Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Formal Methods for Information Security

Project Report

Group 8
Eiman Alnuaimi - 22-944-367
Oliver Bergqvist - 21-957-394

Department of Computer Science
ETH Zurich
May 31, 2024

Contents

1 PACE Protocol

1.1 A simple challenge-response protocol

We implement a simple MAC-based challenge-response protocol between an initiator A and a responder B in theory P1 defined as follows:

$$A \to B : x$$
$$B \to A : [x]_{k(B,A)}$$

To implement the MAC, we use user-defined binary function *mac\2* and define the equation *verify(k, m, mac(m, k)) = true* to verify the correctness of the MAC. We force the use of unidirectional keys by defining a restriction that eliminates traces where A and B have setup the same key to be used on both directions of the communication. Moreover, we define an executability lemma to verify the correctness of the model. The initiator's injective agreement and the executability lemmas are verified by Tamarin. By inspecting the dependecy graph, shown in figure 1, generated from the executability lemma, we can see that an attacker can still act as a regular protocol participant by replacing the messages being exchanged in the protocol (i.e. the nonce x and the mac message).

1.2 Mutual authentication

a) We combine two instances of P1, with alternating senders who agree on two nonces x and y. We model theory P2a as follows:

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [y]_{k(A,B)}$$
$$B \to A : [x]_{k(B,A)}$$

In P2a, we enhance the executability lemma by specifying all parameters that must be agreed upon at the conclusion of the protocol run in the *Finish* action fact statement for both parties, A and B. Moreover, we enhance the unidirectional key restriction such that both parties, A and B, can own two distinct keys, $k(A, B)$ and $k(B, A)$.

The injective agreement lemma for both the responder and the initiator is falsified by Tamarin. Tamarin finds a reflection attack that exploits the symmetry of the protocol, allowing two agents with the Initiator role to complete the protocol without the involvement of an agent in the Responder role.

b) The attack found can be described as follows, where E(agent) is an attacker pretending to be the agent in the brackets.

