1 Protocols problem set

This document presents four possible protocols that we propose to use for the protocol side of the crypto ladder. The proposed protocols are, ordered by increasing proof difficulty in the crypto world:

- Basic-Hash (Example 1), a RFID tag/reader access protocol, meant to provide unlinkability, a strong notion of anonymity, between the tags. This is the base pattern for electronic passport authentication.
- Signed DH (Example 2), a simple key-exchange providing secrecy and authentication, which is the historic pattern for SSH/TLS.
- Signed KEM (Example 3), a bilateral authenticated variant relying on KEMs, illustrating subtle attacks against key-exchanges.
- Signed DH+KEM (Example 4), a hybrid variant with both KEM and DH following the pattern for the ongoing hybridization of SSH/TLS.
- A simplified NTOR protocol (Example 5), a DH key-exchange using long term DH keys for authentication instead of signatures, which is the base pattern for Wireguard and PQXDH.

Rational behind this current set:

- provide a variety of security properties (secrecy, authentication, privacy);
- have examples linked with widely deployed real-world applications or real-world attacks;
- start with a very simple example and increase difficulty.

Open questions:

- should we formalize the expected security properties for each problem ? The issue is that no common formalism might make sense for both the ProVerif/Tamarin side and the more crypto like tools.
- should we consider advanced compromise scenarios for the key-exchanges (long term key compromise, ephemeral key compromise) ?
- is this too many examples ? too few ?

Notations Secret keys and key pairs are denoted by using the subscripts sk/pk to distinguish between secret/public parts. We rely on usual algorithms for KEM, signatures and DH. We use a generic hash function Hash without a fixed arity, modelers may use the most canonical way to represent those computations in their tools.

1.1 Basic-Hash

The protocol is given in Figure 1

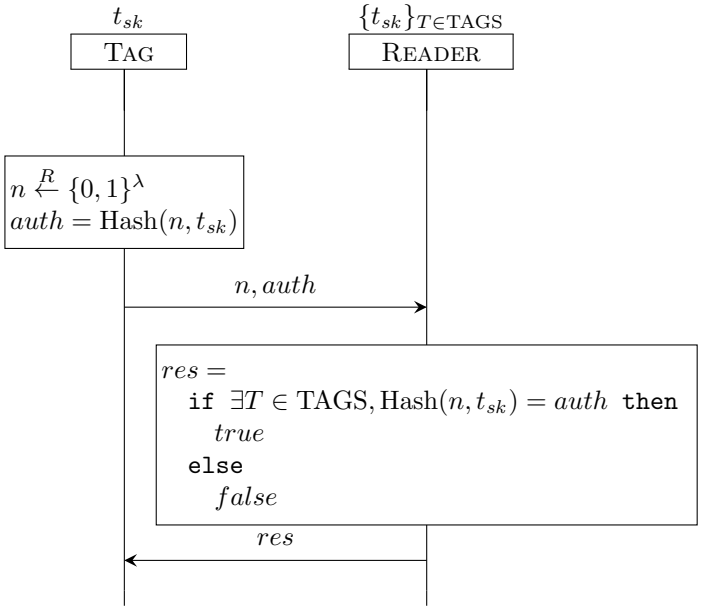

Figure 1: Basic Hash

Description For a set TAGS of valid identities, a RFID reader has a database of valid secret keys $\{t_{sk}\}_{T \in \text{TAGS}}$. A RFID tag with identity T and key t_{sk} authenticates to the reader by sampling a fresh challenge n, and sending the pair $(n, \text{Hash}(n, t_{sk}))$ to the reader, which looks into its database to see if the hash can be mapped to a valid identity and then answers true or false.

Properties This protocol should provide two guarantees against an active machine-in-the-middle attacker:

- non-injective authentication - if the reader accepts some message as coming from some tag, the corresponding tag did send this message at some point in the past.
- unlinkability - it should be impossible to decide whether two sessions of the protocol correspond to the same tag or not. That is, it should be impossible to distinguish a scenario where there is a single tag executing n times the protocol in sequence with a scenario where n distinct tags all execute the protocol each a single time.

