ETH zürich

ETH Library

Symbolic Analysis of Identity-Based Protocols

Conference Paper

Author(s):
Basin, David; Hirschi, Lucca; Sasse, Ralf

Publication date:
2019

Permanent link:
https://doi.org/10.3929/ethz-b-000344054

Rights / license:
In Copyright - Non-Commercial Use Permitted

Originally published in:
Lecture Notes in Computer Science 11565, https://doi.org/10.1007/978-3-030-19052-1_9

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Symbolic Analysis of Identity-Based Protocols *

David Basin[1], Lucca Hirschi[2], and Ralf Sasse[1]

[1] Department of Computer Science, ETH Zurich, {basin,ralf.sasse}@inf.ethz.ch
[2] Inria & LORIA, lucca.hirschi@inria.fr

Dedicated to Catherine Meadows on her 65th Birthday.

Abstract. We show how the Tamarin tool can be used to model and reason about security protocols using identity-based cryptography, including identity-based encryption and signatures. Although such protocols involve rather different primitives than conventional public-key cryptography, we illustrate how suitable abstractions and Tamarin's support for equational theories can be used to model and analyze realistic industry protocols, either finding flaws or gaining confidence in their security with respect to different classes of adversaries.
Technically, we propose two models of identity-based cryptography. First, we formalize an abstract model, based on simple equations, in which verification of realistic protocols is feasible. Second, we formalize a more precise model, leveraging Tamarin's support for bilinear pairing and exclusive-or. This model is much closer to practical realizations of identity-based cryptography, but deduction is substantially more complex. Along the way, we point out the limits of precise modeling and highlight challenges in providing support for equational reasoning. We also evaluate our models on an industrial protocol where we find and fix flaws.

1 Introduction

Context and Problem Statement. Networked information systems are deeply embedded in modern society. Communication, finance, energy distribution, transport, and even our social lives all critically depend on their correct and secure operation. In such domains, the use of cryptographic protocols is essential, but such protocols require predistributed secrets and this in turn necessitates key distribution and management. In general, security does not come for free: you need pre-established secrets to create new secrets, needed to protect communication.

For application domains where not all parties know each other and may not have pre-established relationships, a starting point is to use cryptographic protocols based on asymmetric cryptography to bootstrap security associations. This requires the use of public keys, which are authentically distributed using a public key infrastructure (PKI). This approach underpins the modern web. Website owners generate private/public key pairs and Certificate Authorities (CAs) sign certificates stating that the public key belongs to the entity controlling the

* This work was done while the second author was also at ETH Zurich.

website. The certificates are then used by web clients in security protocols like TLS to authenticate servers and setup the additional keys used to secure client-server communication. Although theoretically pleasing, this approach is prone to problems in practice, as amply documented in the literature.

An intriguing alternative is *identity-based cryptography* [25], where each participant is assigned a public key based on her public identity. The corresponding private key is generated by an entity running a *private key generator* (PKG). Identity-based cryptography therefore has a different setup, with different primitives and different assumptions than traditional public-key cryptography. For example, the PKG (not the participant) computes the private keys, which enables private key escrow, and the PKG must therefore be trusted not to reveal these keys.

Formal methods, in particular those based on a symbolic model of cryptography, have been shown to be effective for reasoning about cryptographic protocols built from standard primitives like symmetric and asymmetric encryption and the PKI setup explained before. Our focus in this paper is to explore their use in the context of identity-based cryptography. We present a case study on formalizing and analyzing identity-based cryptographic primitives and protocols based on them. Our emphasis is on how, with the right modeling language and deduction support, one can easily formalize such primitives and explore associated protocols. In this way, one can quickly identify design issues, clarify trust assumptions, and produce security proofs.

Contribution. We present the first symbolic models of identity-based cryptography. We present an abstract model that captures the basic functionality of identity-based signatures and encryption. This model is parametric and can be used for signatures, encryption, and their combination. Afterwards, we present a second, more precise, model based on a symbolic model of bilinear pairings (BP). Pairings are often used to realize identity-based cryptography and thus this model captures more of the cryptographic details found in implementations, which can be potentially exploited by an adversary.

We exemplify the use of both models with a simple protocol, and show the limitations of the precise model in terms of efficient deduction. Afterwards, we use the abstract model for a case study analyzing different versions of an industry-proposed protocol for identity-based Authentication and Key Agreement (AKA). The protocol combines key agreement based on bilinear pairing supporting key escrow, and identity-based signatures. Using Tamarin, we find numerous weaknesses. After fixing these weaknesses, we prove the security of the resulting protocol in our abstract model.

Related Work. We will discuss identity-based cryptography in detail in Section 2. Here we focus on symbolic analysis and provide some historical background.

The use of tools for the symbolic analysis of security protocols has a long history starting with Millen's Interrogator [22], the Longley–Rigby search tool [18], and Meadow's NRL Protocol Analyzer (NRLPA) [19]. These tools were based on a *symbolic model* of cryptography going back to the seminal work of Dolev and

Yao [13], which represents cryptographic operations within a term algebra that is amenable to efficient mechanized proof. The early tools constituted proto-model checkers. Indeed the NRLPA offered many features of a modern model checker, including an automated means of proving that exhaustive search of a finite state space implied exhaustive search of the infinite state space, and later, a temporal logic language, NPATRL [26], for describing protocol security properties. A good overview of symbolic security protocol analysis is available in an article in the Handbook of Model Checking by Basin, Cremers, and Meadows [5].

The current generation of tools includes the spiritual successor to the NRLPA tool, Maude-NPA [16], which follows the same basic ideas but is built on backwards narrowing within Maude [11]. Our tool of choice is Tamarin [23,20], which uses a combination of constraint solving, as introduced by Millen and Shmatikov [21], as well as backwards search as in Maude-NPA. We will provide further details in Section 3, in particular for Tamarin.

Outline. We introduce identity-based cryptography in Section 2. Afterwards we provide background on security protocol analysis in the symbolic model in Section 3. We present our symbolic model for identity-based cryptography in Section 4 and our case study in Section 5. We draw conclusions in Section 6.

2 Identity-Based Cryptography

Identity-based cryptography was proposed by Shamir [25] in the early 1980s. Realizations of an identity-based signature scheme quickly followed [17], whereas a realization of an encryption scheme took longer and was first proposed by Boneh and Franklin [9] in 2001. The basic idea, found in all realizations, has a feature that makes the use of identity-based cryptography less attractive than public-key cryptography in many settings: intrinsic key escrow.[3] In identity-based cryptography, the private key generation center stores a master secret that it uses to create the private keys for all participants. This is deeply embedded into identity-based cryptography although there have been proposals to split the master secret, using secret sharing schemes, so that there are multiple private key generators instead of one. However no such trust-reduction mechanisms have found wide-spread use. The public key of any participant can be generated using the public master key and the participant's name. Thus, only the public master key needs to be authentically distributed, not individual public keys.

We shall first describe abstract signature and encryption schemes, and then present some realizations. Our account here closely follows [2].

2.1 Signature Scheme

An Identity-based Signature (IBS) scheme relies on a trusted party, called the *private key generator* (PKG), for generating users' private signature keys. Users

[3] There are settings where key escrow may be desirable or even required, for example due to legal reasons. In such cases, identity-based cryptography fits perfectly.

