Mechanized Computational Proof of the TLS 1.3 Standard Candidate

Bruno Blanchet

INRIA Paris
Bruno.Blanchet@inria.fr

April 2017

Summary of the result

- Mechanized verification of TLS 1.3 Draft-18 in the computational model.
  - + Handshake with PSK and/or DHE.
  - + Handshake with and without client authentication.
  - + 0-RTT and 0.5-RTT data, key updates.
  - – No post-handshake authentication.
  - – No version or ciphersuite negotiation: only strong algorithms.
  - – For PSK-DHE, we do not prove forward secrecy wrt. the compromise of PSK.
- We prove security properties of the initial handshake, the handshake with pre-shared key, and the record protocol using CryptoVerif.
- We compose these pieces manually.

CryptoVerif, http://cryptoverif.inria.fr/

CryptoVerif is a semi-automatic prover that:

- works in the computational model.
- generates proofs by sequences of games.
- proves secrecy and correspondence properties.
- provides a generic method for specifying properties of cryptographic primitives which handles MACs (message authentication codes), symmetric encryption, public-key encryption, signatures, hash functions, Diffie-Hellman key agreements, ...
- works for N sessions (polynomial in the security parameter), with an active adversary.
- gives a bound on the probability of an attack (exact security).

Proofs by sequences of games

CryptoVerif produces proofs by sequences of games, like those of cryptographers [Shoup, Bellare&Rogaway]:

- The first game is the real protocol.
- One goes from one game to the next by syntactic transformations or by applying the definition of security of a cryptographic primitive. The difference of probability between consecutive games is negligible.
- The last game is "ideal": the security property is obvious from the form of the game.
  (The advantage of the adversary is 0 for this game.)

Game 0: Protocol to prove $\longleftrightarrow$ p_1 negligible $\longleftrightarrow$ **Game 1** $\longleftrightarrow$ p_2 negligible $\cdots$ $\longleftrightarrow$ p_n negligible **Game n**: Property obvious

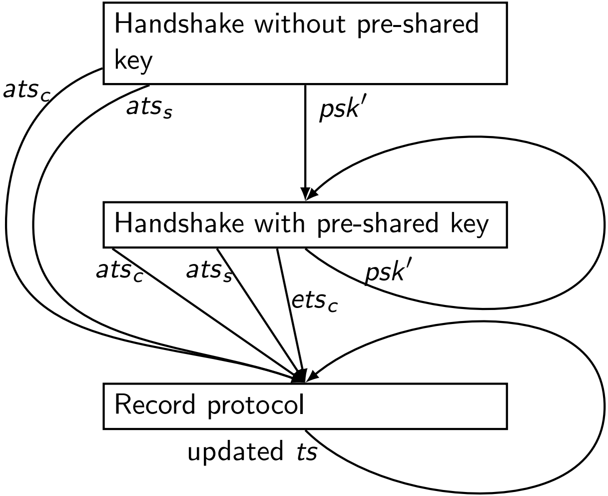
Input and output of the tool

1. Prepare the input file containing
   - the specification of the protocol to study (initial game),
   - the security assumptions on the cryptographic primitives,
   - the security properties to prove.
2. Run CryptoVerif
   - Automatic proof strategy or manual guidance.
3. CryptoVerif outputs
   - the sequence of games that leads to the proof,
   - a succinct explanation of the transformations performed between games,
   - an upper bound of the probability of success of an attack.

Structure of the proof

1. Computational assumptions
2. Lemmas on primitives
3. Protocol pieces
    - Handshake without pre-shared key
    - Handshake with pre-shared key (PSK and PSK-DHE)
    - Record protocol
4. Compose the pieces together

Structure of the proof: final composition



Key schedule (Draft-18, excerpt)

0 → HKDF-Extract

PSK → HKDF-Extract

HKDF-Extract → Early Secret

Early Secret → Derive-Secret(., "external psk binder key" | "resumption psk binder key", "") $=$ *binder_key*

Early Secret → Derive-Secret(., "client early traffic secret", *ClientHello*) $=$ *client_early_traffic_secret*

Early Secret → HKDF-Extract

(EC)DHE → HKDF-Extract

HKDF-Extract → Handshake Secret

Assumptions (1)

- Diffie-Hellman:
  - gap Diffie-Hellman (GDH)
    - needed in particular for 0.5-RTT
  - Diffie-Hellman group of prime order
  - Diffie-Hellman group elements different from $0^{len_{\mathrm{H}()}}$
    - avoids confusion between handshakes with and without Diffie-Hellman exchange.
  - Diffie-Hellman group elements different from $len_{\mathrm{H}()} \| \text{“}\mathtt{TLS\ 1.3,}\text{”} \| l \| h \| \mathtt{0x01}$.
    - avoids collision between HKDF-Extract(es, e) and Derive-Secret(es, pbk, “”) or Derive-Secret(es, ets_c, $\log_1$).
    - independently discovered and discussed on the TLS mailing list.
    - change in Draft-19 makes this assumption unnecessary: add a Derive-Secret stage before HKDF-Extract.

