Symbolically Analyzing Security Protocols using TAMARIN

David Basin*, Cas Cremers†, Jannik Dreier‡, and Ralf Sasse*

* Department of Computer Science, ETH Zurich, Switzerland
† University of Oxford, Oxford, UK
‡ LORIA, Université de Lorraine, INRIA, CNRS, Nancy, France

Abstract. The TAMARIN prover is a powerful tool for the symbolic modeling and analysis of security protocols. It has expressive languages for specifying protocols, adversary models, and properties, including trace and observational equivalence properties. TAMARIN features both automatic and interactive theorem proving modes, it has built-in support for equational theories such as the one modeling Diffie-Hellman key exchanges, and it can handle protocols with mutable state. TAMARIN also supports a form of induction, and efficiently parallelizes its proof search. This combination of features has proven to be highly successful in practice.

1 Introduction

During the last three decades, there has been considerable research devoted to the symbolic analysis of security protocols and existing tools have had considerable success both in detecting attacks on protocols and showing their absence. Nevertheless, there is still a large discrepancy between the symbolic models that one specifies on paper and the models that can be effectively analyzed by tools.

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

In more detail, and as will be explained in subsequent sections, TAMARIN's execution model is that of a labeled transition system. The state space is made up of multi-sets of *facts*, representing the adversary's knowledge, messages on the network, and the protocol participants' state. The protocol and adversary capabilities are then specified by multi-set rewriting rules. A sequence of transitions gives rise to a trace, which is the sequence of the labels of the applied rules. Properties are specified in a guarded fragment of first-order logic that allows quantification over messages and timepoints, and formulas are interpreted over

traces. Proofs are constructed using backward search with support for reasoning modulo equational theories. As practical examples, these features enable the tool to handle: protocols with non-monotonic mutable global state and complex control flow such as loops; complex security properties such as the eCK model [33] for key exchange protocols; and equational theories such as Diffie-Hellman, bilinear pairings, and convergent user-specified theories with the finite variant property [17].

TAMARIN provides two ways to construct proofs. It has an efficient, fully automated mode that combines deduction and equational reasoning with heuristics to guide the proof search. If the tool's automated proof search terminates, it returns either a proof of correctness (for an unbounded number of role instances and fresh values) or a counterexample, representing an attack that violates the stated property. However, since the correctness of security protocols is an undecidable problem, the tool may not terminate on a given verification problem. Hence, users may need to resort to TAMARIN's interactive mode to explore the proof states, inspect attack graphs, and seamlessly combine manually guided proofs with automated proof search.

TAMARIN is based on a number of key ideas. Algorithmically, it builds upon and generalizes the backwards search used by the Scyther tool [19] to enable protocol verification. Support for the theory for Diffie-Hellman exponentiation was developed in [41]. In the theses of Meier [34] and Schmidt [40], the approach was extended with trace induction and with support for bilinear pairings and operators modulo associativity-commutativity (AC). Recent work [7] has extended TAMARIN to handle equivalence properties. Tamarin now supports user-defined convergent equational theories with the finite variant property [24], while previously only the smaller set of subterm-convergent user defined theories was supported.

TAMARIN **Resources.** The main webpage of the TAMARIN Prover is hosted at [43] and provides links for downloading the tool, an extensive user manual, and further reading. TAMARIN's development is a collaborative effort, and we encourage contributions to the tool, the manual, and the case studies. See the webpage for details on how to contribute.

Outline. The remainder of this paper is structured as follows. In Section 2 we provide an overview of the TAMARIN system. In Section 3, we summarize some of the more prominent applications of TAMARIN. We compare to related work in Section 4 and conclude in Section 5 with a brief discussion of future perspectives.

2 System Overview

We start with an example that illustrates TAMARIN's use. Afterwards, we describe its underlying foundations and implementation.

2.1 Example: Diffie-Hellman Key Exchange

Input. TAMARIN takes as its command-line input the name of a theory file that defines the equational theory modeling the protocol messages, the multi-set rewriting system modeling the protocol, and a set of statements specifying the protocol's desired properties. To analyze the security of a variant of the Diffie-Hellman protocol, we use a theory file that consists of the following parts.