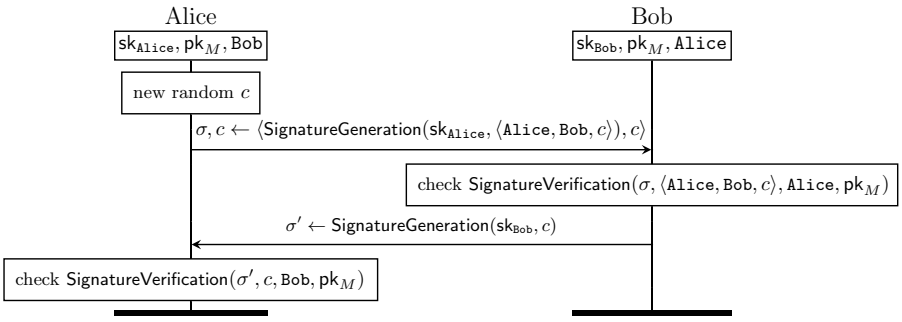

Fig. 1. Protocol flow of our running example.

are identified by their id, which can be their name, an email address, or other uniquely identifying information. The PKG is also in charge of assigning private keys, which are bound to the identities id of the appropriate users, while the master public key can be freely distributed. The PKG must therefore adopt appropriate authentication methods to verify that a user can claim a given identity id.

In terms of functionality, an IBS scheme should offer the following functions:

- $\mathsf{Setup}() \mapsto \mathsf{sk}_M, \mathsf{pk}_M$: The PKG creates its master private and public keys sk_M and pk_M, where pk_M can be made public to all users.
- $\mathsf{KeyRequest}(\mathsf{id}) \mapsto \mathsf{sk}_{\mathsf{id}}$: A user with the identity id requests a private signature key from the PKG. Upon authenticating the user, the PKG creates a signing key $\mathsf{sk}_{\mathsf{id}}$ associated to id and sends it to the user.
- $\mathsf{SignatureGeneration}(\mathsf{sk}_{\mathsf{id}}, m) \mapsto \sigma_{m,\mathsf{id}}$: A user with the signing key $\mathsf{sk}_{\mathsf{id}}$ produces a signature $\sigma_{m,\mathsf{id}}$ for the message m.
- $\mathsf{SignatureVerification}(\sigma, m, \mathsf{id}, \mathsf{pk}_M) \mapsto \mathsf{yes/no}$: Anyone possessing a signature σ supposedly signed by a user with the identity id for the message m can verify the signature using id (and pk_M).

Note that users can verify signatures without prior knowledge of any user-specific verification keys; only the master public key pk_M and the supposed signer's id are required.

Example 1 (Running example). We shall use a simple, signed, challenge response protocol as a running example throughout the paper. We illustrate how an IBS scheme can be used to implement such a protocol in Figure 1. In this protocol, Alice challenges Bob with a random nonce c and a signature on (Alice,Bob,c), and then expects to receive from Bob a valid signature on the nonce c.

2.2 Encryption Scheme

Just like IBS schemes, Identity-based Encryption (IBE) schemes rely on a PKG for generating private decryption keys. IBE schemes offer the following functions:

– $\mathsf{Setup}() \mapsto \mathsf{sk}_M, \mathsf{pk}_M$: The PKG creates its master private and public keys sk_M and pk_M, where pk_M can be made public to all users.
– $\mathsf{KeyRequest}(\mathsf{id}) \mapsto \mathsf{sk}_{\mathsf{id}}$: A user with identity id requests a private decryption key from the PKG. Upon authenticating the user, the PKG creates a decryption key $\mathsf{sk}_{\mathsf{id}}$ associated to id and sends it to the user.
– $\mathsf{Encryption}(\mathsf{id}, m, \mathsf{pk}_M) \mapsto \{m\}_{\mathsf{id}}$: Anyone possessing id (and pk_M) produces a ciphertext $\{m\}_{\mathsf{id}}$ that can be decrypted only with $\mathsf{sk}_{\mathsf{id}}$.
– $\mathsf{Decryption}(\{m\}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \mapsto m$: A user with the decryption key $\mathsf{sk}_{\mathsf{id}}$ decrypts a ciphertext $\{m\}_{\mathsf{id}}$ that has been encrypted with identity id (and pk_M).

Note that users can encrypt messages for other users without prior knowledge of any encryption keys; only pk_M and id are required. The intention is that decryption can only be performed by the user whose identity is id, as she is the only one to receive the associated private key. In contrast, conventional PKIs require users to request, and authentically receive, public keys of the intended recipients before encryption.

Example 2 (Continuing Example 1). One can use an IBE scheme to implement a challenge response mechanism as follows: Alice sends a random nonce encrypted with Bob's public key and then expects to receive the nonce c in plaintext.

2.3 Realization using Bilinear Pairing

Many recent IBE and IBS schemes are implemented using bilinear pairing [2]. We briefly recall bilinear pairing and explain its use in two identity-based schemes.

Bilinear Pairing. Let $\mathsf{e} : G \times G \mapsto G'$ be a symmetric bilinear mapping for G, an additive cyclic group[4] of prime order q, and G' a multiplicative cyclic group of order q. For e to be bilinear, it must satisfy the following algebraic property:

$$\forall a, b \in \mathbb{F}_{\mathsf{q}}^*. \forall Q_1, Q_2 \in G.\ \mathsf{e}(aQ_1, bQ_2) = \mathsf{e}(Q_1, Q_2)^{ab}. \tag{1}$$

It is often required that e is non-degenerate (*i.e.*, different from the constant $\mathbb{1}_{G'}$) and computable. In the following, we use P as the generator.

The cryptographic assumption under which BP cryptographic schemes can be proven secure is called the *Bilinear Diffie-Hellman (BDH)* assumption and is defined as follows:

> *BDH*: Given $(G, \mathsf{q}, \mathsf{e}, P, aP, bP, cP)$ where G, q, and e are as specified above and where $P \in G$ and a, b, c are chosen at random from $\mathbb{F}_{\mathsf{q}}^*$, it is infeasible to compute $\mathsf{e}(P, P)^{abc}$.

[4] In general, bilinear pairings can take values in two different groups, provided that they are of the same order. For simplicity and because our formal model will eventually require it, we only present bilinear pairing taking values in the same group.

Note that the BDH assumption is relevant for computational proofs, which manipulate bitstrings, probabilities, and probabilistic-polynomial time computations. Working in such a *computational model* typically necessitates manual proofs. In contrast we work in the *symbolic model* using a term representation and that is amenable to automated proofs. The guarantees one gets from the computational model are generally considered more fine-grained. However, given that they require time-consuming manual proof construction, computational proofs do not in practice cover entire protocols with all their modes and options. Compare, for example, the computational proofs for TLS 1.3 (e.g., [14]), which cover single modes only, to the symbolic proofs using Tamarin that cover the interaction of all modes and the protocol's entire internal state machine [12].

Since we work in the symbolic model, we are only interested in capturing algebraic properties of cryptographic primitives; *i.e.*, Equation 1 for bilinear pairing. As we later see, this equation can be directly modelled in Tamarin.