Bonus: Show that injective authentication does not hold. The corresponding attack trace is given in Appendix A, Figure 6..

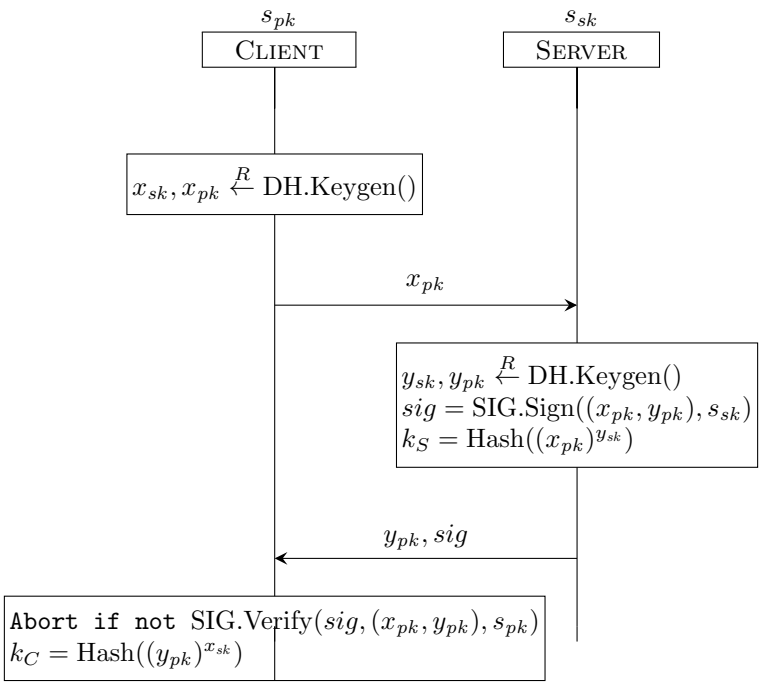

Figure 2: signed Diffie-Hellman key exchange

1.2 signed DH key exchanges

The protocol is given in Figure 2.

Description The server has a long term signature keypair s_{sk}, s_{pk}. This key is used to authenticate the server, while the client and the server exchange fresh DH values to derive a shared secret key.

Properties This protocol should provide two guarantees against an active machine-in-the-middle attacker:

- injective unilateral client-side authentication - if a client derives a key with some parameters (the set of public keys), a corresponding server session derived the same key with the same parameters.
- secrecy - nobody except the two matching sessions can derive the key. The key might be proven to be indistinguishable from a random value.

1.3 signed KEM key exchanges

The protocol is given in Figure 3.

Description The client has a long term signature keypair c_{sk}, c_{pk}. This key is used to authenticate the client, while the client and the server exchange fresh KEM values to derive a shared secret key.

Properties This protocol should provide two guarantees against an active machine-in-the-middle attacker:

- bilateral authentication - TODO
- secrecy - nobody except the two matching sessions can derive the key. The key might be proven to be indistinguishable from a random value.

We want to consider in this problem a KEM that is just IND-CCA, the KEM may thus be implemented using an asymmetric encryption function KEM.AEnc, and with

$$\text{KEM.Encaps}(x_{pk}) := ss \xleftarrow{R} \{0,1\}^{\lambda}; ct \xleftarrow{R} \text{KEM.AEnc}(ss, x_{pk}); \text{return } (ct, ss)$$

Shows that when instantiating the KEM with an asymmetric encryption of a fresh secret, this property does not hold.

Bonus: when considering a scenario where some clients are willing to talk to dishonest parties, this protocol suffers from an Unknown Key Share attack. The corresponding attack trace is given in Appendix A, Figure 7.

Bonus: when allowing attackers to compromise ephemeral keys x_{sk}, we should on a good protocol still be able to prove that whenever a client session and a server session derive the same secret key, then if the corresponding client session x_{sk} was not compromised, the key is fully secret. This corresponds to session independence, where allowing to compromise the material of other sessions does not impact the security of uncompromised sessions. The corresponding attack trace is given in Appendix A, Figure 8.