Input: Equational Theory. To specify the set of protocol messages, we write:

```
builtins: diffie-hellman
functions: mac/2, shk/0 [private]
```

This enables support for Diffie-Hellman (DH) exponentiation and defines two additional function symbols, while the DH built-in includes constant `g` already. The support for DH exponentiation defines the operator `^` for exponentiation, which satisfies the equation `(g^x)^y=(g^y)^x`, and additional operators and equations. We use the binary function symbol `mac` to model a message authentication code (MAC), the constant `g` to model the generator of a DH group, and the constant `shk` to model a shared secret key, which is declared as private and therefore not directly deducible by the adversary. Support for pairing and projection using `<_,_>`, `fst`, and `snd` is provided by default.

Input: Protocol. Our protocol definition consists of three (labeled) multi-set rewriting rules. Each rule is a triple: sequences of facts as left-hand-sides, labels, and right-hand-sides. Facts are of the form $F(t_1, \ldots, t_k)$ for a fact symbol F and terms t_i. The protocol rules use the fixed unary fact symbols `Fr` and `In` in their left-hand-side to obtain fresh names (unique and unguessable constants) and messages received from the network. To send a message to the network, they use the fixed unary fact symbol `Out` in their right-hand-side. Note that both participants in this exchange can send their initial message to their partner independently, unlike in the often used initiator-and-responder setup.

Our first rule models the creation of a new protocol thread `tid` that chooses a fresh exponent `x` and sends out `g^x` concatenated with a MAC of this value and the participants' identities:

```
rule Step1: [ Fr(tid:fresh), Fr(x:fresh) ] –[  ]→
            [ Out(<g^(x:fresh), mac(shk, <g^(x:fresh), A:pub, B:pub>)>)
            , Step1(tid:fresh, A:pub, B:pub, x:fresh) ]
```

In this rule, we use the sort annotations `fresh` and `pub` to ensure that the corresponding variables can only be instantiated with fresh and public names. An instance of the `Step1` rule rewrites the state by *consuming* two `Fr`-facts to obtain the fresh names `tid` and `x` and *generating* an `Out`-fact with the sent message and a `Step1`-fact denoting that the given thread has completed the first step with the given parameters. The arguments of the `Step1`-fact denote the thread identifier, the actor, the intended partner, and the chosen exponent. As the rule has no label it has no direct effect on the trace. However, it does change the state, thereby enabling further rules that consume the state facts in its conclusion.

Our second rule models the second step of a protocol thread:

```
rule Step2: [ Step1(tid, A, B, x:fresh), In(<Y, mac(shk, <Y, B, A>)>) ]
              −[ Accept(tid, Y^(x:fresh)) ]→ []
```

Here, a `Step1`-fact, which must have been created in an earlier `Step1`-step, is consumed in addition to an `In`-fact. The `In`-fact uses pattern matching to verify the MAC. The corresponding label `Accept(tid, Y^(x:fresh))` denotes that the thread `tid` has accepted the session key `Y^(x:fresh)`.

Our third rule models revealing the shared secret key to the adversary:

```
rule RevealKey: [] −[ Reveal() ]→ [ Out(shk) ]
```

The constant `shk` is output on the network and the label `Reveal()` ensures that the trace reflects *whether* and *when* a reveal has happened.

The set of protocol traces is defined via multi-set rewriting (modulo the equational theory) with these rules and the built-in rules for fresh name creation, message reception by the adversary, message deduction, and message sending by the adversary, which is observable via facts of the form `K(m)`. More precisely, the trace corresponding to a multi-set rewriting derivation is the sequence of the labels of the applied rules.

Input: Properties. We define the desired security properties of the protocol as trace or equivalence properties. In the case of trace properties, the labels of the protocol rules must contain sufficient information to state these properties. In Tamarin, properties are specified as so-called lemmas, which are then discharged or disproven by the tool.

```
lemma Accept_Secret:
  ∀ i j tid key. Accept(tid,key)@i & K(key)@j ⇒ ∃ n. Reveal()@n & n < i
```

The lemma quantifies over timepoints i, j, and n[1] and messages *tid* and *key*. It uses predicates of the form F`@`i to denote that the trace contains the fact F at index i and predicates of the form i`<`j to denote that the timepoint i is earlier than the timepoint j. The lemma states that if a thread `tid` has accepted a key `key` at timepoint `i` and `key` is also known to the adversary, then there must be a timepoint `l` prior to `i` where the shared secret was revealed.

Since 2015 [7], Tamarin can also handle equivalence properties. Equivalence properties are used to represent privacy properties, including anonymity and unlinkability, but can also be used for strong secrecy as well as real-or-random secrecy. This allows analysis of protocols for voting or e-cash. Equivalence properties are specified using a special `diff`-operator, similar to the ProVerif tool [10]. The `diff`-operator takes two parameters and can be used inside the terms in the protocol specification. A protocol specification gives rise to a labeled transition system. Using `diff`-terms creates two systems that are identical except in the values under such `diff`-terms. Tamarin will then try to prove that the two systems obtained by (1) replacing the `diff`-terms by their first parameter, and (2) replacing the `diff`-terms by their second parameter, are observationally equivalent.