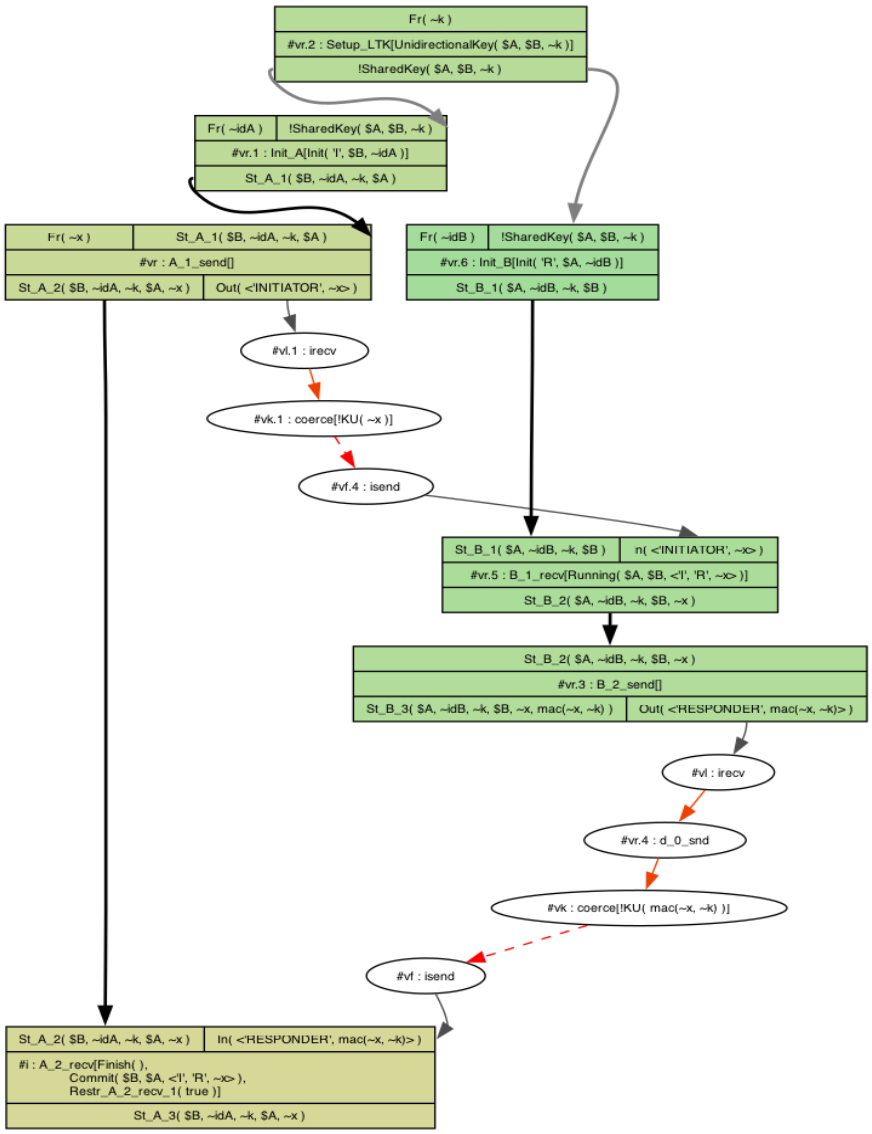

Figure 1: Attack found by the executability lemma in theory P1 where an attacker can still act as a regular protocol participant by replacing messages in the protocol

$$A_1 \to A_2 : x \tag{1}$$
$$A_2 \to E(A_1) : x* \tag{2}$$
$$E(A_1) \to A_2 : x \tag{3}$$
$$E(A_2) \to A_1 : y \tag{4}$$
$$A_1 \to A_2 : [y]_{k(A_1,A_2)} \tag{5}$$
$$A_2 \to A_1 : [x]_{k(A_2,A_1)} \tag{6}$$

The problem with this attack is that the attacker can get two parties to commit to nonces that aren't part of the original communication. The nonces that should be committed to by the end of the protocol are x and x*. However, in step 4, the attacker sends nonce y of their choosing instead of the expected nonce x* coming from A_2. Moreover, the attacker exploits the symmetry of the protocol to force two agents A_1 and A_2 of the same role Inititor to carry out the protocol without the involvement of an agent B of Role Responder.

To fix the protocol in theory P2b, both parties need to agree on both nonces, x and y. This can be achieved by including both x and y in the MAC messages from both A and B. We noticed that this solution is sufficient to break the symmetry in the protocol and prevent the reflection attack. Our proposed fix is the following:

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [x, y]_{k(A,B)}$$
$$B \to A : [x, y]_{k(B,A)}$$

1.3 Introducing a session key

a) In theory P3a, we introduce the session key $K_{ab} = kdf(k(A, B), x, y)$ to MAC the messages in both directions and implement tagging, where each sender's role is included in the MACed message with the other role's nonce. We use tagging to avoid symmetry based attacks.

Tamarin verifies the injective agreement lemma for both the responder and the initiator. Morevoer, the secrecy lemma of the session key was verified too. The implemented protocol can be defined as the following:

$$A \to B : x$$
$$B \to A : y$$
$$A \to B : [{}'I', y]_{K_{ab}}$$
$$B \to A : [{}'R', x]_{K_{ab}}$$

b) We can include each role's own nonce in the agreement, even though the nonce is not MACed, because the session key is derived using both nonces, x and y. Therefore, agreeing on the session key implies that both nonces x and y are agreed upon, despite not MACing each role's nonce. This

approach would not have worked in P2, as the keys used for MACing did not depend on the role's nonce. Consequently, in the case of P2, MACing both nonces x and y was essential to achieve agreement.

1.4 Replace the password by a nonce

As instructed, the long-term password K(A,B) has been replaced by a nonce s generated by A such that the derived session key is now $K_{ab} = kdf(s, x, y)$. The protocol in theory P4 is defined as the following.

$$
\begin{aligned}
&A \to B : [s]_{h(K(A,B))}, x \\
&B \to A : y \\
&A \to B : [{'I'}, y]_{K_{ab}} \\
&B \to A : [{'R'}, x]_{K_{ab}}
\end{aligned}
$$

Similarly to part 1.3, Tamarin verifies the injective agreement lemma for both the responder and the initiator and the secrecy of the session key.