1.4 signed DH+KEM key exchanges

The protocol is given in Figure 4.

Description The server has a long term signature keypair s_{sk}, s_{pk}. This key is used to authenticate the server, while the client and the server exchange fresh DH and KEM values to derive a shared secret key.

Properties This protocol should provide two guarantees against an active machine-in-the-middle attacker:

- authentication - if a client derives a key with some parameters (the set of public keys), a corresponding server session derived the same key with the same parameters.
- secrecy - nobody except the two matching sessions can derive the key. The key might be proven to be indistinguishable from a random value.

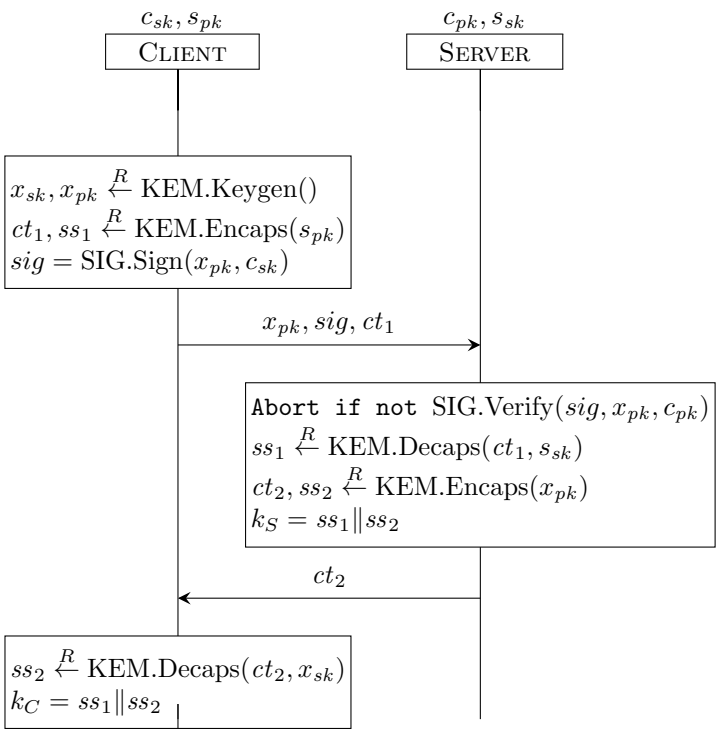

Figure 3: signed KEM key exchange

Bonus: provides proofs where only either the KEM or the DH part is assumed secure.

1.5 full DH key exchanges

The protocol is given in Figure 5.

Description The server has a long term DH keypar s_{sk}, s_{pk}. This key is used to authenticate the server, while the client and the server exchange fresh DH values to derive a shared secret key.

Properties This protocol should provide two guarantees against an active machine-in-the-middle attacker:

- injective authentication - if a client derives a key with some parameters (the set of public keys), a corresponding server session derived the same key with the same parameters.