Assumptions (2)

- Signatures: sign is UF-CMA.
- Hash functions: H is collision-resistant.
- HMAC:
  - $x \mapsto \text{HMAC-H}^{0^{len_{\text{H}()}}}(x)$ and $x \mapsto \text{HMAC-H}^{\text{kdf}_0}(x)$ are independent random oracles.
  - HMAC-H is a PRF, for keys different from $0^{len_{\text{H}()}}$ and kdf_0.
- Authenticated Encryption: IND-CPA and INT-CTXT provided the same nonce is never used twice with the same key.

Lemmas on primitives: MAC and signatures

- $\mathsf{mac}^k_{\mathsf{H}}(m) = \mathsf{mac}^k(\mathsf{H}(m))$ is an SUF-CMA MAC.
- $\mathsf{sign}^{sk}_{\mathsf{H}}(m) = \mathsf{sign}^{sk}(\mathsf{H}(m))$ is an UF-CMA signature.

Lemmas on primitives: key schedule

Lemma

When es is a fresh random value,

- $e \mapsto \text{HKDF-Extract}(es, e)$ *and*
- $\log_1 \mapsto \text{Derive-Secret}(es, \text{ets}_c, \log_1)$
  are indistinguishable from independent random functions, and
- $k^b = \text{Derive-Secret}(es, \text{pbk}, \text{``''})$ *and*
- $\text{HKDF-Extract}(es, 0^{len_{\mathrm{H}()}})$
  are indistinguishable from independent fresh random values independent from these random functions.

- Proved using CryptoVerif.
- Similar lemmas for other parts of the key schedule.
- Used as assumption in the proof of the protocol.

Handshake without pre-shared key: model

- Model a honest client and a honest server.
- May interact with dishonest clients and servers included in the adversary.
- Ignore negotiation (`RetryRequest`).
- Give the handshake keys to adversary:
  - The adversary can encrypt and decrypt messages.
  - The security proof does not rely on that.
- Server always authenticated.
- With and without client authentication.
- The honest client and server may be dynamically compromised.

Handshake without pre-shared key: honest sessions

- The client is in a honest session if
  - the server public key is the one of the honest server, and
  - the honest server is not compromised, or it is compromised and the messages received by the client have been sent by the honest server.
- The server is in a honest session if
  - client authenticated:
    - the client public key is the one of honest client, and
    - the honest client is not compromised, or it is compromised and the messages received by the server have been sent by the honest client.
  - client not authenticated: the Diffie-Hellman share received by the server has been sent by the honest client.

Handshake without pre-shared key: security (1)

- Key authentication:
  - If the honest client terminates a honest session, then the honest server has accepted a session with that client, and they agree on:
    - keys ats_c, ats_s, and ems,
    - all messages until the server `Finished` message.
  - If the honest server terminates a honest session, then the honest client has accepted a session with that server, and they agree on the keys and on all messages.
- Replay prevention: the previous properties are injective.
- Key secrecy: the keys
  - ats_c, ems, psk' client side, when the client terminates a honest session;
  - ats_s server side, when the server sends its `Finished` message and the received Diffie-Hellman share comes from the client (for 0.5-RTT)

  are indistinguishable from independent fresh random values.

Handshake without pre-shared key: security (2)

- Same key:
  - If the honest client terminates a honest session and the honest server has accepted a session with the same messages, then they have the same key.
  - If the honest server terminates a honest session and the honest client has accepted a session with the same messages, then they have the same key.
- Unique channel identifier:
  - psk' or $\mathrm{H}(\mathrm{log}_7)$:
    If a client session and a server session have the same psk' or $\mathrm{H}(\mathrm{log}_7)$, then all their parameters are equal (collision-resistance).
  - *ems*:
    If a client session and a server session have the same *ems*, then they have the same log_4 (collision-resistance), so all their parameters are equal (CryptoVerif).

Handshake without pre-shared key: guidance

- Signature under sk_S.
- Introduce tests to distinguish cases, depending on
  - whether the Diffie-Hellman share received by the server is a share $g^{x'}$ from the client,
  - and whether the Diffie-Hellman share received by the client is the share g^y generated by the server upon receipt of $g^{x'}$.
- Random oracle assumption on $x \mapsto \text{HMAC-H}^{\text{kdf}_0}(x)$.
- Replace variables that contain $g^{x'y}$ with their values to make equality tests $m = g^{x'y}$ appear.
- Gap Diffie-Hellman assumption.
- $\Rightarrow$ the handshake secret hs is a fresh random value.
- Lemmas on key schedule $\Rightarrow$ other keys are fresh random values.
- MAC.
- Signature under sk_C.