1.5 Introducing Diffie-Hellman: The PACE protocol

Please note that for this subsection, Tamarin doesn't not terminate when using automated proving and we had to prove the theories manually.

a) We refine theory P4 such that the nonces x and y are now replaced by Diffi-Hellman half-keys g^x and g^y. The generator g is defined as $g = map(s, p)$ where p is a public domian paramater and s is a nonce generated by A. Moreover, the session key is now defined as $K_{ab} = h(g^{xy})$. We tag the MACs with the sender's role. Theory P5ab is defined as the following:

$$
\begin{aligned}
&A \to B : g^x, [s]_{h(k(A,B))} \\
&B \to A : g^y \\
&A \to B : [{'I'}, g^y]_{K_{ab}} \\
&B \to A : [{'R'}, g^x]_{K_{ab}}
\end{aligned}
$$

Tamarin verifies the injective agreement lemma for both the responder and the initiator and the secrecy of the session key.

b) In this part, we don't need to change anything in the protocol hence the protocol definition from part a still applies. We add the the perfect forward secrecy lemma to theory P5ab which gets proven by Tamarin.

c) In the text-book definition of Diffie-Hellman key exchange, the base g and the prime number p are parameters that Alice and Bob agree upon publicly. In the case of the PACE protocol, g is

derived from the secret s and the public field p hence, the secrecy of g hinges on the secrecy of s. The randomness in s provides an additional layer of security. Even if an attacker knows p, without knowledge of s they cannot determine g hence they can't participate in the protocol. This ensures that only Bob can establish a session key with Alice since they have the key $k(A, B)$ to decrypt $[s]_{h(k(A,B))}$ and participate in the protocol.

If g was ever to become public, it means that an attacker was able to attain the long-term key $k(A, B)$ and now is able to participate in the protocol. One attack that can be performed in this case is Man in the Middle attack where Eve can establish two different session keys, one with Alice and the other with Bob, hence, intercept their communication. Eve can establish a session key with Alice as follows:

$$\begin{aligned} A &\to E(B) : g^x, [s]_{h(k(A,B))} \\ E(B) &\to A : g^z \\ A &\to E(B) : [{}'I', g^z]_{h(g^{xz})} \\ E(B) &\to A : [{}'R', g^x]_{h(g^{xz})} \end{aligned}$$

Eve can pick any exponent, here, she picks z. A session key $K_{ae} = h(g^{xz})$ is derived from their half-keys. Similarly, Eve can carry out a similar attack with Bob where she can establish a session key $K_{be} = h(g^{yz})$ with Bob and start intercepting the communication between Alice and Bob.

d) The injective agreement of the initiator no longer holds after removing the tags from the MACs. We notice that the initiator carries out the protocol by talking to itself which means with unification, the protocol is prone to reflection attacks. Figure 2 shows the dependency graph of the attack and we model the attack as follows:

$$\begin{aligned} A &\to A : g^x, [s]_{h(k(A,B))} \\ A &\to A : g^y \\ A &\to A : [g^y]_{k_{aa}} \\ A &\to A : [g^x]_{k_{aa}} \end{aligned}$$

To fix this issue, we ensure the g^x and g^y differ by using the embedded restriction $_restrict(not(gy = g^x))$. Tamarin now discards half-keys that are equal, hence, reflection attack is no longer possible. All the lemmas are now verified by Tamarin.