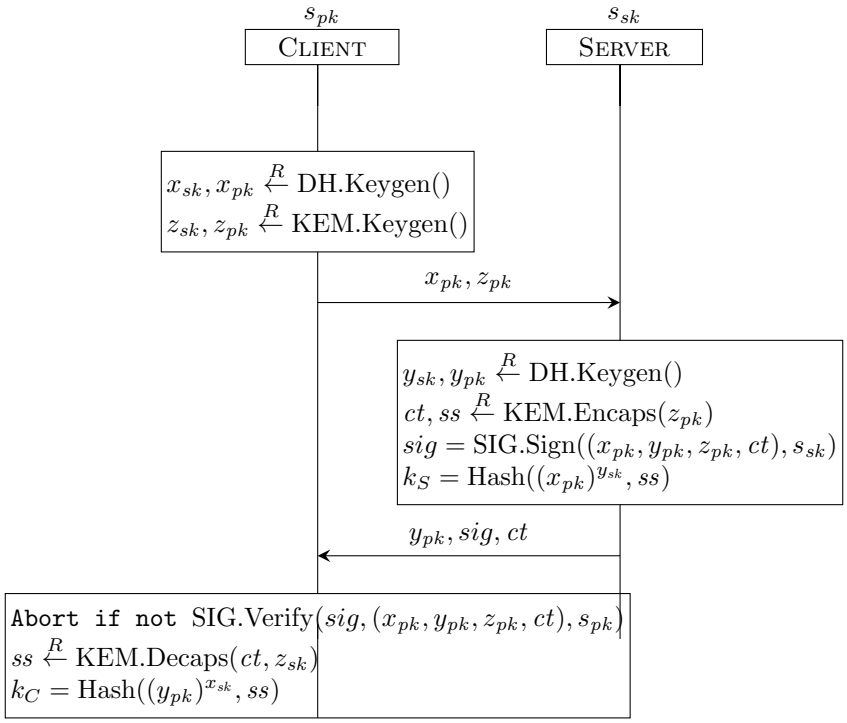

Figure 4: Hybridization of a signed Diffie-Hellman key exchange

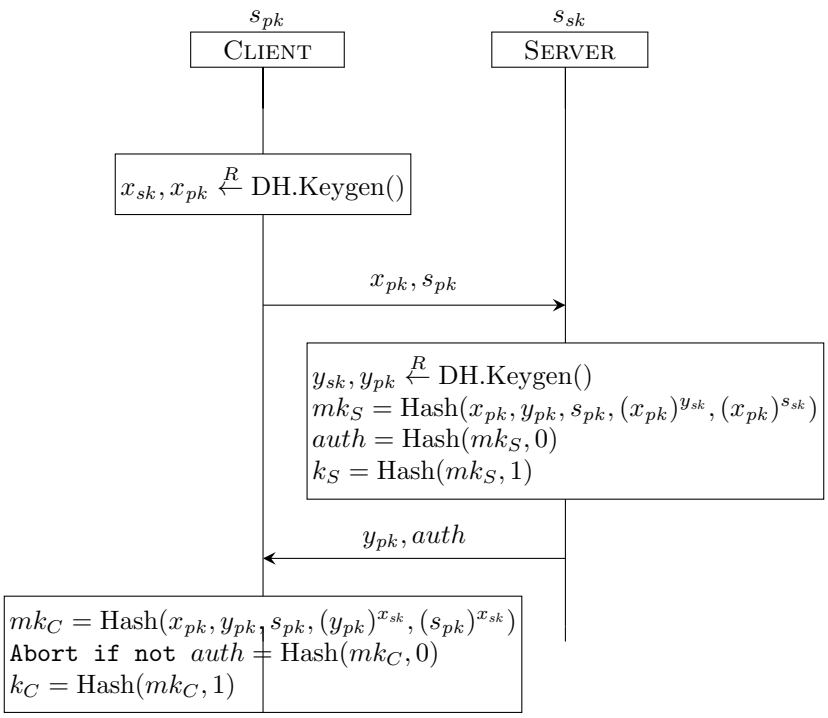

Figure 5: Simplified NTOR

- secrecy - nobody except the two matching sessions can derive the key. The key might be proven to be indistinguishable from a random value.

