Rule Variant Restrictions for the Tamarin Prover

Felix Linker[0009−0000−7886−4480]

Department of Computer Science, ETH Zurich, Zürich, Switzerland
flinker@inf.ethz.ch

Abstract. We introduce an optimization to the Tamarin prover that reduces its search space. The optimization applies to protocol models that use equational theories with cancellative operators, for example, when modelling Diffie-Hellman groups or bilinear pairings. We prove the optimization's soundness and evaluate its performance.

1 Introduction

The Tamarin prover [7,9] is a state-of-the-art model checker for security protocols. It was originally developed in two doctoral theses [6,8], supervised by David Basin. In this work, I will present an optimization to the Tamarin prover that I identified while formally verifying the SecureDrop protocol [2]. I worked on the SecureDrop project while I was finalizing my doctoral thesis, also supervised by David. While the optimization may seem conceptually simple, I would neither have been able to identify it nor have been able to prove its soundness prior to my doctorate. This work showed me how much David had taught me during my doctorate, which is why I would like to dedicate this work to him.

Tamarin operates in the so-called *symbolic model*, in which cryptography is modelled by an *equational theory*. Concretely, cryptographic messages are represented by terms and uninterpreted functions, and the effects of cryptographic operations by equations on these terms. Tamarin reasons modulo such an equational theory but only considers equational theories with specific properties. These properties allow Tamarin to compute the finite set of *variants* associated to a term, which represent all potential shapes the term might take. Unifying two terms modulo a full equational theory, including associativity and commutativity, can be implemented by unifying the two terms' variants modulo associativity and commutativity only, which greatly simplifies Tamarin's proof search.

Constructs that are more complex than terms, for example, protocol rules, can also have variants, and Tamarin computes protocol rule variants during a precomputation phase. In this paper, we prove that certain rule variants correspond to protocol rule instances that could never occur in a run of a protocol. Thus, any constraint system containing them can be considered contradictory and one can safely ignore such rule variants. We experienced significant performance improvements when using this pruning mechanism, in particular when modelling protocols that use Diffie-Hellman group operators. Our optimization has been integrated into Tamarin's code base.[1]

[1] See https://github.com/tamarin-prover/tamarin-prover/pull/763.

Structure We proceed as follows. We introduce relevant background in Section 2. Then, in Section 3, we introduce our optimization and prove its soundness by relating it to a soundness proof in one of the original theses describing Tamarin. We evaluate our optimization's performance in Section 4.

As our optimization's soundness proof requires substantial background, Section 2 is quite long. The benefit of that, though, is that this paper is mostly self-contained, and may serve as a helpful introduction to the finite variant property and its application to the Tamarin prover. For a more complete introduction, we refer the reader to [3].

2 Background

We introduce the Tamarin prover following [6]. We omit certain details for clarity, and refer to [6] for full details. To contextualize each subsection, we briefly provide an overview of this section's contents.

Tamarin works in the *symbolic model*, where protocol messages are represented as elements of a term algebra over a signature Σ. An equational theory encodes the semantics of these terms. Tamarin only considers theories that have the *finite variant property*, which allows it to efficiently implement unification. In Section 2.1, we introduce equational theories, the finite variant property, and other background relevant for Tamarin protocol models and their semantics.

In Section 2.2, we introduce Tamarin protocol models, which comprise an equational theory and a set of multiset rewriting rules. Multiset rewriting rules constitute a protocol model's state transition system and model the behavior of honest protocol participants, the adversary, and domain assumptions such as who controls which cryptographic key.

In Section 2.3, we define the semantics of Tamarin protocol models. In short, a protocol model's semantics are the set of traces associated with its state transition system, i.e., sequences of multiset rewriting rule applications.

Utilizing the finite variant property, Tamarin only considers traces that are in a normal form with respect to the protocol model's equational theory, and we describe Tamarin's normal-form reasoning in Section 2.4. For our optimization, we will require the result of Theorem 3, which shows that reasoning about a set of protocol model rules modulo a full equational theory is equivalent to reasoning about a set of protocol model rule *variants* modulo associativity and commutativity (if applicable).