BP-based IBE Scheme by Boneh and Franklin [9]. Let the following be given: a plaintext length $l \in \mathbb{N}$, a set of identities $\mathsf{identities}$, and two hash functions $h_1 : \mathsf{identities} \mapsto G^*$ and $h_2 : G' \mapsto \{0,1\}^l$. Using the pairing function e, the IBE functionalities are then implemented as follows:

- $\mathsf{Setup}() \mapsto \mathsf{sk}_M, \mathsf{pk}_M$: The PKG picks s at random in $\mathbb{F}_\mathsf{q}^*$ and computes $\mathsf{sk}_M := s$ and $\mathsf{pk}_M := sP$, where P is a generator in G.
- $\mathsf{KeyRequest}(\mathsf{id}) \mapsto \mathsf{sk}_\mathsf{id}$: Upon reception of id, the PKG creates a private decryption key $\mathsf{sk}_\mathsf{id} := \mathsf{sk}_M Q_\mathsf{id}$, where $Q_\mathsf{id} := h_1(\mathsf{id})$.
- $\mathsf{Encryption}(\mathsf{id}, \mathsf{pk}_M, m) \mapsto \{m\}_\mathsf{id}$: To encrypt a message $m \in \{0,1\}^l$ using id and pk_M, a user picks r at random from $\mathbb{F}_\mathsf{q}^*$, computes $U = rP$ and $V = h_2(\mathsf{e}(Q_\mathsf{id}, \mathsf{pk}_M)^r) \oplus m$, where $\oplus$ denotes the eXclusive-OR (XOR) operator. The ciphertext is $\{m\}_\mathsf{id} := (U, V)$.
- $\mathsf{Decryption}(\{m\}_\mathsf{id}, \mathsf{sk}_\mathsf{id}) \mapsto m$: A user possessing $\mathsf{sk}_\mathsf{id} = \mathsf{sk}_M Q_\mathsf{id}$ can decrypt $\{m\}_\mathsf{id} := (U, V)$ by computing $V \oplus h_2(\mathsf{e}(\mathsf{sk}_\mathsf{id}, U)) = (h_2(\mathsf{e}(Q_\mathsf{id}, \mathsf{sk}_M P)^r) \oplus m) \oplus h_2(\mathsf{e}(\mathsf{sk}_M Q_\mathsf{id}, rP)) = m$.

We show in Section 4.2 how this scheme can be modelled in Tamarin.

BP-based IBS scheme by Cha and Cheon [10]. This IBS scheme uses the hash function h_1 described above and an additional hash function $h_3 : \{0,1\}^* \times G \mapsto \mathbb{F}_\mathsf{q}^*$. The IBS functionalities are then implemented as follows, also using e:

- $\mathsf{Setup}() \mapsto \mathsf{sk}_M, \mathsf{pk}_M$: The PKG picks s at random in $\mathbb{F}_\mathsf{q}^*$ and computes $\mathsf{sk}_M := s$ and $\mathsf{pk}_M := sP$, where P is a generator in G.
- $\mathsf{KeyRequest}(\mathsf{id}) \mapsto \mathsf{sk}_\mathsf{id}$: Upon reception of id, the PKG creates a private signing key $\mathsf{sk}_\mathsf{id} := \mathsf{sk}_M Q_\mathsf{id}$, where $Q_\mathsf{id} := h_1(\mathsf{id})$.
- $\mathsf{SignatureGeneration}(\mathsf{sk}_\mathsf{id}, m) \mapsto \sigma_{m,\mathsf{id}}$: A user with the signing key sk_id can sign a message $m \in \{0,1\}^*$ by picking an $r \in \mathbb{F}_\mathsf{q}^*$ at random and computing: $U = rQ_\mathsf{id}$, $V = (r + h_3(m, U))\mathsf{sk}_\mathsf{id}$, and, $\sigma_{m,\mathsf{id}} = (U, V)$, where $+$ is addition in $\mathbb{F}_\mathsf{q}^*$.

– $\mathsf{SignatureVerification}(\sigma_{m,\mathsf{id}}, m, \mathsf{id}, \mathsf{pk}_M) \mapsto \mathsf{yes/no}$: Anyone having the signature $\sigma_{m,\mathsf{id}} := (U, V)$ supposedly signed by the user with the identity id can verify the signature using pk_M and id by checking whether $\mathsf{e}(P, V) \stackrel{?}{=} \mathsf{e}(\mathsf{pk}_M, U + h_3(m, U) \cdot Q_{\mathsf{id}})$. Indeed if $\sigma_{m,\mathsf{id}}$ is a genuine signature, then:

$$\begin{aligned}\mathsf{e}(P, V) &= \mathsf{e}(P, (r + h_3(m, U))\mathsf{sk}_{\mathsf{id}}) \\ &= \mathsf{e}(P, \mathsf{sk}_M \cdot (r + h_3(m, U)) \cdot Q_{\mathsf{id}}) \\ &= \mathsf{e}(\mathsf{pk}_M, r \cdot Q_{\mathsf{id}} + h_3(m, U) \cdot Q_{\mathsf{id}}) \\ &= \mathsf{e}(\mathsf{pk}_M, U + h_3(m, U) \cdot Q_{\mathsf{id}}).\end{aligned}$$

We shall see in Section 4.2 that the algebraic properties involved in this scheme are too complex to automate reasoning about using state-of-the art verification tools. This is the main reason why we also explore in Section 4.1 a less precise abstraction of IBS schemes that we can more efficiently reason about.

We summarize next the symbolic model of protocol analysis in Section 3, before applying it to identity-based cryptography in Section 4.

3 Symbolic Analysis of Protocols

In this section, we briefly introduce the symbolic model for security protocols and the tool Tamarin which automates reasoning in this model. We also describe how security properties are modeled using Tamarin.

3.1 The Tamarin Prover

Tamarin is a state-of-the-art protocol verification tool that automates reasoning in the *symbolic model* of cryptographic protocols. Tamarin supports stateful protocols specified using a large collection of equationally defined operators, including bilinear pairing. It has previously been applied to numerous real-world protocols with complex state machines, numerous messages, and complex security properties such as TLS 1.3 [12] and 5G-AKA [6].

In the symbolic model, messages are described as terms. For example, $enc(m, k)$ represents the message m encrypted using the key k. The algebraic properties of the cryptographic functions are then specified using equations over terms. For example, the equation $dec(enc(m, k), k) = m$ specifies the expected property of symmetric encryption: decryption with the encryption key k yields the plaintext m. As is common in the symbolic model, cryptographic messages do not satisfy other properties than those specified by explicit algebraic properties. This reflects the so-called *black box cryptography assumption*: one cannot exploit potential weaknesses in the cryptographic primitives themselves. Tamarin also supports further algebraic properties, including hashing, XOR, Diffie-Hellman, and bilinear pairing.

In Tamarin, a protocol is described using multiset rewrite rules. These rules manipulate multisets of *facts*, with *terms* as arguments, which model the system's state.

Example 3. The following rules describe a simple protocol that sends a MACed message. The first rule creates a new long-term shared key k (the fact $!\mathsf{Ltk}$ is *persistent*: it can be used as a premise multiple times). The second rule describes the agent A who sends a fresh message m together with its MAC with the shared key k to B. Finally, the third rule describes B, who is expecting as input a message and a corresponding MAC with k. Note that the third rule can only be triggered if the input matches the premise, *i.e.*, if the input message is correctly MACed with k.

$$\begin{aligned}
&\mathit{Create_Ltk}: \ [\mathsf{Fr}(k)] -\!\![\ ]\!\!\rightarrow [!\mathsf{Ltk}(k)],\\
&\mathit{Send_A}: \ [!\mathsf{Ltk}(k), \mathsf{Fr}(m)] -\!\![\ \mathsf{Sent}(m)\ ]\!\!\rightarrow [\mathsf{Out}(\langle m, \mathit{mac}(m,k)\rangle)],\\
&\mathit{Receive_B}: \ [!\mathsf{Ltk}(k), \mathsf{In}(\langle m, \mathit{mac}(m,k)\rangle)] -\!\![\ \mathsf{Received}(m)\ ]\!\!\rightarrow [\ ]
\end{aligned}$$

These rules (written $[l] -\!\![\ a\ ]\!\!\rightarrow [r]$ with a the actions) yield a labeled transition system describing the possible protocol executions (see [1,23] for details on Tamarin's syntax and semantics), where the traces are sequences of the action labels. Tamarin combines the rules formalizing the protocol with rules formalizing a Dolev-Yao [13] style adversary. This adversary controls the entire network and can thereby intercept, delete, modify, delay, inject, and build new messages. However, the adversary is limited by the cryptography: he cannot forge signatures or decrypt messages without knowing the key due to the black box cryptography assumption. He can nevertheless apply any function, e.g., hashing, XOR, encryption, pairing, etc., to messages he knows and thus compute new messages.