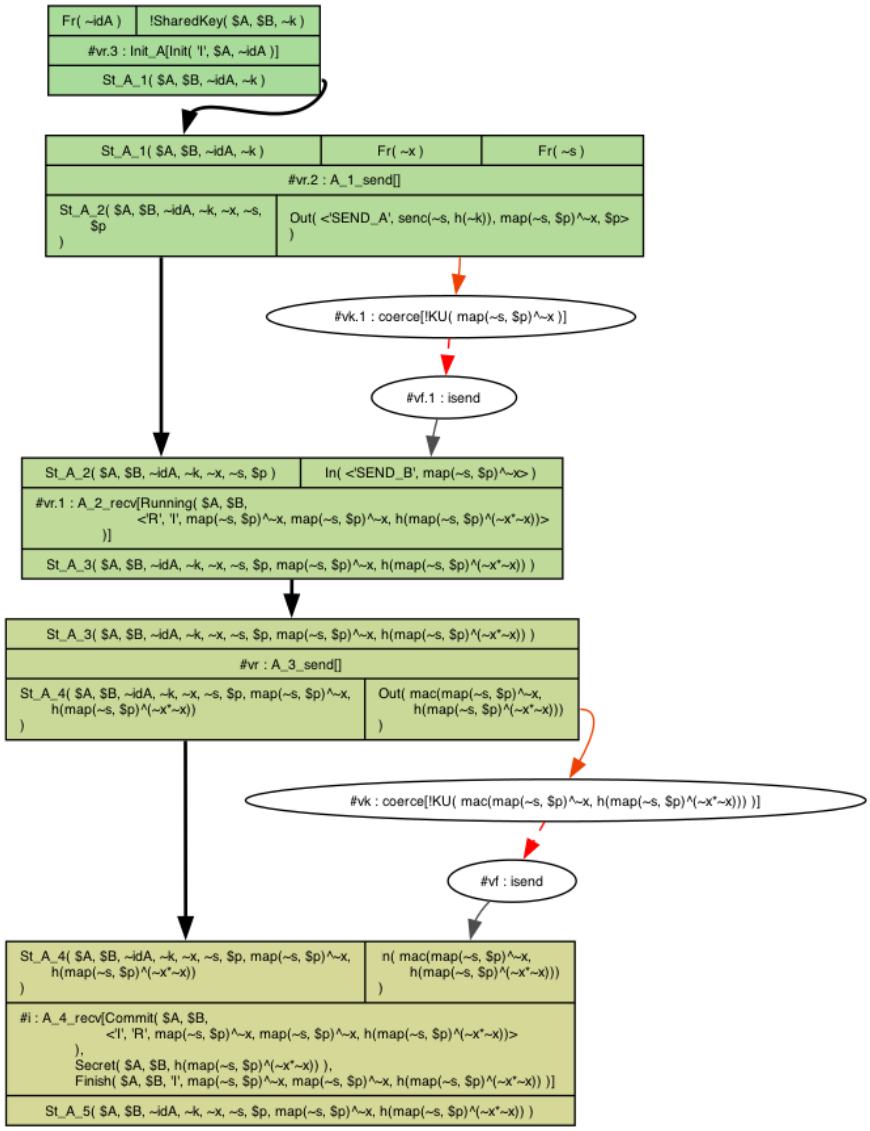

Figure 2: Attack found by the initiator injective agreement lemma in theory P5d where an attacker performs a reflection attack

2 The Off-the-Record Messaging Protocol

The Off-the-Record Messaging protocol consists of an authenticated key exchange followed by one or more messages, continuously refreshing the session key [1]. This section describes a Tamarin model of the key exchange and various improvements. They work with primitives such as Diffie-Hellman key exchange, digital signatures and public-key cryptography.

For the following Tamarin models we use standard key registration action facts:

```
rule Generate_key_pair:
  [ Fr(~sk) ] --> [ !Ltk($A, ~sk), !Pk($A, pk(~sk)), Out(pk(~sk)) ]

rule Reveal_secret_key_A:
  [ !Ltk(A, ka) ]--[ Reveal(A)]->[ Out(ka) ]

rule Reveal_secret_key_B:
  [ !Ltk(B, kb)]--[Reveal(B) ]->[ Out(kb) ]
```

2.1 Modeling the original OTR Key Exchange

The original OTR Key Exchange as described in [1] consists of the following steps:

$$A \to B : g^x, \{g^x\}_{sk(A)}, PK(sk(A))$$
$$B \to A : g^y, \{g^y\}_{sk(B)}, PK(sk(B))$$

Here, g denotes a publicly known Diffie-Hellman group generator (it may not be the neutral element, for example), $sk(\cdot)$ denotes a player's secret key, $PK(sk(\cdot))$ denotes a player's public key, and $\{\cdot\}$ denotes encryption. This protocol is modeled in `OTR1.spthy`, where it executes as expected.