2.1 Preliminaries

Order-Sorted Term Algebras An *order-sorted signature* $\Sigma = (S, \leq, F)$ consists of a set of sorts S, a partial order $(\leq) \subseteq S \times S$ on the sorts, and a set of function symbols F. We write f/n for a function symbol $f \in F$ with arity n. Every sort $s \in S$ has an infinite set $\mathcal{V}_s$ of variables and an infinite set $\mathcal{C}_s$ of constants; all these sets are pairwise disjoint. We write $v{:}s$ for $v \in \mathcal{V}_s$ and $\mathcal{V} = \bigcup_{s \in S} \mathcal{V}_s$ for the set of all variables. Function symbols in F have the form

$f : s_1 \times \cdots \times s_n \to s$ (where $s_1, \ldots, s_n, s \in S$). Variables, constants, and sort-respecting applications of function symbols to terms are terms. For $s \in S$, we use $\mathcal{T}^s_\Sigma$ to denote the terms of sort s. The set of all terms of an order-sorted signature is $\mathcal{T}_\Sigma = \bigcup_{s \in S} \mathcal{T}^s_\Sigma$. A term is *ground* if it contains no variables.

A substitution is a function $\sigma : \mathcal{V} \to \mathcal{T}_\Sigma$. We require that substitutions respect sorts, i.e., variables of sort s must be mapped to terms of sort $s' \leq s$. We extend substitutions homomorphically to terms and other constructs (to be defined later) as usual, and write $t\sigma$ for an application of σ to a term $t \in \mathcal{T}_\Sigma$. A *valuation* is a substitution mapping variables to ground terms.

Term Rewriting A *position* p in a term is a list of indices. We write $t|_p$ for t*'s subterm at position* p, and define it inductively by

$$t|_p := \begin{cases} t & p = [] \\ t_{i_0}|_{[i_1,\ldots,i_n]} & p = [i_0, i_1, \ldots, i_n], t = f(t_1, \ldots, t_k), 1 \leq i_0 \leq k \\ \bot & \text{otherwise} \end{cases}$$

We write $t[s]_p$ to denote the term obtained by replacing $t|_p$ in t with s. We say that t is a *syntactic subterm* of t' if there exists a position p such that $t'|_p = t$. Whenever this is the case, we write $t \sqsubseteq t'$.

A *rewrite rule* is an ordered pair of terms $l \to r$. A *rewriting system* $\mathcal{R}$ is a set of rewrite rules. The corresponding *rewrite relation* $\to_\mathcal{R}$ is defined as $s \to_\mathcal{R} t$ if there is a position p in s, a rewrite rule $l \to r \in \mathcal{R}$, and a substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$.

A rewriting system $\mathcal{R}$ is *terminating* if there is no infinite sequence of terms $(t_i)_{i \in \mathbb{N}}$ such that $t_i \to_\mathcal{R} t_{i+1}$ $(i \in \mathbb{N})$. $\mathcal{R}$ is *confluent* when for all terms t, s_1, and s_2, if $t \to^*_\mathcal{R} s_1$ and $t \to^*_\mathcal{R} s_2$, then there exists a term t' such that $s_1 \to^*_\mathcal{R} t'$ and $s_2 \to^*_\mathcal{R} t'$. $\mathcal{R}$ is *convergent* if it is terminating and confluent. We use $t{\downarrow_\mathcal{R}}$ to denote a term t's normal form in a convergent rewriting system $\mathcal{R}$, which is the term t' such that $t \to^*_\mathcal{R} t'$ and there exists no term t'' such that $t' \to_\mathcal{R} t''$. Observe that every term's normal form is uniquely determined.

Equational Theories Protocol models are defined with respect to an order-sorted signature Σ and a set of equations E that have the form $s \simeq t$ for $s, t \in \mathcal{T}_\Sigma$. Typically, the equations model the relationships between different cryptographic operations, e.g., that decryption is the inverse operation to encryption. $\Xi = (\Sigma, E)$ is an *equational presentation*. An *equational theory* $=_\Xi$ is the smallest congruence relation over Σ containing all instances of E. We often leave Σ implicit and identify Ξ by E. Two terms s and t are *equal modulo* E if $t =_E s$. We use an E-subscript to denote relations modulo E.