Handshake with pre-shared key: model

- Includes handshakes with and without Diffie-Hellman exchange.
- Includes 0-RTT.
- Ignore the ticket $\mathrm{enc}^{k_t}(psk)$; consider a honest client and a honest server that share the PSK.
- Give the handshake keys to adversary (as before).
- Certificates optional, since the client and server are already authenticated by the PSK.

Handshake with pre-shared key: security (1)

Same properties as for the initial handshake, but

- No compromise of PSK.
  - Limitation of CryptoVerif: cannot prove forward secrecy wrt. to the compromise of PSK for PSK-DHE.
- Weaker properties for 0-RTT:
  - Key authentication: No authentication for ets_c:
    - several binders, and only one of them is checked;
    - the adversary can alter the others, yielding a different ets_c server-side.
  - Replay prevention: No replay protection for ets_c.
  - Secrecy of keys: The keys ets_c server-side are not independent of each other, due to the replay.

Handshake with pre-shared key: security (2)

For 0-RTT, we show:

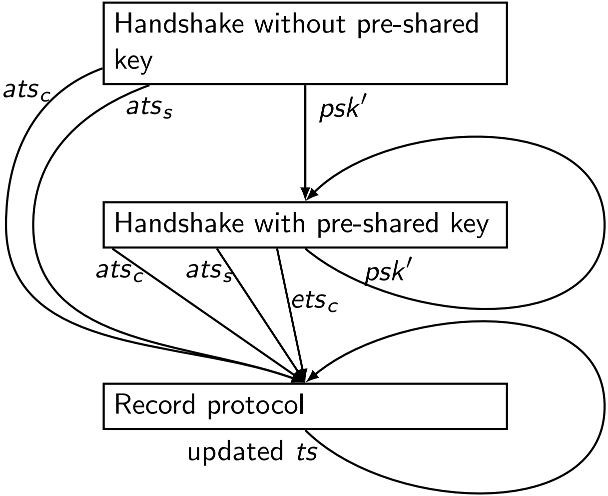
- Client-side: The keys ets_c are indistinguishable from independent random values.
- Server-side:
  - If the received ClientHello message has been sent by the client, then this session matches a session of the client with same key ets_c.
  - Otherwise,
    - If the ClientHello message has been received before, then the key ets_c computed by the server is the same as in the previous session with the same ClientHello message.
    - Otherwise, the key ets_c computed by the server is indistinguishable from a fresh random value, independent from other keys.

Record protocol

The client and the server share a fresh random traffic secret.

- Key secrecy: The updated traffic secret is indistinguishable from a fresh random value.
- Message secrecy: When the adversary provides two sets of plaintexts m_i and m'_i of the same padded length, it is unable to determine which set is encrypted, even when the updated traffic secret is leaked.
- Message Authentication: If a message m is decrypted by the receiver with a counter c, then the message m has been encrypted and sent by an honest sender with the same counter c.
- Replay Prevention: The authentication property above is injective.

Composition



Composition: main theorem (informal)

- System S: key exchange; A and B obtain a key such that:
  - Key secrecy: The keys obtained by A are indistinguishable from independent random values.
  - One-way injective authentication: For each session of B that obtains a key k after sending/receiving $\widetilde{msg}$, there is a distinct session of A that obtains the key k after sending/receiving $\widetilde{msg}$.
  - Same key: If B obtains a key k after sending/receiving $\widetilde{msg}$ and A obtains a key k' after sending/receiving $\widetilde{msg}$, then $k = k'$.
- System S' assumes a fresh random key shared by A' and B'.
- The composed system $S_{composed}$ runs the key exchange followed by A' with the key obtained by A and B' with the key obtained by B.
- We have:
  - $S_{composed}$ is indistinguishable from an adversary using S and
  - $S_{composed}$ is indistinguishable from an adversary using S'

  The security properties of S and S' carry over to $S_{composed}$.

Composition

- The previous theorem allows to perform most compositions.
- More tricky composition theorems for 0-RTT, because the properties are weaker.
- A simpler composition theorem for key update.

Conclusion

- Mechanized verification of TLS 1.3 Draft-18 in the computational model.
  - + Handshake with PSK and/or DHE.
  - + Handshake with and without client authentication.
  - + 0-RTT and 0.5-RTT data, key updates.
  - – No post-handshake authentication.
  - – No version or ciphersuite negotiation: only strong algorithms.
  - – For PSK-DHE, we do not prove forward secrecy wrt. the compromise of PSK.
- CryptoVerif proves properties of the handshake with (resp. without) pre-shared-key and of the record protocol.
- We infer properties of the whole system by manual composition.
- Modular approach essential to be able to handle such a complex protocol.
- TLS 1.3 Draft-18 is well-designed to allow such a proof.