The Tamarin dependency graph is essentially a one-to-one translation of the alice-and-bob notation using the key infrastructure above.

2.2 Authentication Failure

The above OTR Key Exchange model ought fulfill security properties such as secrecy of g^{xy} and mutual injective agreement on g^x and g^y. Although it fulfills secrecy and injective *responder* agreement, it violates injective *initiator* agreement. The failure can be found under `OTR-proofs/OTR2-proof.spthy` As one can see in the Tamarin attack graph in figure 3, the adversary intercepts the initiator `A`'s message, and substitutes in its own keypair `sk.2`. This is the same attack as described in [1]:

$$\begin{aligned} A \to E[B] &: g^x, \{g^x\}_{sk(A)}, PK(sk(A)) \\ E \to B &: g^x, \{g^x\}_{sk(E)}, PK(sk(E)) \\ E \leftarrow B &: g^y, \{g^y\}_{sk(B)}, PK(sk(B)) \\ A \leftarrow E[B] &: g^y, \{g^y\}_{sk(B)}, PK(sk(B)) \end{aligned}$$

2.3 Improvement

This attack is largely possible, because the responder is unable to convey the intended recipient of its message. In order to amend this, `OTR/OTR3.spthy` models the improvement as described in

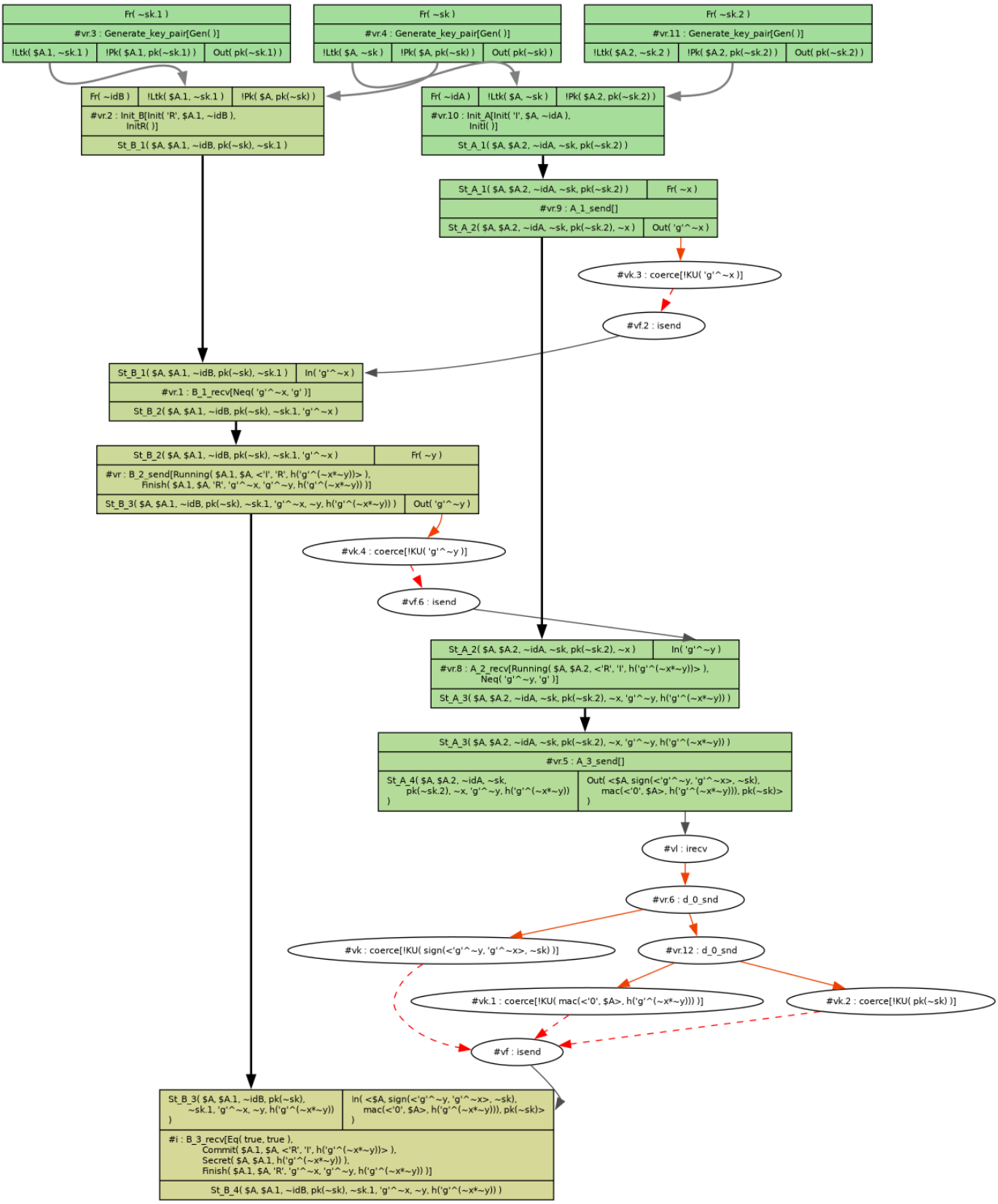

Figure 3: OTR attack on injective responder agreement

Section 3.1 in [1], satisfying mutual agreement and secrecy:

$$A \to B : g^x, \{g^x, B\}_{sk(A)}, PK(sk(A))$$
$$A \leftarrow B : g^y, \{g^y, A\}_{sk(B)}, PK(sk(B))$$

Now, the initiator is able to detect that the responder's message was not intended for the initiator, and does not commit to the protocol run:

$$\begin{aligned} A \to E[B] &: g^x, \{g^x, B\}_{sk(A)}, PK(sk(A)) \\ E \to B &: g^x, \{g^x, E\}_{sk(E)}, PK(sk(E)) \\ E \leftarrow B &: g^y, \{g^y, E\}_{sk(B)}, PK(sk(B)) \\ !!A \leftarrow E[B] &: g^y, \{g^y, E\}_{sk(B)}, PK(sk(B)) \end{aligned}$$