If an equational theory E's equations can be oriented to a convergent rewriting system $\mathcal{R}$, then $t =_E s$ if and only if $t{\downarrow_\mathcal{R}} = s{\downarrow_\mathcal{R}}$ [5,4]. Some equations, e.g., commutativity, cannot be oriented, and in those cases we split an equational theory E into oriented rewriting rules $\mathcal{R}$ and unoriented equations AX. The *rewrite relation* $\to_{\mathcal{R},AX}$ is defined as $s \to_{\mathcal{R},AX} t$ if there is a position p in s, a rewrite

rule $l \to r \in \mathcal{R}$, and a substitution σ such that $s|_p =_{AX} l\sigma$ and $t = s[r\sigma]_p$. We say that $\mathcal{R}$ is AX-convergent if $\to_{\mathcal{R},AX}$ is terminating and confluent modulo AX. We write $t{\downarrow}^{\mathcal{R}}_{AX}$ to denote t's normal form with respect to $\to_{\mathcal{R},AX}$. $\mathcal{R}$ is AX-coherent when for all terms t_1, t_2, and t_3, if $t_1 \to_{\mathcal{R},AX} t_2$ and $t_1 =_{AX} t_3$, then there exist terms t_4 and t_5 such that $t_4 =_{AX} t_5$ and $t_2 \to^*_{\mathcal{R},AX} t_4$ and $t_3 \to^+_{\mathcal{R},AX} t_5$. We write $\mathcal{R}^{\simeq} := \{l \simeq r \mid l \to r \in \mathcal{R}\}$ for a rewriting system $\mathcal{R}$'s associated set of equations. For $E = \mathcal{R}^{\simeq} \cup AX$, if $\mathcal{R}$ is AX-convergent and AX-coherent, then $t =_E s$ if and only if $t{\downarrow}^{\mathcal{R}}_{AX} =_{AX} s{\downarrow}^{\mathcal{R}}_{AX}$ [5,4].

Multisets A *multiset* $\{a, b, \dots\}^\sharp$ is a set of items a, b, etc., and items can repeat. We extend the usual operations and constants for sets to multisets using the $\sharp$ superscript, e.g., $\emptyset^\sharp$ is the empty multiset. Whenever we use relations, e.g., $\subseteq^\sharp$, or functions, e.g., $\cup^\sharp$, on multisets, they take cardinality into account.

2.2 Protocol Specifications

A Tamarin model is a state transition system, defined by multiset rewriting rules. The global state consists of a multiset of ground facts, and the rewriting rules define how the state is updated. Facts are similar to predicates in first-order logic and are of the form $F(t_1, \dots, t_n)$. F is a fact symbol from a fact signature Σ_{Fact}, and $t_1, \dots, t_n$ are terms defined over the model's signature Σ. We fix the sorts in Σ to *msg*, *fresh*, *pub*, and *nat*. The sorts *fresh*, *pub*, and *nat* are subsorts of *msg*. The sort *fresh* models fresh, unguessable values such as random numbers.

A fact is ground if all its terms are ground. Facts typically model the local state of participants or the network, e.g., who possesses which keys, or what messages have been sent over the network. We consider the special, reserved fact symbols Fr, In, Out, $!\mathsf{K}^\uparrow$, and $!\mathsf{K}^\downarrow$ of arity 1 in Σ_{Fact}. The symbol Fr models generating a fresh, unguessable value, In and Out model receiving and sending a message over an insecure network, and $!\mathsf{K}^\uparrow$ and $!\mathsf{K}^\downarrow$ model adversary knowledge.

Multiset rewriting rules in Tamarin have the form

$$[l_1, \dots, l_m] \; -[a_1, \dots, a_n]\!\to \; [r_1, \dots, r_o].$$

l_i, a_j, and r_k are facts and may contain variables ($1 \le i \le m$, $1 \le j \le n$, $1 \le k \le o$). $l_1, \dots, l_m$ are the rule's *premises*, $a_1, \dots, a_n$ are the rule's *actions* (also called *action facts*), and $r_1, \dots, r_o$ are the rule's *conclusions*. We write $\to$ instead of $-[]\!\to$ when a rule has no actions. Rules can be instantiated by a substitution σ. One rule ri is another rule r's *instance* if there exists a substitution σ such that $r\sigma = ri$. We write $prems(r)$, $acts(r)$, and $concs(r)$ for a rule r's list of premises, actions, and conclusions respectively. A rule or rule instance is ground if all its facts are ground.

Facts are either linear or persistent (preceded by a !). When a rule is applied, the linear (but not the persistent) facts in its premise are removed from the global state and replaced by the facts in its conclusion.