3.2 Formalizing Security Properties in Tamarin

In Tamarin, security properties are specified in two ways. First, trace properties, such as secrecy or variants of authentication, are specified using formulas in a first-order logic with timepoints. Second, equivalence properties, e.g., for unlinkability, can be given as *diff-terms* [7]; these are not considered in this paper.

Example 4. Consider the multiset rewrite rules given in Example 3. The following property specifies a form of non-injective agreement on the message m, *i.e.*, that any message received by B was previously sent by A:

$$\forall i, m. \mathit{Received}(m)@i \Rightarrow (\exists j. \mathit{Sent}(m)@j \wedge j \lessdot i).$$

For each specified property, Tamarin checks that the property holds for all possible protocol executions and all possible adversary behaviors. To achieve this, Tamarin explores all possible executions in a backward manner, searching for reachable attack states, which are counterexamples to the security properties.

In fully automatic mode, Tamarin either returns a proof that the property holds, or a counterexample representing an attack if the property is violated. It may also fail to terminate, which is unsurprising given that the underlying problem is undecidable. Tamarin can also be used in an interactive mode where the user can guide the proof search. Moreover, the user can supply heuristics called *oracles* to guide the proof search in a sound way.

```
functions: IBPriv/2, IBPub/2, IBMasterPubK/1,                          1
           idsign/2, idverify/3, true/0                                2
equations: idverify(idsign(m,IBPriv(A, IBMasterPrivK)),                3
                   m,                                                  4
                   IBPub(A, IBMasterPubK(IBMasterPrivK))) = true       5
```

Fig. 2. Function symbols and equation declaration for IBS. Undeclared identifiers represent variables.

4 Modeling Identity-Based Cryptography

We now show how identity-based cryptography can be modeled symbolically and how protocols that use the associated cryptographic primitives can be analyzed.

We first present an abstract model in Section 4.1 that captures, at a high-level, the intended algebraic properties of identity-based schemes as described in Sections 2.1 and 2.2. Although abstract, this model suffices to identify nontrivial logical attacks, as illustrated by our case study in Section 5.

In practice, many identity-based schemes are based on bilinear pairings and thus have additional algebraic properties (see Section 2.3) that can be exploited by the adversary. We therefore provide a second, more concrete model in Section 4.2, inspired by the state-of-the-art, bilinear pairing-based, identity-based schemes described in Section 2.3. Unsurprisingly, protocols formalized in this second model are harder to reason about. As we shall see, we reach Tamarin's limits, which suggests that our abstract model is a good compromise.

Note that for each of our two identity-based cryptography models, we provide a Tamarin model [8] that illustrates its use on our running example (Example 1 for IBS and Example 2 for IBE).

4.1 Abstract Model of Identity-Based Schemes

We now present a parametric model of Identity-Based Schemes, formalized in Tamarin. Most of this model is parametric in the type of scheme (IBE or IBS). We first describe our model of IBS and afterwards explain the main differences with our model of IBE.

Modeling IBS schemes. Our abstract model of the *private key generator*'s (PKG) capabilities consists of a user-defined equational theory together with setup and initialization rules that generate the identity-based signing private master key and sets up users with their private keys.

The equational theory of Sign-PKG with its signature and equation is depicted in Figure 2. Here, `idsign` models identity-based signing, `idverify` models signature verification, and `IBMasterPubK` is the operator deriving the master public key from the master private key. The signature also includes the function `IBPriv` used by the PKG to provision a user with a private key (as this

```
// Create the trusted entity holding the id-based master private key
rule create_IB_PrivateKeyGenerator:    // Setup()
  [ Fr(~IBMasterPrivK) ]
-->
  [ !IB_MasterPrivateKey('PKG', ~IBMasterPrivK)
  , Out(<'PKG', IBMasterPubK(~IBMasterPrivK)>) ] //adversary gets pkM
// Setup rules for identities
rule create_IB_identity:               // KeyRequest($A)
  let Master_pk = IBMasterPubK(IBMasterPrivK)
      User_sk = IBPriv($A, IBMasterPrivK) in
  [ !IB_MasterPrivateKey('PKG', IBMasterPrivK)
  , Fr(~id) ]
--[ CreateId($A, <Master_pk, User_sk>), User() ]->
  [ !IB_Identity(~id, $A, Master_pk, User_sk) ]
```

Fig. 3. Modeling PKG. Variables prefixed with ~are fresh.

provisioning requires the private master key) and the function `IBPub` that derives public keys from identities. The single equation guarantees that a properly created signature can be successfully validated by anyone holding the signer's name and the PKG's public signature verification key, while no other signature is accepted for this message `m` by this participant `A`.

In Figure 3, we depict the two rules that comprise our theory. The setup rule `create_IB_PrivateKeyGenerator` generates the signing master key and the associated public key, which is constructed by applying `IBMasterPubK` to the secret. The public key is published by sending it out on the network while the private key is, of course, stored by the PKG. The rule `create_IB_identity` models the setup of a new participant with identity `$A`[5] (KeyRequest(`$A`)), initialized with (i) the master public key `Master_pk` that will be used to compute signature verification keys, and (ii) the user's private signing key, which is provided by the PKG. The user's private key is derived from the signing master key and the user's identity, `IBPriv($A, IBMasterPrivK)`.

Finally, we model the adversary's compromise capabilities in Figure 4. We model a strong adversary who can completely compromise the system by revealing the PKG's master private key (rule `Reveal_IB_MasterPrivateKey`) as well as compromising individual agents by revealing their private signing key (rule `Reveal_IB_privkey`). Security properties would then typically be conditioned by the absence of specific compromises.

We build on this representation of identity-based signatures to model different versions of a protocol that combines bilinear pairing-based authentication and key agreement, with key escrow and identity-based signatures (see Section 5). We also develop a complete Tamarin model for Example 1 (see [8]) including this representation of IBS; we depict the rule corresponding to Alice's first action in Figure 5.

[5] `$A` denotes a variable that can be instantiated by any public constant.

```
// Reveals the id-based master private key of the PKG
rule Reveal_IB_MasterPrivateKey:
  [ !IB_MasterPrivateKey(PKG, IBMasterPrivK) ]
--[ Reveal('PKG',PKG) ]->
  [ Out(IBMasterPrivK) ]
// Reveals the id-based private key of an agent A
rule Reveal_IB_privkey:
  [ !IB_Identity(~id, A, Master_pk, User_sk) ]
--[ Reveal('USER',A) ]->
  [ Out(User_sk) ]
```

Fig. 4. Adversary compromise rules.

```
rule Alice_send:
let m = <'Alice', 'Bob', ~c>
    mOut = <idsign(m, User_sk),~c> in   // SignatureGeneration(...)
  [ !IB_Identity(~id, 'Alice', Master_pk, User_sk)
  , Fr(~) ]
--[ Running('Alice', 'Bob', <'Initiator', 'Responder', ~c>) ]->
  [ Out(mOut)
  , St_Alice_0(~id, Master_pk, User_sk, ~c) ]
```

Fig. 5. Example usage of our abstract IBS model for our running example.