This can be verified in the proof `OTR-proofs/OTR3-proof.spthy`.

2.4 SIGMA

According to [1], another possible solution to OTR's vulnerability is the SIGMA protocol:

$$\begin{aligned} &A \to B : g^x \\ &A \leftarrow B : g^y \\ &A \to B : A, \{g^y, g^x\}_{sk(A)}, [0, A]_{h(g^{xy})}, PK(sk(A)) \\ &A \leftarrow B : B, \{g^x, g^y\}_{sk(B)}, [1, B]_{h(g^{xy})}, PK(sk(B)) \end{aligned}$$

When directly modeling this protocol in Tamarin, it satisfies injective initiator agreement and secrecy. Now, although it violates responder agreement – even noninjective responder agreement – this is to be expected. This is because the responder simply receives a Diffie-Hellman share g^x, responds with a share g^y of its own, and finally receives $X, \{g^y, g^x\}_{sk(X)}, [0, X]_{h(g^{xy})}, PK(sk(X))$, where X is any player and B is the responder. Note that, at this stage the adversary knows g^x as well as g^y. Based on the final message, the responder has no way of deducing that the initiator *believes* it is running with the responder. This is best illustrated in the Tamarin attack graph in Figure 4, and corresponds to the following in alice-and-bob notation:

$$\begin{aligned} A \to E &: g^x \\ E \to B &: g^x \\ A \leftarrow B &: g^y \qquad \text{A claims } \texttt{Running}(h(g^x, g^y)) \text{ with E here!} \\ A \to E &: A, \{g^y, g^x\}_{sk(A)}, [0, A]_{h(g^{xy})}, PK(sk(A)) \\ E \to B &: A, \{g^y, g^x\}_{sk(A)}, [0, A]_{h(g^{xy})}, PK(sk(A))!! \\ A \leftarrow B &: B, \{g^x, g^y\}_{sk(B)}, [1, B]_{h(g^{xy})}, PK(sk(B)) \end{aligned}$$

This attack, along with other security properties, can be verified in `OTR-proofs/OTR4-proof.spthy`.