A set of multiset rewriting rules P is a *multiset rewriting system* if it satisfies certain well-formedness restrictions. We introduce three restrictions here, but omit others that are irrelevant to our work:

(i) no rule uses a fresh constant,
(ii) no rule's conclusion contains an Fr fact, and
(iii) all *fresh* and *msg* variables used in a rule's conclusion are introduced in the rule's premise.

Built-In Rules Tamarin uses a number of built-in protocol rules. One rule, FRESH, models the generation of random values by introducing a new Fr fact in its conclusion. Importantly, no other rule introduces an Fr fact.

$$\text{FRESH} := [] \rightarrow [\mathsf{Fr}(x{:}\mathit{fresh})].$$

Other rules, referred to as ND (normal-form message deduction rules), model adversary deduction according to the model's equational theory E. For this work, the following rule in ND, which allows the attacker to generate new fresh values, is particularly relevant:

$$[\mathsf{Fr}(x{:}\mathit{fresh})] \; -[\mathsf{!K}^{\uparrow}(x{:}\mathit{fresh})]\rightarrow \; [\mathsf{!K}^{\uparrow}(x{:}\mathit{fresh})].$$

2.3 Semantics

Executions and Traces Given a multiset rewriting system P, adversary deduction rules ND, an equational theory E, and $R = P \cup ND \cup \{\text{FRESH}\}$, the tuple (R, E) is a *Tamarin* or *protocol model*. Tamarin models induce a labeled state transition system. The initial state is the empty multiset $\emptyset^{\sharp}$. A state transition is the application of a ground rule instance $ri = l \; -[a]\rightarrow \; r$. We use $\mathsf{lfacts}(x)$ and $\mathsf{pfacts}(x)$ to denote the linear and persistent facts in x, respectively. A rule instance ri can be applied if its premises are contained in the global state, which is then updated accordingly. Formally, the state transition relation $\Rightarrow_{(R,E)}$ is:

$$S \Rightarrow_{(R,E)} (S \setminus^{\sharp} \mathsf{lfacts}(l)) \cup^{\sharp} r \qquad \text{if } \mathsf{lfacts}(l) \subseteq^{\sharp} S, \mathsf{pfacts}(l) \subseteq \text{set}(S)$$

An *execution* is a sequence of labeled state transitions. We only consider executions where each instance of FRESH is unique. Each execution has a corresponding *trace*, which is the list of sets of action facts labeling each transition. The semantics of a Tamarin model is its set of traces.

Dependency Graphs Dependency graphs capture the sequence of rule applications and dependencies between rule premises and conclusions.

Definition 1 (Dependency graph). *Let (R, E) be a Tamarin model. A* dependency graph *is a tuple $dg = (I, D)$ where I is a finite list of ground rule instances in R, and $D \subseteq \mathbb{N}^2 \times \mathbb{N}^2$. A rule's index in I is its* timepoint. *We write $(i, u) \rightarrowtail (j, v)$ for $((i, u), (j, v)) \in D$ when D is clear from the context. Here, (i, u) denotes the conclusion with index u of the rule with index i in I. Similarly, (j, v) denotes the premise with index v of the rule with index j. We require that edges correctly connect premises and conclusions, i.e., $i < j$ and the conclusion (i, u) is equal modulo E to the premise (j, v). Furthermore, each premise must have exactly one incoming edge, every linear conclusion has at most one outgoing edge, and distinct instances of* FRESH *must generate distinct fresh values.*

Definition 2 (Trace). *A* trace *induced by a dependency graph $dg = (I, D)$ is a list $trace(dg)$ of sets of action facts. Concretely, $trace(dg)$ is of length $|I|$, and every index $i \in I$ corresponds to the set of action facts at timepoint i, i.e., to* $\mathrm{set}(acts(I_i))$.

One can show that the dependency graphs of a model (R, E) induce the same set of traces as the model's executions [6, Theorem 3]. We write $dgraphs_E(R)$ for all dependency graphs of a protocol model (R, E).

Trace Properties In Tamarin, protocol properties are expressed as *trace properties*, which are given by closed first-order *trace formulas*. In the following, we briefly relate trace properties to protocol models, but do not fully define them and their semantics. We refer to [6] for a complete definition.