Modeling IBE schemes. Our IBS representation can be simply modified to model IBE schemes by making minor changes to the function symbols and equations as shown in Figure 6.

```
functions: IBPriv/2, IBPub/2, GetIBMasterPublicKey/1,
           idenc/2, iddec/2
equations: iddec(idenc(plaintext,
                       IBPub(A,
                       GetIBMasterPublicKey(IBMasterPrivateKey))),
                 IBPriv(A, IBMasterPrivateKey)) = plaintext
```

Fig. 6. Function symbols and equation declaration for IBE.

The rules for modeling the PKG and the reveals are identical. A complete Tamarin model of Example 2 is given in [8].

4.2 More Precise Modeling of Bilinear Pairing-based ID-based Schemes.

By leveraging TAMARIN's built-in theory for bilinear pairing, it is possible to model concrete IBE or IBS schemes much more precisely than in Section 4.1. Our next theory features four function symbols: `pmult`, `em`, $\hat{}$, and, $*$. The function symbol `pmult` is of arity 2 and models the multiplication of a group element (*e.g.*, $P \in G$) by a scalar (*e.g.*, $s \in \mathbb{F}_q^*$), `em` models the bilinear pairing `e` and is modulo AC, $\hat{}$ models the exponentiation of group elements (*e.g.*, $g \in G'$) by a scalar, and $*$ of arity 2 models the multiplication between scalars.

The function symbols $\hat{}$ and $*$ are subject to the equations of the built-in Diffie-Hellman theory, while `pmult` and `em` are subject to the equational theory for bilinear pairing, also built-in [24], which is summarized as:

```
pmult(x,(pmult(y,p)) = pmult(x*y,p)
pmult(1,p)           = p
em(p,q)              = em(q,p)
em(pmult(x,p),q)     = pmult(x,em(q,p)).
```

Modeling IBE Scheme Using Bilinear Pairing. One can leverage the built-in theories for bilinear pairing and XOR, recently added to TAMARIN [15], which can be combined, to model the Boneh and Franklin IBE scheme.

Using this more precise representation, we have developed a full model of Example 2 (see [8]). Our model introduces two function symbols h_1 and h_2 for modeling the two independent hash functions. The setup rules are now a bit different as the master private key and the user's private keys are computed with `pmult`, as depicted in Figure 7. We also show in Figure 8 an example of a protocol rule using encryption, which corresponds to Alice's first output in our example.

Our model of Example 2 loads in ca. 1 hour of (pre-computation) CPU time. However, due to the heavy branching required to explore all possible variants, proofs or counterexamples cannot be automatically computed in reasonable time. However, when simplifying the protocol by removing Bob's response (hence with only one encryption and decryption), the pre-computation and the proof of secrecy of the plaintext could be computed in minutes, with both BP and XOR.

We thus conclude that, thanks to recent advances in the scope of the equational theories that TAMARIN handles, TAMARIN supports such a precise model of IBE schemes. However, the real limitation is now the efficiency of the proof search, which is negatively impacted by the numerous variants introduced by the combination of the built-in theories for bilinear pairing and XOR. Therefore, we do not use this precise model for our case study in Section 5.

Modeling IBS Scheme using Bilinear Pairing. The Cha and Cheon IBS scheme described in Section 2.3 relies on $\mathbb{F}_q^*$ being a field, and *a fortiori*, a ring with an associative and commutative $+$, where $\cdot$ distributes over $+$. Unfortunately, TAMARIN, and all other symbolic tools, are currently unable to deal with such equational theories.

```
// Create the trusted entity holding the master private key
rule create_IB_PrivateKeyGenerator:     // Setup()
  let Master_pk = pmult(~IBMasterPrivateKey, 'P') in // pkM
  [ Fr(~IBMasterPrivateKey) ]
--[ Once('PKG') ]->
  [ !IB_MasterPrivateKey('PKG', ~IBMasterPrivateKey)
  , Out(<'PKG', Master_pk>) ]           // adversary gets pkM
// Setup rules for identities
rule create_IB_identity:                // KeyRequest($A)
 let Master_pk = pmult(~IBMasterPrivateKey, 'P')
     Qid = h1($A)
     User_sk = pmult(~IBMasterPrivateKey, Qid) in  // 'skM.Qid'
  [ !IB_MasterPrivateKey('PKG', ~IBMasterPrivateKey)
  , Fr(~id) ]
--[ CreateId($A, <Master_pk, User_sk>) ]->
  [ !IB_Identity(~id, $A, Master_pk, User_sk) ]
```

Fig. 7. PKG model for our precise IBE scheme.

At first sight, the random scalar r in $V = (r + h_3(m, U))\mathsf{sk}_{\mathsf{id}}$ seems to be only useful for randomizing the signature $\sigma = \langle rQ_{\mathsf{id}}, V\rangle$. Therefore, we modeled a simplified scheme that does not rely on $+$ over $\mathbb{F}_q^*$ for which $\sigma = \langle Q_{\mathsf{id}}, h_3(m, U))\mathsf{sk}_{\mathsf{id}}\rangle$ and SignatureVerification() is modified straightforwardly. Interestingly, when using this model, Tamarin automatically finds an attack on a supposedly secure protocol (namely, our fixed variant of the case study we describe in Section 5). After inspection, we found that this attack actually reveals a flaw in our simplified scheme where we omit r. Indeed, an adversary obtaining a signature $\sigma = \langle Q_{\mathsf{id}}, h_3(m, U)\mathsf{sk}_{\mathsf{id}}\rangle =: \langle U, V\rangle$ over a message m can forge a new signature over any other message m' as follows: $\sigma' := \langle U, h_3(m', U) \cdot (h_3(m, U))^{-1} \cdot V\rangle = \langle Q_{\mathsf{id}}, h_3(m', U)\mathsf{sk}_{\mathsf{id}}\rangle$. We have not found another simplification of the original scheme that would still be secure in the symbolic model.

All IBS realizations in the literature rely on the same kind of problematic equational theories. Hence, to the best of our knowledge, it is currently out of the scope of existing verification tools to reason about such a BP-based model of IBS schemes. This limitation could be tackled by developing new built-in theories with dedicated automated reasoning algorithms. We leave this task for future work.

5 Case Study

As a case study, we formalize and analyze an identity-based *Authentication and Key Agreement* (AKA) protocol, provided by an industry partner. The key agreement is based on identity-based signing and on bilinear pairing (see [1,24]) for key derivation, which also supports key escrow. We refer to this protocol as *BP-IBS* and depict it in Figure 9.

```
rule Alice_send:
let plaintext = <'Alice', 'Bob', ~c>
    Qbob = h1('Bob')
    U = pmult(~r, 'P')
    V = h2(em(Qbob, Master_pk)^(~r)) XOR plaintext
    mOut = <U,V> in     // Encryption(...)
  [ !IB_Identity(~id, 'Alice', Master_pk, User_sk)
  , Fr(~c)
  , Fr(~r) ]
-->
  [ Out(mOut)
  , St_Alice_0(~id, Master_pk, User_sk, ~c) ]
```

Fig. 8. Example usage of our precise IBE model for Example 2.