A Attack traces

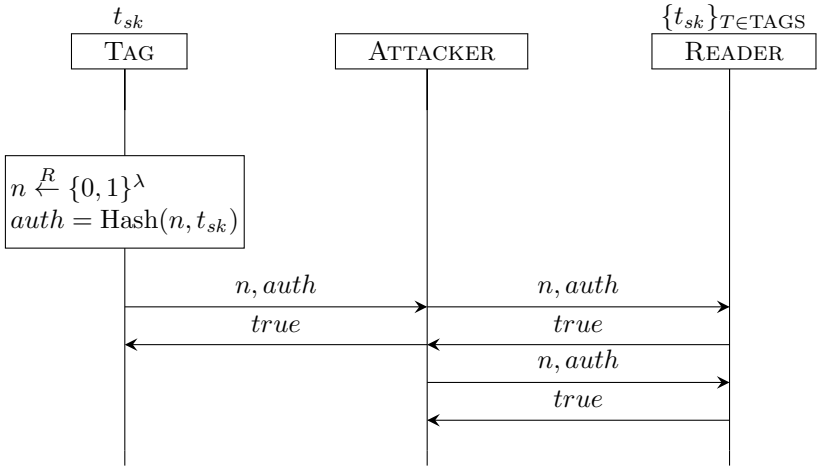

Figure 6: Basic Hash - basic replay attack breaking injective authentication

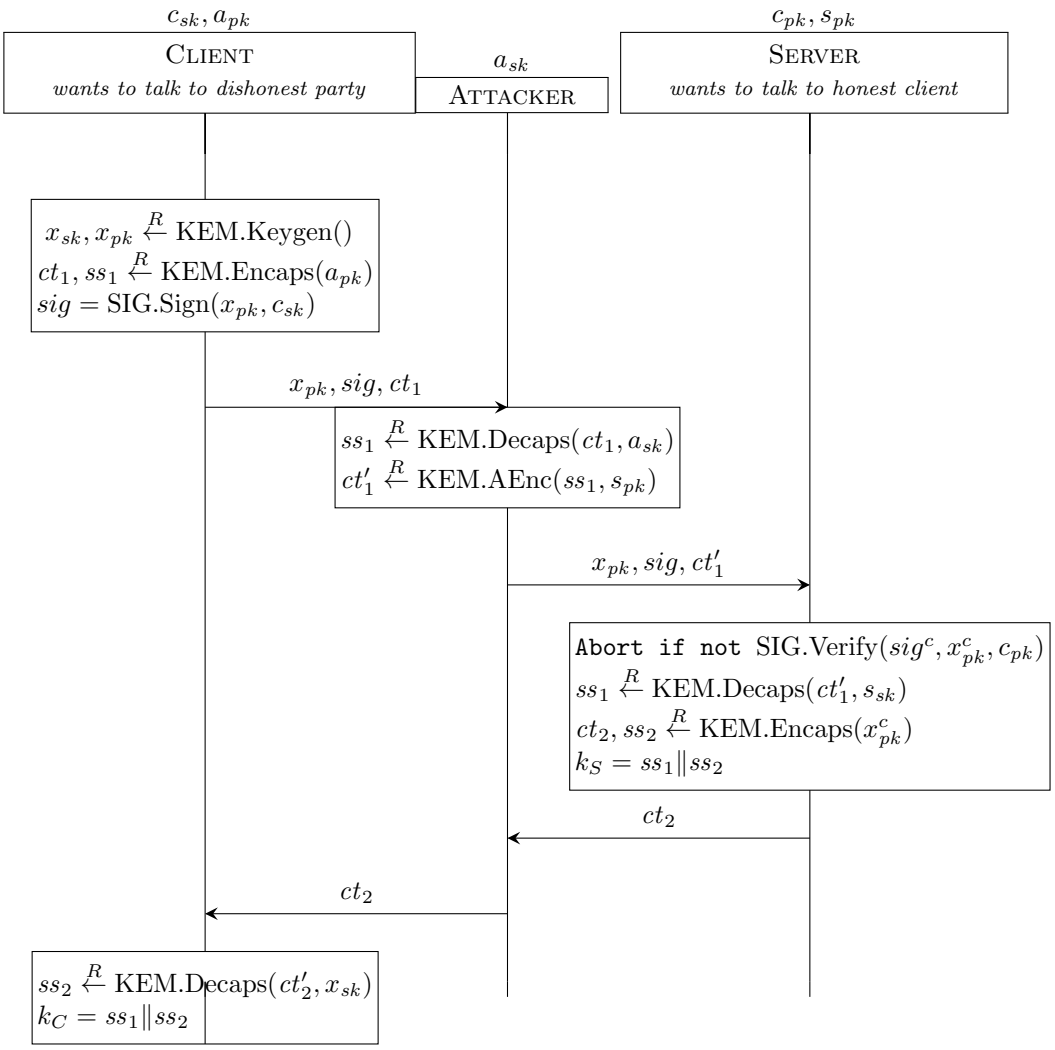

Figure 7: signed KEM key exchange - Unknown Key-share Attack through reencapsulation attack .
Both honest client and honest server derive the same key, but do not believe to be talking together.