Given a valuation θ of the free variables of φ, we write $(tr, \theta) \vDash_E \varphi$ to mean that the trace tr satisfies φ under the valuation θ. A trace formula φ is *valid* for a protocol model (R, E), written $R \vDash_E^\forall \varphi$, if for all traces tr of (R, E) and every valuation θ, one has $(tr, \theta) \vDash_E \varphi$.

2.4 Normal Form Dependency Graphs

Let E be an equational theory, let $\mathcal{R}$ be a rewriting system, and let $AX \subseteq E$ contain only equations for associativity and commutativity. Assume that (i) $E = \mathcal{R}^{\simeq} \cup AX$ and (ii) $\mathcal{R}$ is AX-convergent and AX-coherent. E has the *finite variant property* for $\mathcal{R}$ and AX if for all terms t there is a finite set of substitutions $T = \{\tau_1, \ldots, \tau_k\}$ such that for all substitutions σ there is a substitution $\tau \in T$ and a substitution σ' so that $t\sigma{\downarrow}_{AX}^{\mathcal{R}} =_{AX} (t\tau{\downarrow}_{AX}^{\mathcal{R}})\sigma'$ and, for all variables x occurring in t, we have $x\sigma{\downarrow}_{AX}^{\mathcal{R}} =_{AX} x\tau\sigma'$.

In the following, we will only consider equational theories E and associated equational presentations $\mathcal{R}^{\simeq} \cup AX$ of $=_E$ such that E has the finite variant property and such that $\mathcal{R}$ is AX-convergent and AX-coherent. When E has the finite variant property for $\mathcal{R}$, the set $\lceil t \rceil_{AX}^{\mathcal{R}} := \{(t\tau_i{\downarrow}_{AX}^{\mathcal{R}}, \tau_i) \mid 1 \leq i \leq k\}$ is a *complete set of $\mathcal{R}, AX$-variants of t*. We extend the notion of complete variants to multiset rewriting rules and multiset rewriting systems straightforwardly. Intuitively, the variants of a term or rule allow us to reason modulo E by reasoning modulo normal forms and AX only.

Theorem 3 (Changing the Equational Theory [6, p. 114]). *For every guarded*[2] *trace property φ and protocol model (R, E) where $\mathcal{R}^{\simeq} \cup AX$ is an equational presentation of $=_E$:*

$$R \vDash_E^\forall \varphi \iff \{trace(dg) \mid dg \in dgraphs_{AX}(\lceil R \rceil_{AX}^{\mathcal{R}}), dg \text{ is } {\downarrow}_{AX}^{\mathcal{R}}\text{-normal}\} \vDash_{AX}^\forall \varphi.$$

[2] Tamarin only considers trace properties that are *guarded*, i.e., a subset of all possible trace properties. The details of guardedness are irrelevant to this work, and we thus omit a formal definition. For a definition, see [6].

3 Rule Variant Restrictions

Our optimization applies to theories that use operators which can cancel out and lead to multiple rule variants. For example, Tamarin provides built-in support for an equational theory modelling Diffie-Hellman groups. That theory supports the operators $(\cdot)/2$ (multiplication), $\circ^{\circ}/2$ (exponentiation), $\circ^{-1}/1$ (inverse), and $\epsilon/0$ (neutral element) as well as, in particular, the following equations:

$$x \cdot x^{-1} = \epsilon$$
$$x^{\epsilon} = x$$

Our optimization was motivated by rules such as the following:

$$[\mathsf{Fr}(x{:}\mathit{fresh}), \mathsf{In}(m)] \rightarrow [\mathsf{Out}(m^x)].$$

In the above rule, a protocol participant receives a message m over the network, raises it to the power of a fresh, unguessable value x, and sends the result over the network. During precomputation, Tamarin will consider the case in which raising m to x might cancel out, which it represents with the rule variant:

$$[\mathsf{Fr}(x{:}\mathit{fresh}), \mathsf{In}(z^{(x:\mathit{fresh})^{-1}})] \rightarrow [\mathsf{Out}(z)].$$

Note, though, that constructing $m = z^{(x:\mathit{fresh})^{-1}}$ requires knowledge of x to raise z to x's inverse. The above rule, though, freshly introduces x. Thus, constructing m requires x to be used before it is introduced, which is impossible.