[1] In Tamarin's input language, timepoint variables are prefixed with `#`, which we leave implicit here.

Previously, TAMARIN only supported user-defined equational theories that are subterm-convergent, meaning their right-hand side is a strict subterm of the left-hand side. Since 2017 [24], TAMARIN supports a much larger set of user-defined equational theories, which must only be convergent and have the finite variant property. This larger set of supported equational theories enables modeling, e.g., blind signatures. The equational theory for blind signatures contains an equation of the form `unblind( sign( blind(m,b), sk), b ) = sign(m,sk)` which was not admissible before as `sign(m,sk)` is not a subterm of the left-hand side of the equation. Using the new version of TAMARIN, verification of both e-cash and voting protocols was completed.

Output. Running TAMARIN on this input file yields the following output.

```
analyzed example.spthy: Accept_Secret (all-traces) verified (9 steps)
```

The output states that TAMARIN successfully verified that all protocol traces satisfy the formula in `Accept_Secret`.

Alternative Input. For the trace mode only, an alternative input language, similar to the applied-pi calculus, is available. This input gets automatically translated to TAMARIN's multi-set rewriting input using a sound and complete translator [29]. This simplifies use of TAMARIN for users experienced in applied-pi calculus based tools, like ProVerif, and also enables easier reuse of already existing protocol specifications that have been written for such tools.

2.2 Theoretical Foundations

A formal treatment of TAMARIN's foundations is given in the theses of Schmidt [40] and Meier [34]. For an equational theory E, a multi-set rewriting system R defining a protocol, and a guarded formula φ defining a trace property, TAMARIN can either check the validity or the satisfiability of φ for the traces of R modulo E. As usual, validity checking is reduced to checking the satisfiability of the negated formula.

For satisfiability checking, constraint solving is used to perform an exhaustive, symbolic search for executions with satisfying traces. The states of the search space are constraint systems. For example, a constraint can express that some multi-set rewriting step occurs in an execution or that one step occurs before another step. We can also directly use formulas as constraints to express that some behavior does *not occur* in an execution. Applications of constraint reduction rules, such as simplifications or case distinctions, correspond to the incremental construction of a satisfying trace. If no further rules can be applied and no satisfying trace was found, then no satisfying trace exists.

For symbolic reasoning, we exploit the finite variant property [17] to reduce reasoning modulo E with respect to R to reasoning modulo AC with respect to the variants of R using folding variant narrowing [27]. This enables TAMARIN to deal with a very large class of equational theories and since the last extension [24], user-specified equational theories only have to be convergent and ensure the finite variant property.

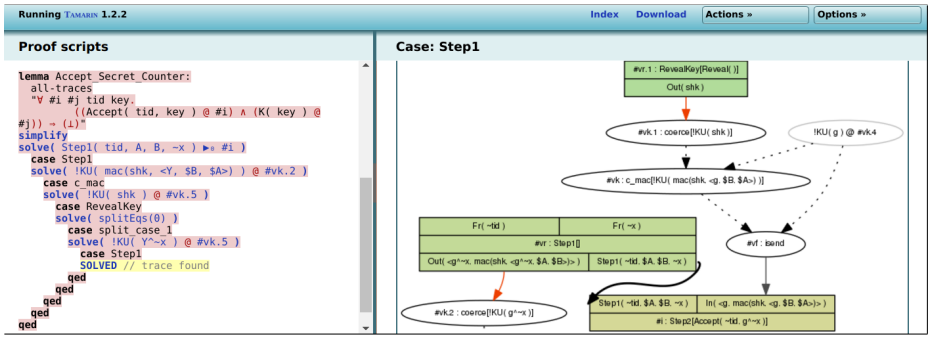

Fig. 1. Tamarin's interactive mode

Tamarin's equivalence mode is documented in [8]. In a nutshell, Tamarin computes all possible executions of the protocol on both systems using its constraint solving, and tries to find equivalent executions on the other side by mirroring. If these mirrors exist for all executions, then equivalence holds. If at least one execution does not have a mirror, then this represents a potential attack. As the equivalence mode is sound but not complete, such an attack may be spurious.