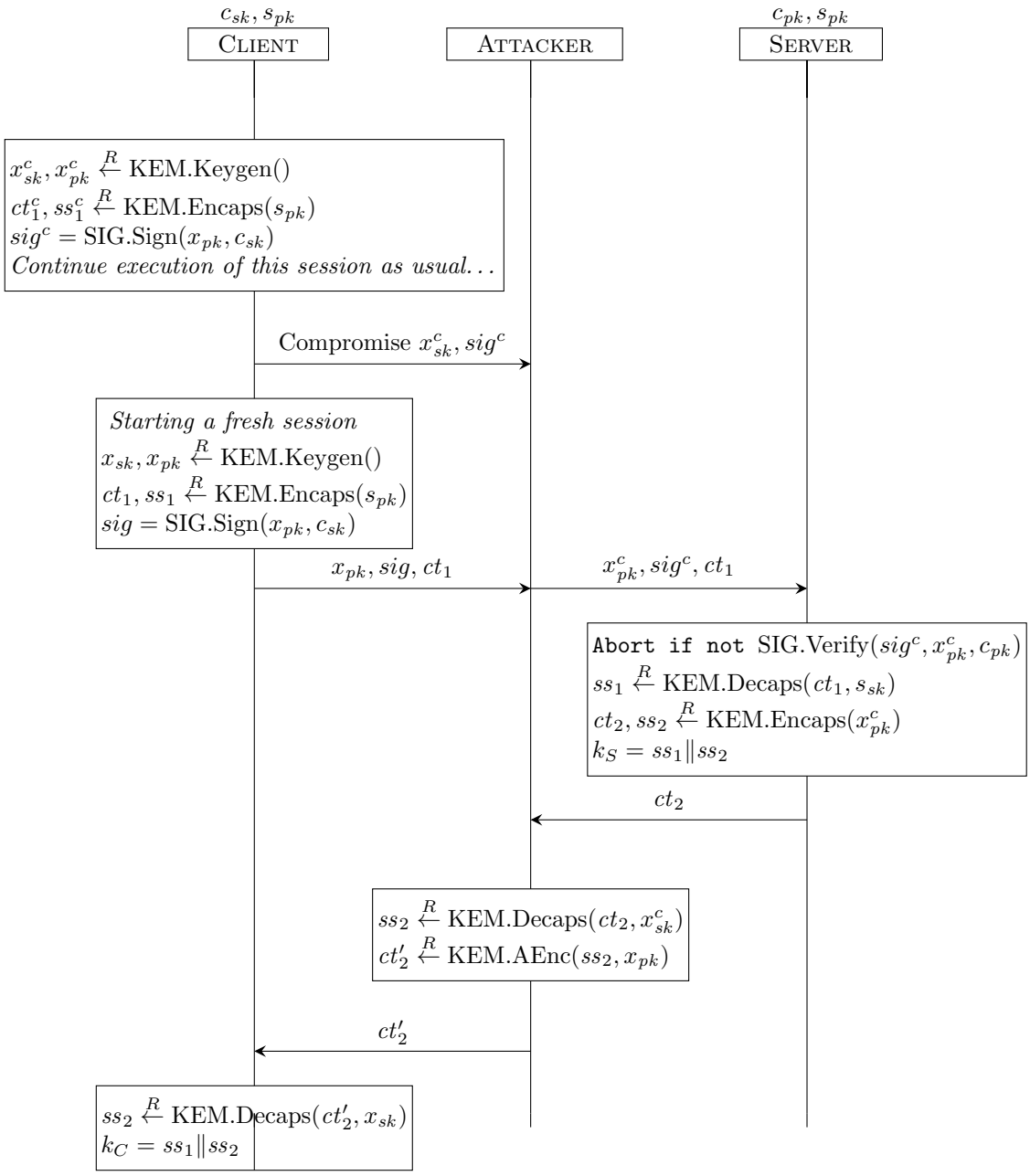

Figure 8: signed KEM key exchange - Reencapsulation attack against session independence.
The attacker learns half of the key of a fresh session by compromising material from an independent session.