Fresh-contradictory rule variants, which we define below, formalize the above intuition. Fresh-contradictory rule variants use a fresh value before it is introduced. We show that one can remove such variants from a protocol model without changing the model's associated set of normal-form dependency graphs.

Definition 4. *Let* $r = [p_1, \ldots, p_n] \; -[\ldots]\rightarrow \; [\ldots]$ *be a protocol rule and let* $\lceil r \rceil^{\mathcal{R}}_{AX} = \{(r_i, \tau_i)\}_{1 \leq i \leq m}$ *be a complete set of associated variants for a rewriting system* $\mathcal{R}$ *and associative and commutative equations* AX*. A variant* $(r_i, \tau_i) \in \lceil r \rceil^{\mathcal{R}}_{AX}$ *is* fresh-contradictory *if* r *has two distinct premises* $p_k = \mathsf{F}(\ldots, t, \ldots)$ *(*$\mathsf{F} \neq \mathsf{Fr}$*) and* $p_l = \mathsf{Fr}(x)$ *such that* $x\tau_i \sqsubseteq t\tau_i{\downarrow}^{\mathcal{R}}_{AX}$*.*

We write $\lceil r \rceil^{\mathcal{R},\perp}_{AX} \subseteq \lceil r \rceil^{\mathcal{R}}_{AX}$ for the fresh-contradictory variants of a rule r and extend this notation to sets of multiset rewriting rules.

Lemma 5. *For every protocol model* (R, E) *where* $\mathcal{R}^{\simeq} \cup AX$ *is an equational presentation of* $=_E$ *and* AX *only contains equations for associativity and commutativity:*

$$\{dg{\downarrow}^{\mathcal{R}}_{AX} \mid dg \in \mathit{dgraphs}_{AX}(\lceil R \rceil^{\mathcal{R}}_{AX})\} = \{dg{\downarrow}^{\mathcal{R}}_{AX} \mid dg \in \mathit{dgraphs}_{AX}(\lceil R \rceil^{\mathcal{R}}_{AX} \setminus \lceil R \rceil^{\mathcal{R},\perp}_{AX})\}.$$

Proof. Let

$$NDG := \{dg{\downarrow}^{\mathcal{R}}_{AX} \mid dg \in dgraphs_{AX}(\lceil R \rceil^{\mathcal{R}}_{AX})\}$$
$$NDG_{\neg\bot} := \{dg{\downarrow}^{\mathcal{R}}_{AX} \mid dg \in dgraphs_{AX}(\lceil R \rceil^{\mathcal{R}}_{AX} \setminus \lceil R \rceil^{\mathcal{R},\bot}_{AX})\}.$$

It is clear that $NDG_{\neg\bot} \subseteq NDG$. We show that $NDG \subseteq NDG_{\neg\bot}$ by contradiction. Suppose that there is a dependency graph $dg = (I, D) \in NDG \setminus NDG_{\neg\bot}$. By assumption, we know that all rule instances $ri \in I$ are ${\downarrow}^{\mathcal{R}}_{AX}$-normal (as dg is ${\downarrow}^{\mathcal{R}}_{AX}$-normal). Thus, by the finite variant property, we know that for each rule instance $ri \in I$ there must be a rule variant $(r, \tau) \in \lceil R \rceil^{\mathcal{R}}_{AX}$ so that $ri =_{AX} r\sigma$ for some substitution σ (note that r is ${\downarrow}^{\mathcal{R}}_{AX}$-normal by definition). There must be some rule instance ri for which its associated variant (r, τ) (as per above) is in $\lceil R \rceil^{\mathcal{R},\bot}_{AX}$. If not, dg would be in $NDG_{\neg\bot}$. We fix such a rule instance ri and associated variant (r, τ).

As r is a fresh-contradictory variant, there must be some fresh variable x and term t occurring in r such that $x \sqsubseteq t$ and x is introduced in r, i.e., r has a $\mathsf{Fr}(x)$ premise. Because $ri =_{AX} r\sigma$ and because equations in AX can never remove terms, we have that (i) $\mathsf{Fr}(x\sigma)$ occurs in ri, (ii) $t\sigma$ occurs in ri, and (iii) $x\sigma$ occurs in $t\sigma$. As ri is ${\downarrow}^{\mathcal{R}}_{AX}$-normal (cannot be reduced), $t\sigma$ in ri must have been constructed using $x\sigma$. Therefore, there must be another rule instance in dg that introduces $x\sigma$ to construct $t\sigma$ and that thus also contains a premise $\mathsf{Fr}(x\sigma)$. This, however, contradicts that instances of Fresh are unique.