2.3 Implementation and Interactive Mode

Tamarin is written in the Haskell programming language. Its interactive mode is implemented as a webserver, serving HTML pages with embedded JavaScript. Tamarin's source code is publicly available from its webpage [43]. Figure 1 shows Tamarin's interactive mode, which integrates automated analysis and interactive proof guidance, and provides detailed information about the current constraints or counterexample traces. Users can carry out automated analysis of parts of the search space and perform partial unfoldings of the proof tree. Fully automated proof search is available on the command-line without the need to use the interactive mode and GUI.

3 Examples of Applications

Tamarin's flexible modeling framework and expressive property language make it suitable for analyzing a wide range of security problems. Table 1 shows selected results when using Tamarin in the automated mode. These results illustrate Tamarin's scope and effectiveness at unbounded verification and falsification. We now describe applications grouped by the features of Tamarin they used and what was achieved.

Key Exchange Protocols. We used Tamarin to analyze many authenticated key exchange protocols with respect to their intended adversary models [41]. These protocols typically include Diffie-Hellman exponentiation and are designed

	Protocol	Security property	Result	Time [s]	Details in
1.	KAS1	KI with Key Compromise Impersonation	proof	0.7	[41]
2.	NAXOS	eCK	proof	4.4	[41]
3.	STS-MAC	KI, adversary can register arbitrary public keys	attack	4.6	[41]
4.	STS-MAC-fix1	KI, adversary can register arbitrary public keys	proof	9.2	[41]
5.	STS-MAC-fix2	KI, adversary can register arbitrary public keys	proof	1.8	[41]
6.	TS1-2004	KI	attack	0.3	[41]
7.	TS2-2004	KI with weak Perfect Forward Secrecy	attack	0.5	[41]
8.	TS3-2004	KI with weak Perfect Forward Secrecy	non-termination	-	[41]
9.	UM	Perfect Forward Secrecy	attack	1.5	[41]
10.	TLS handshake	secrecy, injective agreement	proof	2.3	[34]
11.	TESLA 1	data authenticity	proof	4.4	[34]
12.	TESLA 2 (lossless)	data authenticity	proof	16.4	[34]
13.	Keyserver	keys are secret or revoked	proof	0.1	[34]
14.	Security Device	exclusivity (left or right)	proof	0.4	[34]
15.	Contract signing protocol	exclusivity (abort or resolve)	proof	0.8	[34]
16.	Envelope (no reboot)	denied access implies secrecy	proof	32.7	[34]
17.	SIGJOUX (tripartite)	Perfect Forward Secrecy	proof	102.9	[40]
18.	SIGJOUX (tripartite)	Perfect Forward Secrecy, ephemeral-key reveal	attack	111.5	[40]
19.	RYY (ID-based)	Perfect Forward Secrecy	proof	10.3	[40]
20.	RYY (ID-based)	Perfect Forward Secrecy, ephemeral-key reveal	attack	10.5	[40]
21.	YubiKey (multiset)	injective authentication	proof	19.3	[30]
22.	YubiHSM (multiset)	injective authentication	proof	7.6	[30]

Table 1. Selected results of the automated analysis of case studies included in the public TAMARIN repository. Here, KI denotes key independence.

to satisfy complex security properties, such as the eCK model [33]. Earlier works had only considered some of these protocols with respect to weaker adversaries, which cannot reveal random numbers and both short-term and long-term keys.

Loops and Mutable Global State. We also used TAMARIN to analyze protocols with loops and non-monotonic mutable global state. Examples include the TESLA protocols, the security device and contract signing examples from [3], the keyserver protocol from [35], and the exclusive secrets and envelope protocol models for TPMs from [23]. In each case, our results are more general or the analysis is more efficient than previous results. Additionally, TAMARIN was successfully used to analyze the YubiKey and YubiHSM protocols [30].

Protocols with Many Messages and Multiple Parties: ARPKI. We proposed a new public key infrastructure, called the Attack Resilient Public Key Infrastructure (ARPKI) [5,9]. ARPKI extended classic public key infrastructures using multiple certificate authorities and log servers. ARPKI was modeled and analyzed using TAMARIN, and only possible due to the support for mutable state.

Group Protocols and Bilinear Pairings. Using TAMARIN's support for bilinear pairing (BP) different group protocols were analyzed [42]. The group protocols STR and GDH based on Diffie-Hellman were verified, as was BP-based Group Joux. Note that these group protocols do not limit the number of participants and were proven for an arbitrary number of participants. Furthermore, the tripartite protocol Signed Joux and TAK1 were both each falsified and verified (property/adversary-model-dependent). Additional identity-based protocols RYY, Scott, and Chen-Kudla were similarly proven, respectively falsified, showing

exactly the weakest assumptions under which the protocols still satisfy their desired security properties. Details on the properties verified and automated verification time measurements are available [42, Table I].