BP-IBS is a 2-party AKA protocol that relies on a trusted PKG. It aims to achieve the following properties:

1. *mutual authentication* and *agreement* on the session key;
2. *session key escrow*: the master secret held by the PKG can be combined with a session transcript to compute the associated session key;
3. *weak forward secrecy*: session keys remain secret even after the long-term keys of one of the two involved agents are revealed; and
4. *strong forward secrecy*: session keys remain secret even after the long-term keys of both involved agents are revealed.

Note that the properties of weak and strong forward secrecy are necessarily violated once one of the authentication properties is violated. Note too that strong forward secrecy implies weak forward secrecy, which itself implies secrecy.

5.1 Specification

Starting from the informal presentation provided by our industrial partner, we developed a more formal specification, which we describe in this section.

Setting. BP-IBS relies on two infrastructures that can be provided by one or two separate PKGs. For generality, we describe the protocol where these two infrastructures are provided by two distinct PKGs, called respectively Sign-PKG and Auth-PKG.

– *Sign-PKG.* A *PKI infrastructure* that provides identity-based signatures. It provides, for any user of identity ID_i, a signing key sk_i. Any other user can verify that a message has been signed by a user ID_i using the master public key of Sign-PKG and ID_i. We model this PKG using our abstract model from Section 4.1.

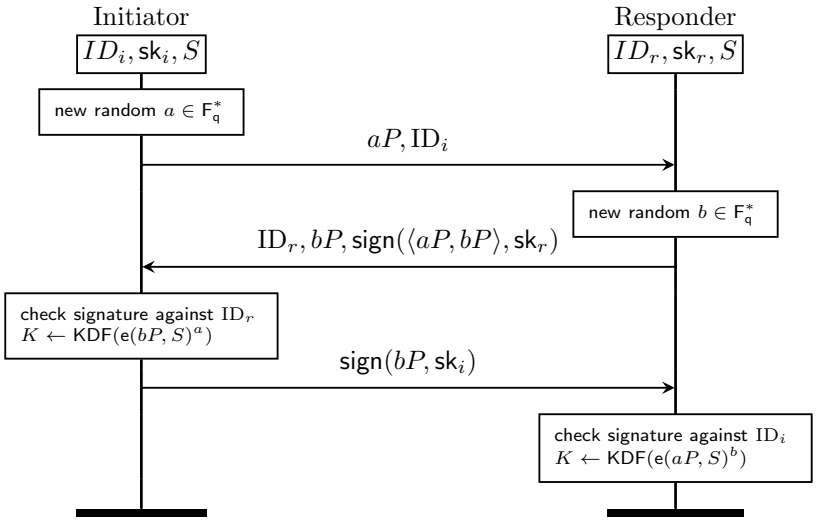

Fig. 9. Alice & Bob Specification of BP-IBS.

– *Auth-PKG.* A *key escrow* setup that is used in the protocol to allow the corresponding PKG to learn session keys, *i.e.*, *session escrow*. This PKG initially generates a master private key $s \in \mathsf{F}_q^*$ and publicly discloses to its users the corresponding master public escrow share $S = sP$. We model this PKG using our precise model based on bilinear pairing (see Section 4.2) without including user encryption and decryption. Indeed, we do not need this PKG to provide users with id-based encryption keys.

Note that we could have modeled a single PKG providing both requirements. Our presentation, however, supports a more fine-grained analysis, *e.g.*, in terms of compromise scenarios.

BP-IBS. The protocol flow is depicted in Figure 9. When a condition fails to hold, the corresponding agent aborts the protocol. Here, KDF models a key derivation function, abstracting away any implementation choices. The protocol shown is a 2-party authenticated key exchange protocol that uses identity-based signatures to authenticate key shares and allows session key escrow. Note that the session key established by the initiator and the responder satisfies:

$$K = \mathsf{KDF}(\mathsf{e}(bP, S)^a) = \mathsf{KDF}(\mathsf{e}(aP, S)^b) = \mathsf{KDF}(\mathsf{e}(P, P)^{abs}) = \mathsf{KDF}(\mathsf{e}(aP, bP)^s).$$

The last term in this equality chain shows that the protocol provides Auth-PKG with the ability to escrow all sessions.

5.2 Modeling BP-IBS

Figure 10 shows the setup of the key generation center, with key escrow functionality. In the rule `create_IB_AUTH_PrivateKeyGenerator`, the PKG's pri-

```
// Create the trusted entity holding the master private key for authentication escrow
rule create_IB_AUTH_PrivateKeyGenerator:
  let pk_master_secret = pmult(~IBMasterPrivK, 'P') in
  [ Fr(~IBMasterPrivK) ]
--[ Once('AUTH') ]->
  [ !IB_MasterPrivateKey('AUTH', ~IBMasterPrivK)
  , !IB_MasterPublicEscrowShare(pk_master_secret)
  , Out(<'AUTH', pk_master_secret>) ] // adversary gets pkM
```

Fig. 10. Modeling the PKG for Authentication Escrow.

vate key is generated randomly as `~IBMasterPrivK`. The associated public key, `pmult(~IBMasterPrivK, 'P')`, is constructed by multiplying the generator `'P'` with the secret. It is output to the adversary and stored and made available to all agents in `!IB_MasterPublicEscrowShare(...)`.

To initialize an agent for a run, we combine the look-up of both its private signing key and the key escrow share. The private signing key look-up works as described in the previous section, using the fact `!IB_Identity(...)`. Access to the key escrow share `x` is available in the argument of the `!IB_MasterPublicEscrowShare(x)` fact. Using the key escrow share, as designated by the protocol, allows the key escrow holder subsequent access to all agreed-upon keys between all users, without being involved in the key exchange.

Modeling the Core Protocol. Based on the aforementioned infrastructures, the core protocol of BP-IBS can be straightforwardly modeled in Tamarin. We provide the full model in [8] (file `BP-IBS_0.spthy`).

Easing Reasoning with an Oracle. Tamarin's default proof strategy is non-terminating, due to our model's complexity. We therefore implemented a dedicated oracle (see [1]). Oracles are lightweight tactics that can be used to guide proof search in Tamarin. Our oracle is available at [8] (file `oracle_BP-IBS`).

5.3 Security Properties

We now describe the different security properties BP-IBS should meet, as stated earlier in Section 5, which we verify in Tamarin.

Threat Model. We model a setting where the communication channel between the initiator and responder is assumed to be insecure. We formalize a standard network adversary (the *Dolev-Yao adversary*), which is an active adversary that can eavesdrop on and tamper with all exchanged messages

As is standard, we assume that the two PKGs, Sign-PKG and Auth-PKG, are honest. However, their long term secrets, including their master private keys, may be revealed to the adversary. Similarly, the long term secrets sk_i and sk_r

may also be revealed. We therefore symbolically model capabilities analogous to those assumed in computational models where the adversary is equipped with "reveal queries" [3,4]; we establish our security properties in the presence of an adversary who can carry out such queries.

Authentication Properties. For each role (*i.e.*, initiator, responder), we analyze four types of authentication properties: aliveness, weak agreement, and non-injective agreement on the session key, and injective agreement on the session key. These authentication properties are only checked when there is no long-term signing key reveal (*i.e.*, $\mathsf{sk}_i, \mathsf{sk}_r$) and no reveal of the master private key of Sign-PKG[6].