We can use this lemma to specialize Theorem 3 as follows.

Theorem 6. *For every guarded trace property φ and protocol model (R, E) where $\mathcal{R}^{\simeq} \cup AX$ is an equational presentation of $=_E$ and AX only contains equations for associativity and commutativity:*

$$R \vDash^{\forall}_{E} \varphi \iff$$
$$\{trace(dg) \mid dg \in dgraphs_{AX}(\lceil R \rceil^{\mathcal{R}}_{AX} \setminus \lceil R \rceil^{\mathcal{R},\bot}_{AX}), dg \text{ is } {\downarrow}^{\mathcal{R}}_{AX}\text{-normal}\} \vDash^{\forall}_{AX} \varphi.$$

Proof. This follows directly from Theorem 3 and Lemma 5.

Theorem 6 allows us to prune fresh-contradictory variants during Tamarin's precomputations.

4 Implementation and Evaluation

We implemented our optimization by discarding fresh-contradictory rule variants during Tamarin's precomputations. The implementation was straightforward and only required around 20 lines of code at its core.[3]

We evaluated whether the optimization improves performance on 19 authenticated key exchange examples, all of which use equational theories with cancellative operators to model Diffie–Hellman exponentiation and bilinear pairings.

[3] See `https://github.com/tamarin-prover/tamarin-prover/pull/763`.

These examples were taken from the Tamarin prover repository [1] (located at `examples/ake/`). We timed both precomputations and proof construction (which includes precomputations) with and without our optimization, taking the average over 6 runs on a laptop with an Apple M2 Max CPU and 32 GB of RAM.

Figures 1 shows that our optimization generally improves the performance of Tamarin's proof search. As shown on the right, proof time improves for about 75% of the theories. For 50% of theories, the improvement is at least 5%, and for 25% of theories, there is an improvement of around 20%-30%. Observe that for large theories, which can take days to verify, such improvements can bring massive time savings. When proof time increases, it does so by only 5% at most.

To understand why our optimization can also degrade performance, we manually investigated some theories, for which proof time increases. With our optimization, we observed that Tamarin solves some equations earlier than before, although solving these equations does not aid the overall proof. This is because Tamarin's heuristics prioritize solving equations with many and few cases differently, and our optimization reduces some equations' cases, leading it to being prioritized differently. We thus attribute worsened performance to unfavorable interactions between our optimization and existing Tamarin heuristics.

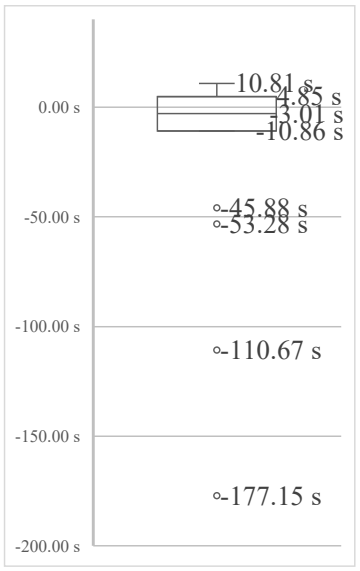
Figure 2 illustrates that our optimization also improves the performance of Tamarin's precomputations for all but one theory. These improvements, however, are not as drastic as for proof search, because the absolute changes are negligible, except for some outliers. The negative outlier, for which precomputation time increases, is a theory that does not benefit from the optimization, but that also has many rule variants due to its equational theory. We conjecture that checking for fresh-contradictory rule variants takes extra time but does not benefit the more complex precomputations that follow.

Overall, we regard our optimization as a net improvement to Tamarin, although it also led to decreased performance in some cases due to unfavorable interactions with Tamarin's proof search heuristics. As for the formal analysis of the SecureDrop protocol [2], which initially motivated this work, we observed that proof search was only feasible when using our optimization. Without it, Tamarin would run out of memory on a machine using 252 GB of memory.