Transport Layer Security (TLS). The largest case study so far in TAMARIN has been the upcoming IETF TLS 1.3 standard, which is the main foundation for Internet security and also widely used to establish secure channels in a variety of contexts. As of writing, TLS 1.3 is nearing completion. TLS comprises a complex combination of sub-protocols with intricate interactions that require loops and complex state. During the development process, TAMARIN was used to analyze different draft versions. For one of these proposals, TAMARIN found a critical attack [21]. TAMARIN was also used to verify the final revision of TLS 1.3 [22].

Non-Subterm Convergent Equational Theories. As TAMARIN supports any convergent equational theory that has the finite variant property, it can also be used to analyze protocols that use, for example, blind signatures or trapdoor commitments [24]. We have used it to study Chaum's digital cash protocol [14] which uses blind signatures and whose modeling also required the use of global state. We have verified anonymity, untraceability, as well as unforgeability, which states that no coins can be maliciously created. We also analyzed the FOO e-voting protocol [28], which relies on blind signatures. We have been able to check vote privacy (modeled as an equivalence property) and furthermore eligibility (modeled as a trace property). We additionally verified the Okamoto e-voting protocol [37], which relies on trapdoor commitments to achieve receipt-freeness. In particular we provided the first automated proof of receipt-freeness for this protocol.

Electronic Payment Protocols. Cortier et al. [18] used TAMARIN to verify a new EMV-compliant payment protocol, which is stateful as it uses tokens and counters. They verified complex security properties including a property stating that stolen payment tokens can only be used within a limited time window.

Liveness Properties and Fair Exchange Protocols. Thanks to the flexible way that properties are specified in TAMARIN, it is possible to express and verify certain liveness properties. For example in the case of fair exchange protocols, one can study timeliness and fairness [4]. This also required specifying resilient channels, i.e., channels where messages are eventually delivered, which can be accomplished using *restrictions* in TAMARIN. Restrictions are guarded first-order logic formulas; their use restricts TAMARIN to only consider traces that satisfy the specified restrictions.

Industrial Control Protocols We also used TAMARIN to verify industrial control protocols such as OPC-UA and variants of MODBUS [25]. We studied flow integrity properties, including liveness properties ("messages will be delivered") and ordering requirements ("messages are received in the same order they were sent").

Standardization While we have successfully used TAMARIN to provide increased assurance for security protocol standards, e.g., TLS 1.3 [21] and DNP3-SAv5 [20], such analyses are not yet routinely performed as part of the development process

of standards. In [6] it is argued that the quality of security protocol standards can be improved by integrating such analyses into the standardization process. TAMARIN's expressive framework is well suited for such analyses.

4 Related Work

There are many tools for the symbolic analysis of security protocols. We focus on those that can provide verification with respect to an unbounded number of sessions for complex properties. In general, the TAMARIN prover offers a novel combination of features that enables it to verify protocols and properties that were previously impossible to verify using other automated tools.

Like its predecessor the Scyther tool [19], TAMARIN performs backwards reasoning. However in contrast to Scyther, it supports equational theories, modeling complex control flow and mutable global state, an expressive property specification language, and the ability to combine interactive and automated reasoning.

The Maude-NPA tool [26] supports protocols specified as linear role-scripts, properties specified as symbolic states, and equational theories with a finite variant decomposition modulo AC, ACI, or C. It is unclear if our case studies that use global state, loops, and temporal formulas can be specified in Maude-NPA. With respect to their support of equational theories, Maude-NPA and TAMARIN are incomparable. For example, Maude-NPA has been applied to XOR and TAMARIN has been applied to bilinear pairing.

The ProVerif tool [10] has been extended to partially handle DH with inverses [32], bilinear pairings [38], and mutable global state [3]. From a user perspective, TAMARIN provides a more expressive property specification language that, e. g., allows for the direct specification of temporal properties. The effectiveness of ProVerif relies largely on its focus on the adversary's knowledge. It has more difficulty dealing with properties that depend on the precise state of agent sessions and mutable global state. The extension [3] for mutable global state is subject to several restrictions and the protocol models require additional manual abstraction steps. Similarly, the DH and bilinear pairing extensions work under some restrictions, e. g., exponents in the specification must be ground.

