/2019).

[135] Adrian Perrig et al. *SCION: A Secure Internet Architecture*. Springer International Publishing AG, 2017. ISBN: 978-3-319-67079-9.

[136] Trevor Perrin. *X25519 and zero outputs*. 2017. URL: https://curves.moderncrypto.narkive.com/rSFu5TRe/x25519-and-zero-outputs (visited on 02/02/2019).

[137] *PKCS 1: RSA Cryptography Specifications Version 2.2, Section 9.2, Note 2*. Tech. rep. RFC 8017, 2016.

[138] Thomas Pornin and Julien P. Stern. 'Digital Signatures Do Not Guarantee Exclusive Ownership'. In: *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*. Vol. 3531. Lecture Notes in Computer Science. 2005, pp. 138–150.

[139] Harry Roberts. *Tendermint Secrect Connection Disclosure*. 2018. URL: https://github.com/tendermint/tendermint/issues/3010 (visited on 02/02/2019).

[140] Michaël Rusinowitch and Mathieu Turuani. 'Protocol insecurity with finite number of sessions is NP-complete'. PhD thesis. INRIA, 2001.

[141] Ralf Sasse et al. 'Protocol Analysis Modulo Combination of Theories: A Case Study in Maude-NPA.' In: *STM* 6710 (2010), pp. 163–178.

[142] Benedikt Schmidt. *Formal analysis of key exchange protocols and physical protocols*. Doctoral thesis, ETH Zurich, Switzerland. 2012.

[143] Benedikt Schmidt et al. '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*. IEEE Computer Society, 2012, pp. 78–94.

[144] Benedikt Schmidt et al. 'Automated analysis of Diffie-Hellman protocols and advanced security properties'. In: *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*. IEEE. 2012, pp. 78–94.

[145] Manfred Schmidt-Schauß. 'An algorithm for distributive unification'. In: *International Conference on Rewriting Techniques and Applications*. Springer. 1996, pp. 287–301.

[146] *Scuttlebutt Protocol Guide*. 2017. URL: https://ssbc.github.io/scuttlebutt-protocol-guide/ (visited on 02/02/2019).

[147] *Scuttlebutt Secret Handshake Go Implementation*. 2019. URL: https://github.com/cryptoscope/secretstream (visited on 02/02/2019).

[148] *Scuttlebutt Secret Handshake Implementations*. 2019. URL: https://github.com/auditdrivencrypto/secret-handshake (visited on 02/02/2019).

[149] *Scuttlebutt Secret Handshake in Tamarin*. 2018. URL: https://github.com/keks/tamarin-shs (visited on 02/02/2019).

[150] *Secure Scuttlebutt*. 2019. URL: https://scuttlebot.io/ (visited on 02/02/2019).

[151] *Securing your Web services with Spring-WS*. 2018. URL: https://docs.spring.io/spring-ws/docs/3.0.4.RELEASE/reference/\#security (visited on 02/02/2019).

[152] Kudelski Security. *Should Curve25519 keys be validated?* 2017. URL: https://research.kudelskisecurity.com/2017/04/25/should-ecdh-keys-be-validated/ (visited on 02/02/2019).

[153] Dawn Xiaodong Song. 'Athena: a new efficient automatic checker for security protocol analysis'. In: *Computer Security Foundations Workshop, 1999. Proceedings of the 12th IEEE*. IEEE. 1999, pp. 192–202.

[154] Jacques Stern et al. 'Flaws in Applying Proof Methodologies to Signature Schemes'. In: *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 93–110.

[155] Dominic Tarr. *Designing a Secret Handshake: Authenticated Key Exchange as a Capability System*. 2015. URL: https://dominictarr.github.io/secret-handshake-paper/shs.pdf.

[156] Tamarin Team. *Tamarin-Prover Manual – Security Protocol Analysis in the Symbolic Model*. Tech. rep. Version of 2018-11-27. https://tamarin-prover.github.io/manual/index.html: tamarin-prover.github.io, 2016. (Visited on 02/02/2019).

[157] *Tendermint Secrect Connection Protocol*. 2019. URL: https://tendermint.com/docs/tendermint-core/secure-p2p.html (visited on 02/02/2019).

[158] *Tendermint Website*. 2019. URL: https://tendermint.com/ (visited on 02/02/2019).

[159] *The Ristretto Group*. 2019. URL: https://ristretto.group/ristretto.html (visited on 02/02/2019).

[160] Ferucio Laurentiu Tiplea, Constantin Enea and Catalin V Bîrjoveanu. 'Decidability and complexity results for security protocols'. In: *VISSAS* 1 (2005), pp. 185–211.

[161] Mathieu Turuani. 'The CL-Atse protocol analyser'. In: *International Conference on Rewriting Techniques and Applications*. Springer. 2006, pp. 277–286.

[162] Luke Valenta et al. 'Measuring small subgroup attacks against Diffie-Hellman'. In: *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. 2017.

[163] Serge Vaudenay. 'The Security of DSA and ECDSA'. In: *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*. 2003, pp. 309–323.

[164] *WCF: Message Security with Mutual Certificates*. 2017. URL: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/message-security-with-mutual-certificates (visited on 02/02/2019).

[165] *WebSphere Application Server Liberty base*. 2019. URL: https://www.ibm.com/support/knowledgecenter/fi/SSEQTP%5Fliberty/com.ibm.websphere.wlp.doc/ae/cwlp%5Fwssec%5Ftemplates.html (visited on 02/02/2019).

[166] *WS-Attacks*. 2019. URL: http://www.ws-attacks.org/Main%5FPage (visited on 02/02/2019).

[167] *(WSS1.1) Mutual Authentication with X.509 Certificates, Sign, Encrypt*. 2019. URL: http://docs.oasis-open.org/ws-sx/security-policy/examples/ws-sp-usecases-examples.html\#%5FToc274723250 (visited on 02/02/2019).

[168] Fuyuan Zhang et al. 'Mechanized network origin and path authenticity proofs'. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. Acm. 2014, pp. 346–357.

[169] Jean-Karim Zinzindohoué et al. 'HACL∗: A verified modern cryptographic library'. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Acm. 2017, pp. 1789–1806.