Formal Methods for Security and Privacy (SS2021) Prof. Matteo Maffei

Project 1 – ProVerif

TU Wien

Group 5

Our group consists of the following members:

Tobias Eidelpes
01527193

Exercise 1.1: Warm Up

Results for protocol (a): Non-injective agreement is not fulfilled because the attacker can act as a man-in-the-middle. He first sniffs the initial `hello` message and the nonce and forwards it to the responder. The responder sends the hash of the nonce and the secret key. The identifier contained in the response is now modified by the attacker to be his own. Since non-injective agreement is not satisfied, injective agreement is also not satisfied.

By adding the id of the initiator into the hash of the response, the attacker cannot pose as the responder, because the initiator would detect the mismatch.

Results for protocol (b): Non-injective and injective agreement hold.

Exercise 1.2: Self-Signed Certificates

Attack 1: The attacker can again pose as the server S and exchange nonces with the client. Then he sends the signed encryption key to the client. The attacker can decrypt the received secret from the client because the client is using the encryption key sent by the attacker. Since the attacker now knows all three parameters to calculate the symmetric key `k`, it can decrypt the shared secret `m`. The attacker also poses as the client and communicates with the server. Since the attacker knows `m`, it can relay all messages exchanged by the two honest participants without either of them knowing that they are communicating with the attacker.

Attack 1: The second attack works similarly but does not include the honest server as a participant at all. The attacker simply poses as the server from the beginning and the honest client will think he is communicating with the honest server when in fact he is not.

Why does introducing a certification authority fix the problem? By signing the identity of the certificate holder, the attacker cannot pose as the server anymore. Clients are able to distinguish the certificate from the attacker from the one signed by the CA, because the attacker's certificate is not signed by the CA.

Exercise 1.3: Commitment Schemes

Results for `commit1.pv`: The message `m` is kept secret (weak secrecy holds). Strong secrecy holds as well because the attacker does not know the key `k`.

Results for `commit2.pv`: The attacker obviously knows the message `m` after phase 2 because it is transmitted over a public channel by Alice. Since weak secrecy does not hold, strong secrecy cannot hold either.

Results for `version1.pv`: Weak and strong secrecy hold through all phases.

Results for `version2.pv`: Both m_1 and m_3 are kept secret. Strong secrecy, however, does not hold.

Why is only one version considered secure? The second version is not secure because there is only one key k being used for the three messages. After revealing m_2, the attacker gets information about m_1 and m_3.

Exercise 1.4: e-Passports

Notes on Exercise 1.5: Unlinkability is not given by the implementation, because the attacker can distinguish different passports by their error messages. Two passports are created at first and they interact with the reader. The attacker can act as the reader and capture the message. Depending on which error message is received in subsequent communication, the attacker can infer which of the two passports it is talking to. If it receives `error2`, the nonce check failed and if it receives `error1`, the signature check failed. Figure 1 provides a trace as generated by ProVerif.

To fix the protocol to guarantee unlinkability, only one error message is emitted (`error1`) instead of two. For this to work, the check is made whether the signature is correct AND the nonce is correct. If one of them is not correct, `error1` is sent. Since there is no second error message, the attacker is not able to tell different passports apart from each other.

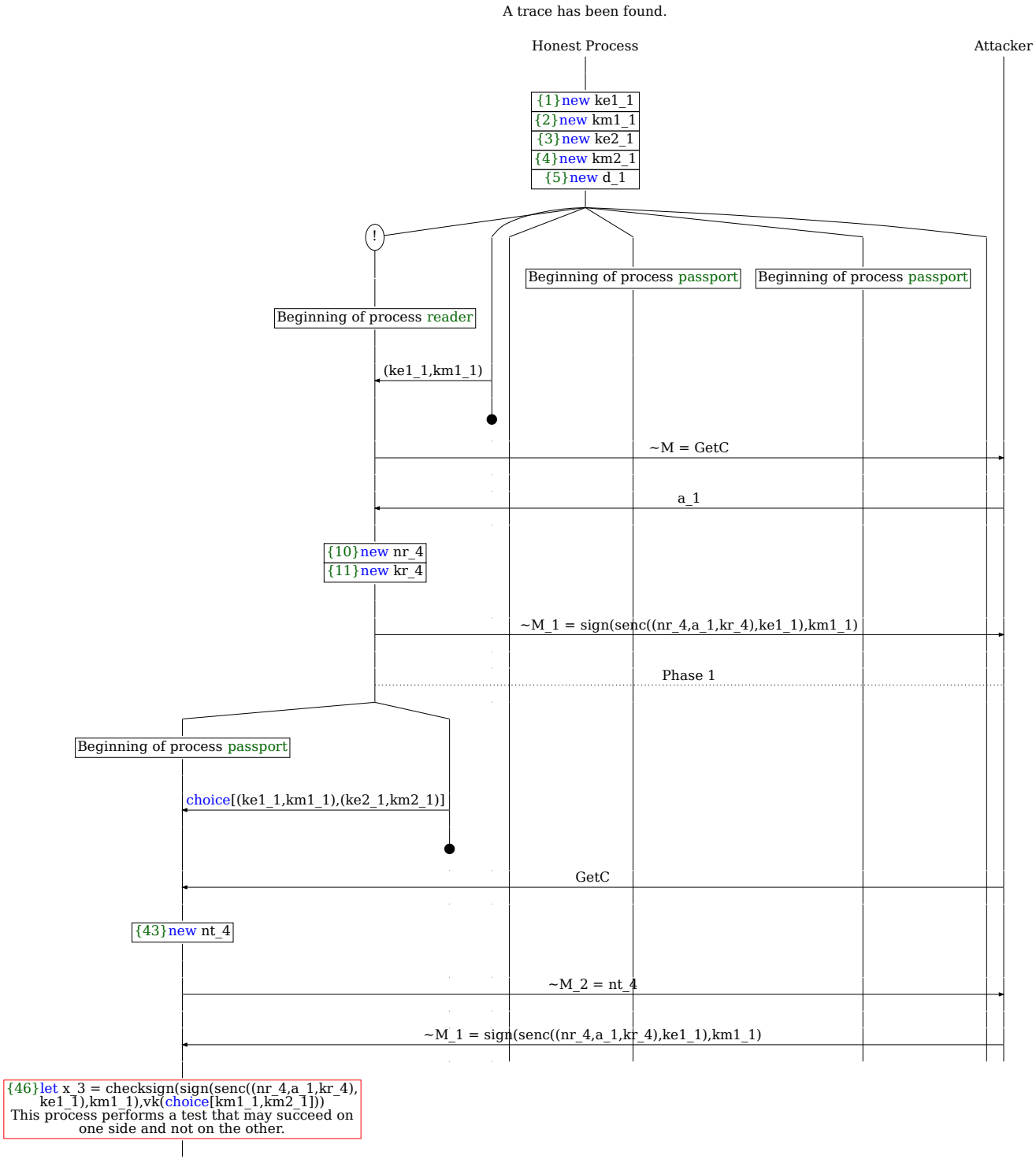

Figure 1: Trace of the attack on the French BAC implementation.