TAMARIN's observational equivalence notion has similarities with other notions of observational equivalence considered in the literature, including trace equivalence [16], bisimulation [1], and notions based on contexts or bi-processes [1,11,16].

Various other tools exist for verifying notions of observational equivalence but most are limited to a bounded number of sessions (e.g., [13,15,16]). ProVerif [11] verifies observational equivalence in the applied π-calculus for an unbounded number of sessions using bi-processes, but it cannot handle mutable state [2], for example, a protocol that switches between the states a and b. Also, TAMARIN supports a larger set of equational theories. For example, ProVerif can only deal with a weaker Diffie-Hellman equational theory approximation [31], which additionally does not support observational equivalence at all.

Another multi-set rewriting-based approach that supports observational equivalence is Maude-NPA [39]. It creates the synchronous product of two very similar protocols. Their approach suffers from termination problems [39] and thus presents only attacks.

5 Future Perspectives for Tamarin

Tamarin's future development will include evolution along the following four axis: Improving the tool's interface, extending the framework, improving reasoning methods, and improving heuristics.

Scaling the Tool's Interface. Tamarin has an extensive interactive mode that has been shown to be effective on many case studies. However, as the complexity of models grows, it becomes harder for humans to inspect the resulting proof states. As Tamarin's ability to deal with more complex models increases, it becomes increasingly important to improve its interactive mode to enable users to efficiently explore the proof states and applicable constraint rules. This may involve incorporating techniques from data visualization, filtering techniques, and heuristics to emphasize the most relevant information. This requires a substantial engineering effort (as opposed to fundamental research) that is critical to making scalable tools that can be adopted by a wide community.

Extending the Framework. As an ongoing avenue of research, there is still plenty of scope to *further support advanced equational* theories. The need to support new equational theories is driven by more detailed modeling of modern cryptographic primitives. As support for equational theories grows, more primitives can be incorporated. Conversely, as more cryptographic primitives are developed by cryptographers, the corresponding symbolic modeling generates the need to support the associated equational theories.

Tamarin currently supports a relatively coarse form of induction over protocol rule instances. However, there is no support for more fine-grained induction over all rule instances, in particular including the adversary's knowledge deduction steps. This means certain proof strategies currently cannot be mechanized, such as inductive arguments about all possible terms that can be derived by an adversary. This leads to the natural question of whether we can *improve support for induction* in Tamarin.

Tamarin currently does not use any form of abstraction or over-approximation of the adversary's behavior. While this makes counterexample generation easier, there is no fundamental reason why the tool should not support abstraction if that would enable it to analyze more problems. An open research question is to determine *under which conditions abstraction methods can improve* Tamarin*'s analysis*. As a starting point, one could consider works such as [36] that apply directly to Tamarin's predecessor, Scyther. The reason why these methods do not trivially translate to Tamarin is that more domain-specific specification

languages (such as Scyther's) have clearly defined notions of protocols, roles, and the adversary. In Tamarin's more expressive specification language, there are only abstract rewriting rules, which allow protocols to be modeled in many different ways. This modeling flexibility means that it is harder to reconstruct what a protocol or role is, which makes it harder to automatically determine when and how to apply domain-specific theorems.

It would be of general interest to further investigate classes of non-trace properties, including further variants of observational equivalence. This is very relevant for the security domain, as such properties also play a fundamental role in security definitions based on formalisms in the spirit of Universal Composability (UC) [12]. Proofs in these formalisms tend to revolve around proving simulatability with respect to so-called *ideal functionalities*, which in turn are processes. Even if we were to construct symbolic counterparts of these definitions, they currently can not be proven by Tamarin, as their structural differences preclude proving Tamarin's notion of equivalence. There are many exciting fundamental open questions in this area.

The Tamarin framework supports the modeling of a wide range of problems, but there are several interesting cases in which it currently does not yet automatically provide either a proof or a counterexample (attack). While this fundamentally cannot be avoided, and the interactive mode means the user is not stuck, we expect that there is substantial room to improve the level of automation by introducing new constraint solving rules (i.e., reasoning methods) and improving Tamarin's heuristics. We address these in turn.