3 References

[1] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure Off-the-Record Messaging. In WPES, pages 81–89, 2005

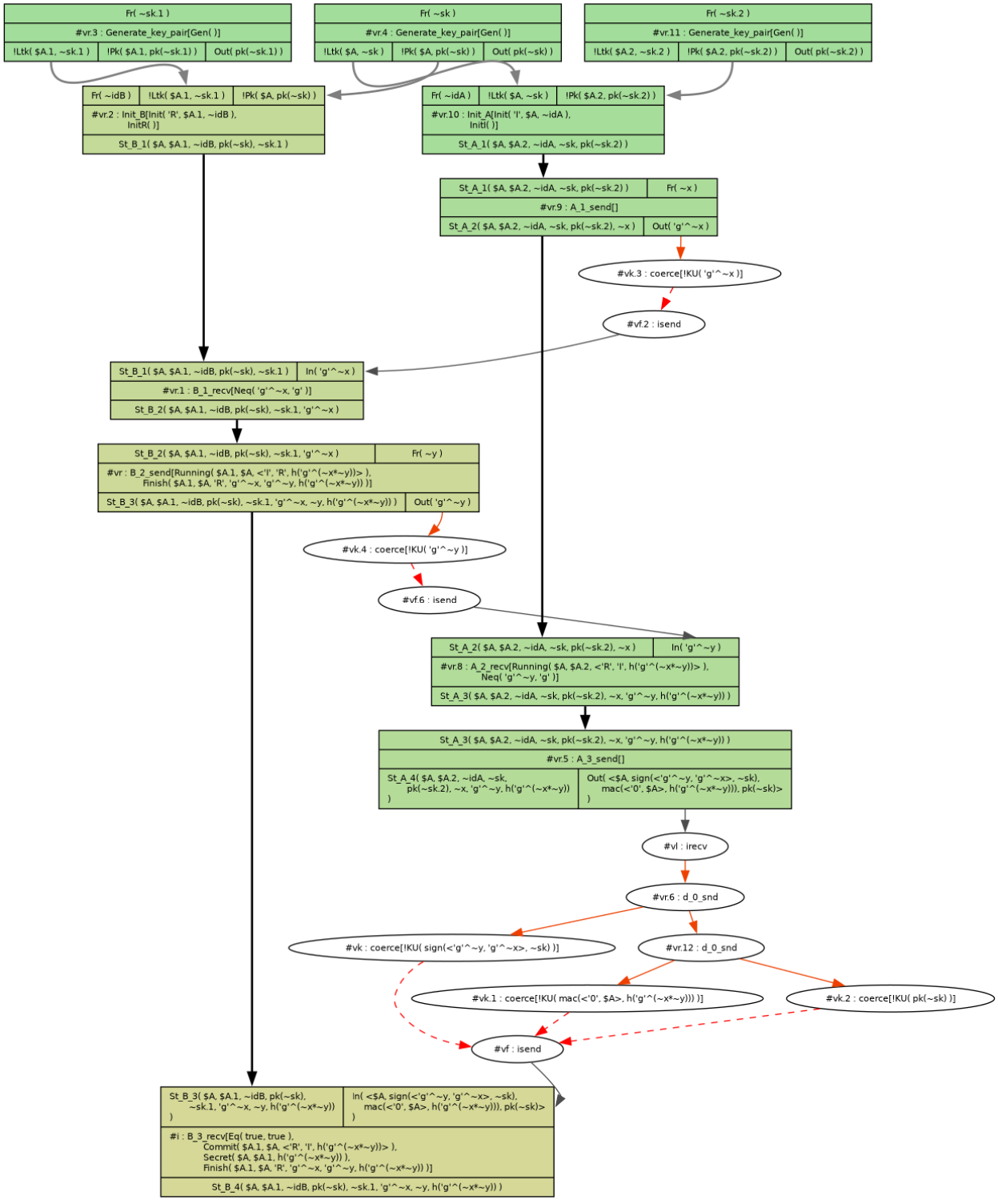

Figure 4: SIGMA Noninjective Responder Agreement Attack