Secrecy Properties. We first check *weak secrecy*: session keys remain secret when there is no reveal at all. *Weak Forward Secrecy* states a stronger property: the session key established by ID_i and ID_r remains secret even when either the key sk_i or the key sk_r is revealed *after* the session. *Strong Forward Secrecy* is even stronger: the session key established by ID_i and ID_r remains secret even when both keys sk_i or sk_r are revealed *after* the session.

Session Escrow. This property simply checks that there must be an execution where the adversary learns the session key between some ID_i and ID_r, when he knows the master private key of Auth-PKG but without further reveals. This represents that the escrow key holder is able to derive session keys as expected.

5.4 Analysis Results

We have automatically analyzed the aforementioned security properties using Tamarin with our oracle; this analysis is supported by our model of BP-IBS, which is amenable to automation. Our analyses revealed several attacks in the original protocol. We have proposed protocol improvements that incorporate countermeasures, and have automatically verified that the improved protocols have the desired security properties. We summarize our results in Table 1.

We stress that all attacks were found automatically with Tamarin, as well as the proofs of the correctness of our improved versions with countermeasures.

Attack 1: Empty shares. This attack violates weak agreement for the initiator as well as weak secrecy. We first discovered that, in the original protocol (modeled in the file `BP-IBS_0.spthy` from [8]), the initiator and responder do not check that the received shares are different from P, which is the generator. In combination with other authentication weaknesses that we describe next, the adversary can trick a responder to accept the share $aP = P$ as coming from an honest initiator. The adversary can then compute the session key without any further key reveal.

[6] We also checked that all authentication properties fail in the presence of signing key reveals but when the master private key of AUTH-PKG is not revealed. This result is as expected since one cannot then rely on signatures to authenticate agents.

Protocol	Aliveness (I/R)	Weak Agr. (I/R)	NI-Agr. (I/R)	I-Agr. (I/R)
BP-IBS (0)	✘$_1$/✓	✘$_1$/✘$_1$	✘$_1$/✘$_1$	✘$_1$/✘$_1$
BP-IBS + check P (1)	✘$_2$/✓	✘$_2$/✘$_2$	✘$_2$/✘$_2$	✘$_2$/✘$_2$
+ tags (2)	✓/✓	✘$_3$/✘$_3$	✘$_3$/✘$_3$	✘$_3$/✘$_3$
+ ID$_r$ (3)	✓/✓	✓/✘$_4$	✓/✘$_4$	✓/✘$_4$
+ ID$_i$ (4)	✓/✓	✓/✓	✓/✓	✓/✓

Protocol	Weak Secrecy	Weak FS	Strong FS	Session Escrow
BP-IBS (0)	✘$_1$†	✘$_1$	✘$_1$	✓
BP-IBS + check P (1)	✘$_2$†	✘$_2$	✘$_2$	✓
+ tags (2)	✓	✘$_3$	✘$_3$	✓
+ ID$_r$ (3)	✓	✘$_4$	✘$_4$	✓
+ ID$_i$ (4)	✓	✓	✓	✓

Table 1. Analysis outcomes. "NI-Agr." stands for "Non-Injective-Agreement", "I-Agr." for "Injective-Agreement" and "FS" for forward secrecy. "(I/R)" means that we first describe the authentication property from the initiator's and then from the responder's point of view. When there is an attack, we indicate its type (see Section 5.4) with a subscript. Results labeled † were not directly obtained with Tamarin, as the attack found on weak agreement for the respective protocol version immediately translates to an attack on secrecy.

From the regular participant's view, the session key is $K = \mathsf{KDF}(\mathsf{e}(P, S)^b)$, and note that the other share uses $a = 1$, so $aP = P$. This key can be computed by the adversary since $K = \mathsf{KDF}(\mathsf{e}(bP, S))$ and he knows the master public key S of Auth-PKG as well as the responder's share bP that is sent in the clear.

Suggested countermeasure. We suggest that the initiator and responder should check that the received shares are different from P. We call the resulting protocol BP-IBS (1), which is modeled in `BP-IBS_1.spthy` in [8].

Attack 2: Reflection attack. Other properties of the protocol BP-IBS (1) are still violated. We have found that a reflection attack violates aliveness for the initiator. This attack is caused by a possible confusion between the signed message sent by the responder and the signed message sent by the initiator. As a direct consequence, one can use the signature produced by an initiator that has received a dishonest share (essentially a type-flaw) to forge a message that is accepted by another initiator as coming from a legitimate responder. One can use this weakness to mount attacks against the aliveness property for the initiator (meaning the supposed partner of the initiator did not take part in the protocol) as well as secrecy of session keys and weak agreement properties for both sides. We show in Figure 11 the attack flow violating aliveness for the initiator.

Suggested countermeasure. We suggest adding tags (i.e., `R` for the responder and `I` for the initiator) in the signed message expressing message origin. We call the resulting protocol BP-IBS (2), which is modeled in the file `BP-IBS_2.spthy` in [8]. We proved that this model ensures aliveness for the initiator and weak secrecy of session keys, hence our fix address the aforementioned attack.

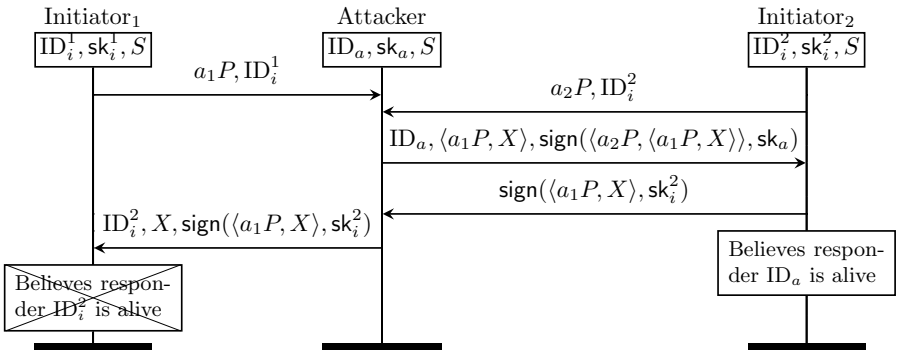

Fig. 11. Attack 2 on BP-IBS (1). Note that X can be chosen by the adversary.

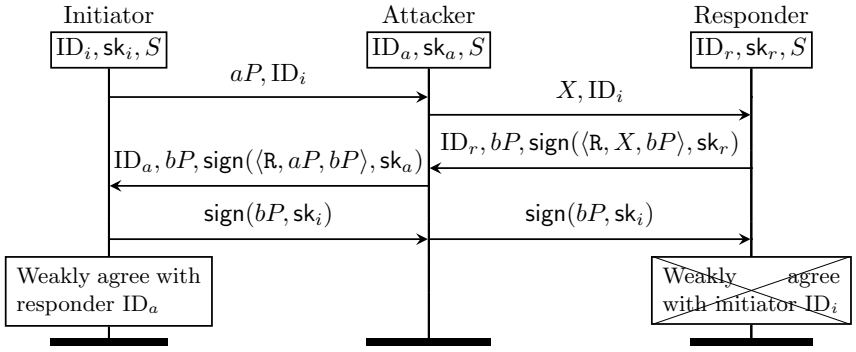

Fig. 12. Attack 3 on BP-IBS (2). The responder ID_r believes it has established a session with ID_i as initiator. But this view does not match with ID_i's view.