Improving Reasoning Methods. One of the core ingredients of Tamarin's ability to construct proofs or find attacks is its *normal form conditions*. These conditions help restrict Tamarin's search space while retaining the correctness of the analysis results. Intuitively, they help by only considering efficient proofs or attacks without redundant steps. While we have proven that Tamarin's current normal form conditions retain correctness, it may well be possible to construct additional normal form conditions that would improve Tamarin's efficiency and even enable automatic proofs for protocols in which Tamarin currently requires manual intervention. However, if Tamarin is extended with further equational theories or constraint types, some of the current normal form conditions might no longer be sound, and weaker ones might need to be developed.

The main ingredient of Tamarin's analysis are the *constraint solving rules*. Intuitively, these encode specific proof methods, such as case distinctions, or drawing conclusions from combinations of constraints. As the tool is applied to more domains, different proof strategies might be needed, and we expect such case studies to drive the development of new constraint solving rules.

Improving the Heuristics. While the previously mentioned extensions would improve Tamarin's ability to manually construct proofs, they do not guarantee improved automation. As more constraint solving rules are introduced, it may

become harder to provide heuristics that are effective and efficient in most cases: if multiple rules can be applied in a certain proof state, which one should be used? In Tamarin, this is dealt with by the so-called *heuristics*: given a proof state and set of applicable constraint solving rules, they aim to select the optimal rule to apply, in the sense that it would yield the fastest termination, by either a proof or a counterexample. While the heuristic does not affect the correctness of the result, it strongly influences Tamarin's termination and efficiency. Improving the heuristic is a long-term goal and requires domain-specific investigations and obtaining further experience in case studies.

The optimal rule to apply strongly depends on the proof state and type of protocol. Thus, it may well be possible that different approaches are better suited to different subdomains. To facilitate this, Tamarin could employ a second type of heuristic, to detect protocol classes, mechanisms, or property types.

Putting all these improvements together should lead to a dramatic increase in Tamarin's scope and automation. This will accelerate its inclusion in the engineering and standardization process for protocols, as seen already with IETF's TLS 1.3 standard, collaboration with the Japanese standardization body for ISO/IEC 9798, and current work with mobile communications device vendors.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115, New York, 2001. ACM.
2. M. Arapinis, J. Phillips, E. Ritter, and M. D. Ryan. Statverif: Verification of stateful processes. *Journal of Computer Security*, 22(5):743–821, 2014.
3. M. Arapinis, E. Ritter, and M. Ryan. Statverif: Verification of stateful processes. In *Proc. CSF*. IEEE, 2011.
4. M. Backes, J. Dreier, S. Kremer, and R. Künnemann. A Novel Approach for Reasoning about Liveness in Cryptographic Protocols and its Application to Fair Exchange. In *2nd IEEE European Symposium on Security and Privacy (EuroS&P'17)*, Proceedings of the 2nd IEEE European Symposium on Security and Privacy, Paris, France, Apr. 2017. Springer.
5. D. Basin, C. Cremers, T. H. Kim, A. Perrig, R. Sasse, and P. Szalachowski. Design, analysis, and implementation of ARPKI: an attack resilient public-key infrastructure. *IEEE Transactions on Dependable and Secure Computing*, PP, Issue: 99, August 2016. http://dx.doi.org/10.1109/TDSC.2016.2601610.
6. D. Basin, C. Cremers, K. Miyazaki, S. Radomirovic, and D. Watanabe. Improving the security of cryptographic protocol standards. *IEEE Security & Privacy*, pages 24–31, 2014.
7. D. Basin, J. Dreier, and R. Sasse. Automated Symbolic Proofs of Observational Equivalence. In *22nd ACM SIGSAC Conference on Computer and Communications Security (ACM CCS 2015)*, pages 1144–1155, Denver, United States, Oct. 2015. ACM.
8. D. Basin, J. Dreier, and R. Sasse. Automated Symbolic Proofs of Observational Equivalence. Technical report, Oct. 2015. `https://hal.archives-ouvertes.fr/hal-01337409/file/ccs2015-extended.pdf`.