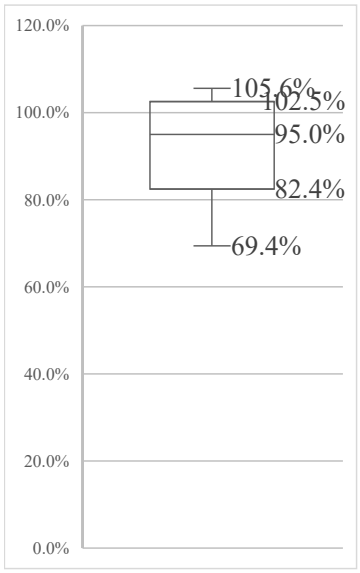
5 Conclusion

In this work, we presented an optimization to the Tamarin prover, proved its soundness, and evaluated its performance. To reason efficiently, Tamarin precomputes rule variants. Our optimization identifies redundant rule variants, which we call fresh-contradictory and which can be omitted from precomputation and thus also from proof search.

Future Work We identified the presented optimization during proof search, when we observed that the adversary would use fresh values before they are introduced. Tamarin already recognizes the use of fresh values before they are introduced as a contradiction, but only when they do not occur below a reducible function



Absolute change of proof search.

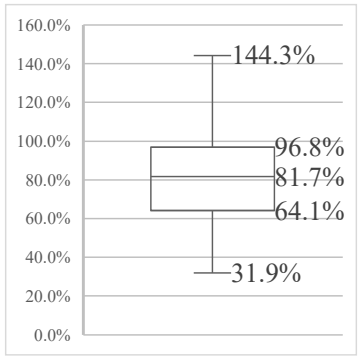


Relative change of proof search.

Fig. 1. Proof-search results of the optimization as box plots. A box plot illustrates the median value (middle line in box), upper and lower quartiles (the boxes' upper and lower halves), maximum and minimum values (lines above and below the boxes), as well as outliers (dots). The plots respectively display the absolute and relative time difference when enabling the optimization, where lower is better. For example, a value of -10 s means that time decreased by 10 seconds, and a value of 60% means that the optimization reduced the time to 60% of what it was without the optimization.



Absolute change of precomputations.



Relative change of precomputations.

Fig. 2. Precomputation-time results of the optimization as box plots. The plots respectively display the absolute or relative time difference when enabling the optimization, where lower is better. The absolute differences are so small that the median, quartiles, minimum, and maximum are all close to 0 s; for clarity, their data labels are omitted.

symbol such as exponentiation. For future work and as an alternative to our optimization, one could explore lifting or relaxing that restriction. Moreover, in Section 4, we discussed that Tamarin's heuristics do not always interact favorably with our optimization. Thus, we suggest updating Tamarin's heuristics to account for our optimization.

References

1. The Tamarin prover repository, https://github.com/tamarin-prover/tamarin-prover/tree/4b6239f1386912a2ae15d71c75bcad1bdcafdce1/
2. Berra, G., Linker, F., Maier, L., Myers, C.F., Paterson, K.G., Shane, R., Veitch, S.: The SecureDrop Protocol: End-to-End Encrypted Whistleblowing for All (Jul 2026), https://eprint.iacr.org/2026/1484
3. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. The Journal of Logic and Algebraic Programming **81**(7) (Oct 2012). https://doi.org/10.1016/j.jlap.2012.01.002
4. Hullot, J.M.: Canonical forms and unification. In: 5th Conference on Automated Deduction Les Arcs, France, July 8–11, 1980 (1980). https://doi.org/10.1007/3-540-10009-1_25
5. Jouannaud, J.P., Kirchner, C., Kirchner, H.: Incremental construction of unification algorithms in equational theories. In: Automata, Languages and Programming (1983). https://doi.org/10.1007/BFb0036921
6. Meier, S.: Advancing Automated Security Protocol Verification. Doctoral Thesis, ETH Zurich (2013). https://doi.org/10.3929/ethz-a-009790675
7. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: Computer Aided Verification (CAV) (2013). https://doi.org/10.1007/978-3-642-39799-8_48
8. Schmidt, B.: Formal Analysis of Key Exchange Protocols and Physical Protocols. Doctoral Thesis, ETH Zurich (2012). https://doi.org/10.3929/ethz-a-009898924
9. Schmidt, B., Meier, S., Cremers, C., Basin, D.: Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In: 25th Computer Security Foundations Symposium (CSF) (Jun 2012). https://doi.org/10.1109/CSF.2012.25