Attacks 3 and 4: Lack of identity-binding. The protocol BP-IBS (2) does not provide weak agreement for either of the two roles. This is because neither role includes the other role's identity in the signed part of their messages. One of these attacks is shown in Figure 12. There is a similar attack that violates weak agreement for the responder.

Suggested countermeasure. We suggest adding the responder's (respectively initiator's) identity in the message signed by the initiator (respectively responder). We call BP-IBS (3) the protocol one obtains from BP-IBS (2) by adding the responder's identity and BP-IBS (4) the protocol where both identities are added. These protocols are respectively modeled in the files `BP-IBS_3.spthy` and `BP-IBS_4.spthy` in [8]. We depict BP-IBS (4) in Figure 13.

5.5 Summary

Due to numerous problems, the original protocol does not meet its security requirements. Fortunately, the discovered attacks are easy to repair as shown by

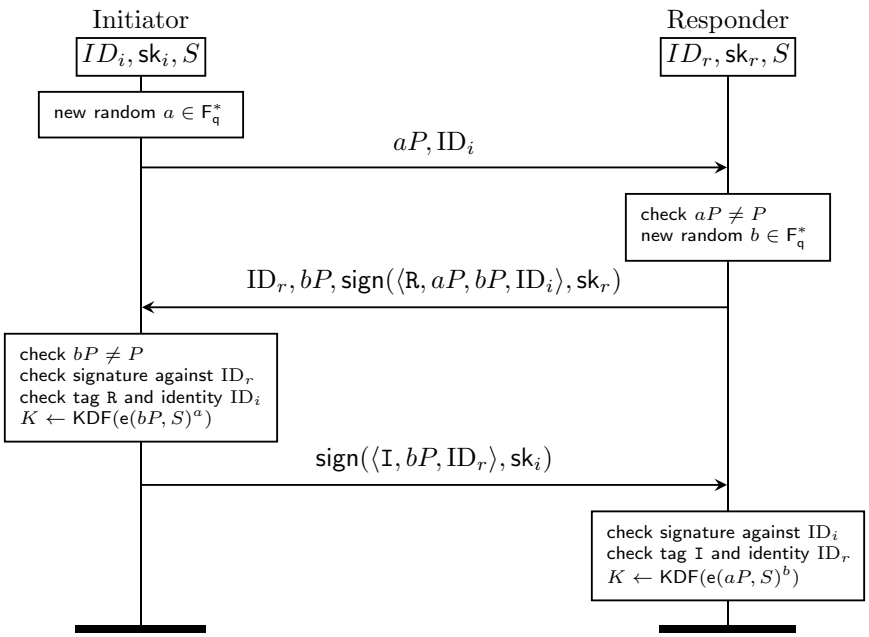

Fig. 13. Description of BP-IBS (4).

BP-IBS (4) in Figure 13. We were able to prove automatically with TAMARIN that our model of the resulting protocol fulfills all its security requirements. We provide the full TAMARIN model of BP-IBS equipped with all our fixes as `BP-IBS_4.spthy` in [8]. Our industrial partner acknowledged the problems.

As expected, we have established that a compromise of AUTH-PKG breaks all secrecy properties of the session keys, and a compromise of SIGN-PKG violates all authentication properties.

6 Conclusion

Our case studies support the thesis that symbolic methods are very useful for improving the security of cryptographic protocols. A prerequisite, of course, is that the methods can handle the protocol's cryptographic primitives and complexity. We showed that current state-of-the-art tools can be used for a larger set of protocols than those previously considered and with reasonable effort. At the same time, our work highlights limitations of the state-of-the-art when faced with realistic protocols involving equational theories, like bilinear pairing in combination with exclusive-or, which lead to combinatorial explosions due to branching during proof search. Another limitation is with respect to equational theories that currently cannot be handled by any state-of-the-art tools,

e.g., Diffie-Hellman exponentiation combined with addition in the exponent that distributes over multiplication.

We see two directions for future work. The first is to expand the set of equational theories that can be handled by tools like Tamarin. This requires progress in unification theory. The second is for tool-builders to improve their tools' efficiency so that higher levels of branching can be handled. Clearly there are limits in both cases due to the undecidability of the underlying problems, but determining the boundaries of what is possible, and feasible, is important.

Finally, we would like to close by encouraging designers of identity-based protocols to consider applying security protocol verification tools. In this paper, we have illustrated how this can be done. As we have shown, it is straightforward to analyze abstract versions of identity-based protocols and thereby identify and eliminate many kinds of mistakes. Moreover, more precise modeling abstractions can be used, provided the protocol itself is of limited complexity.

Acknowledgments. The authors thank Huawei Singapore Research Center for their support for parts of this research.

References

1. Tamarin Manual. https://tamarin-prover.github.io/manual/.
2. Joonsang Baek, Jan Newmarch, Reihaneh Safavi-Naini, and Willy Susilo. A survey of identity-based cryptography. In *Proc. of Australian Unix Users Group Annual Conference*, pages 95–102, 2004.
3. David Basin and Cas Cremers. Modeling and analyzing security in the presence of compromising adversaries. In *Computer Security - ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 340–356. Springer, 2010.
4. David Basin and Cas Cremers. Know your enemy: Compromising adversaries in protocol analysis. *ACM Trans. Inf. Syst. Secur.*, 17(2):7:1–7:31, November 2014.
5. David Basin, Cas Cremers, and Catherine Meadows. Model checking security protocols. In *Handbook of Model Checking*, pages 727–762. Springer, 2018.
6. David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirović, Ralf Sasse, and Vincent Stettler. A formal analysis of 5G authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1383–1396, New York, NY, USA, 2018. ACM.
7. David Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security*, pages 1144–1155. ACM, 2015.
8. David Basin, Lucca Hirschi, and Ralf Sasse. Case study Tamarin models. https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/idbased, 2019. Accessed: 2019-03-05.
9. Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.
10. Jae Cha Choon and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In *International workshop on public key cryptography*, pages 18–30. Springer, 2003.

11. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All about Maude-a high-performance logical framework: how to specify, program and verify systems in rewriting logic*. Springer-Verlag, 2007.
12. Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1773–1788. ACM, 2017.
13. Danny Dolev and Andrew C. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, March 1983.
14. Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1197–1210. ACM, 2015.
15. Jannik Dreier, Lucca Hirschi, Saša Radomirović, and Ralf Sasse. Automated unbounded verification of stateful cryptographic protocols with exclusive OR. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 359–373. IEEE Computer Society, 2018.
16. Santiago Escobar, Catherine Meadows, and José Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. *Theoretical Computer Science*, 367(1-2):162–202, 2006.
17. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO'86*, pages 186–194. Springer, 1986.
18. Dennis Longley and S. Rigby. An automatic search for security flaws in key management schemes. *Computers & Security*, 11(1):75–89, 1992.
19. Catherine Meadows. The NRL Protocol Analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.
20. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.
21. Jonathan Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM, 2001.
22. Jonathan K. Millen, Sidney C. Clark, and Sheryl B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, 13(2):274–288, 1987.
23. Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF)*, pages 78–94, 2012.
24. Benedikt Schmidt, Ralf Sasse, Cas Cremers, and David 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. IEEE Computer Society, 2014.
25. Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.
26. Paul F. Syverson and Catherine Meadows. A formal language for cryptographic protocol requirements. *Designs, Codes and Cryptography*, 7(1-2):27–59, 1996.