9. D. A. Basin, C. J. F. Cremers, T. H. Kim, A. Perrig, R. Sasse, and P. Szalachowski. ARPKI: attack resilient public-key infrastructure. In G. Ahn, M. Yung, and N. Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 382–393. ACM, 2014.
10. B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. CSFW*. IEEE, 2001.
11. B. Blanchet, M. Abadi, and C. Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, Feb.–Mar. 2008.
12. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`.
13. R. Chadha, Ştefan Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In H. Seidl, editor, *ESOP*, volume 7211 of *LNCS*, pages 108–127. Springer, 2012.
14. D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum Press, 1982.
15. V. Cheval. APTE: An algorithm for proving trace equivalence. In *TACAS*, volume 8413 of *LNCS*, pages 587–592. Springer, 2014.
16. V. Cheval, V. Cortier, and S. Delaune. Deciding equivalence-based properties using constraint solving. *Theor. Comput. Sci.*, 492:1–39, 2013.
17. H. Comon-Lundh and S. Delaune. The finite variant property: How to get rid of some algebraic properties. *Term Rewriting and Applications*, pages 294–307, 2005.
18. V. Cortier, A. Filipiak, J. Florent, S. Gharout, and J. Traoré. Designing and proving an emv-compliant payment protocol for mobile devices. In *2nd IEEE European Symposium on Security and Privacy (EuroSP'17)*, pages 467–480, 2017.
19. C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification*, volume 5123 of *LNCS*. Springer, 2008.
20. C. Cremers, M. Dehnel-Wild, and K. Milner. Secure authentication in the grid: A formal analysis of DNP3: SAv5. In *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings*, LNCS. Springer, 2017.
21. C. Cremers, M. Horvat, S. Scott, and T. van der Merwe. Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication. In *IEEE Symposium on Security and Privacy*, pages 470–485. IEEE Computer Society, 2016.
22. C. J. F. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, USA*, 2017.
23. S. Delaune, S. Kremer, M. D. Ryan, and G. Steel. Formal analysis of protocols based on TPM state registers. In *Proc. CSF*, pages 66–80. IEEE, 2011.
24. J. Dreier, C. Duménil, S. Kremer, and R. Sasse. Beyond Subterm-Convergent Equational Theories in Automated Verification of Stateful Protocols. In *6th International Conference on Principles of Security and Trust (POST)*, Uppsala, Sweden, Apr. 2017.
25. J. Dreier, M. Puys, M.-L. Potet, P. Lafourcade, and J.-L. Roch. Formally Verifying Flow Integrity Properties in Industrial Systems. In *SECRYPT 2017 - 14th International Conference on Security and Cryptography*, page 12, Madrid, Spain, July 2017.

26. S. Escobar, C. Meadows, and J. Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. *TCS*, 367:162–202, 2006.
27. S. Escobar, R. Sasse, and J. Meseguer. Folding variant narrowing and optimal variant termination. *J. Log. Algebr. Program.*, 81(7-8):898–928, 2012.
28. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1992.
29. S. Kremer and R. Künnemann. Automated analysis of security protocols with global state. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 163–178, 2014.
30. R. Künnemann and G. Steel. YubiSecure? Formal security analysis results for the YubiKey and YubiHSM. In *Preliminary Proc. STM'12*, 2012.
31. R. Küsters and T. Truderung. Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In *Computer Security Foundations Symposium (CSF)*, pages 157–171. IEEE, 2009.
32. R. Küsters and T. Truderung. Reducing protocol analysis with xor to the xor-free case in the Horn theory based approach. *J. Autom. Reasoning*, 46(3-4):325–352, 2011.
33. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *ProvSec*, volume 4784 of *LNCS*, pages 1–16. Springer, 2007.
34. S. Meier. *Advancing Automated Security Protocol Verification*. PhD thesis, 2013.
35. S. Mödersheim. Abstraction by set-membership: verifying security protocols and web services with databases. In *Proc. CCS*, pages 351–360. ACM, 2010.
36. T. B. Nguyen and C. Sprenger. Abstractions for security protocol verification. In *POST*, volume 9036 of *Lecture Notes in Computer Science*, pages 196–215. Springer, 2015.
37. T. Okamoto. An electronic voting scheme. In *IFIP World Conference on IT Tools*, pages 21–30, 1996.
38. A. Pankova and P. Laud. Symbolic analysis of cryptographic protocols containing bilinear pairings. In *Proc. CSF*. IEEE, 2012.
39. S. Santiago, S. Escobar, C. Meadows, and J. Meseguer. A formal definition of protocol indistinguishability and its verification using Maude-NPA. In *Security and Trust Management (STM) 2014*, pages 162–177. Springer, 2014.
40. B. Schmidt. *Formal Analysis of Key Exchange Protocols and Physical Protocols*. PhD thesis, 2012.
41. B. Schmidt, S. Meier, C. Cremers, and D. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *Proc. CSF*. IEEE, 2012.
42. B. Schmidt, R. Sasse, C. Cremers, and D. A. Basin. Automated verification of group key agreement protocols. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 179–194, 2014.
43. T. Tamarin team. The Tamarin prover: source code, documentation, and case studies, August 2017. Available `http://tamarin-prover.github.io/`.