Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

INRIA Paris
Bruno.Blanchet@inria.fr

March 2018

Introduction

- Composition between
  - a key exchange protocol
  - a protocol that uses the key
- Results stated in the CryptoVerif framework:
  - computational model
  - formal framework for stating the composition theorem
  - prove bigger protocols in CryptoVerif
  - prove protocols with loops in CryptoVerif

Adapt and extend previous computational composition results by Brzuska, Fischlin et al. [CCS'11, CCS'14 and CCS'15]

Application to TLS 1.3

Why TLS 1.3 ?

- Important protocol, in the final stages of development
- Well designed to allow composition
- Contains loops:
  - Unbounded number of handshakes and key updates
- Variety of compositions:
  - In most cases, the key exchange provides injective authentication
  - For 0-RTT data = data sent by the client to the server immediately after the message (ClientHello):
    - possible replay, so non-injective authentication
    - variant for the case of altered ClientHello
  - Simpler composition theorem for key updates

Fills a gap in the proof of TLS 1.3 Draft 18 by Bhargavan et al [S&P'18]

- The composition was stated only informally.

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

CryptoVerif is a semi-automatic prover that:

- works in the computational model.
- generates proofs by sequences of games.
- 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).

Reminder on CryptoVerif

- CryptoVerif represents protocols using a process calculus.
- P, Q: processes
- C: context = process with one or several holes $[\,]$
- Adversaries represented by evaluation contexts:

$C ::=$	evaluation context
$\quad [\,]$	hole
$\quad \textbf{newChannel}\ c; C$	channel restriction
$\quad Q \mid C$	parallel composition
$\quad C \mid Q$	parallel composition

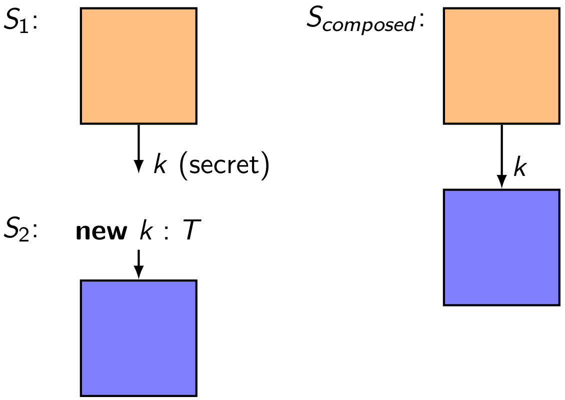
Security properties proved by CryptoVerif

- Indistinguishability: $Q \approx^V Q'$ when an adversary with access to the variables V has a negligible probability of distinguishing Q from Q'.
- Secrecy: Q preserves the secrecy of x with public variables V when an adversary with access to the variables V has a negligible probability of distinguishing the values of x in several sessions from independent random values.
- Correspondences: If some events have been executed, then other events have been executed. Example:

$$\mathbf{event}(e_1(x)) \Longrightarrow \mathbf{event}(e_2(x))$$

  Q satisfies the correspondence $corr$ with public variables V when an adversary with access to the variables V has a negligible probability of breaking $corr$.

The most basic composition theorem



The most basic composition theorem

Theorem (Assumptions)

Let C be any context with one hole, without replications above the hole. Let M be a term of type T. Let

$$S_1 = C[\mathbf{let}\ k = M\ \mathbf{in}\ \overline{c_1}\langle\rangle; Q_1]$$
$$S_2 = c_2(); \mathbf{new}\ k : T; \overline{c_3}\langle\rangle; Q_2$$

where c_1, c_2, c_3 do not occur elsewhere in S_1, S_2; k is the only variable common to S_1 and S_2; S_1 and S_2 have no common channel, no common event, and no common table; and k does not occur in C and Q_1. Let c_1' be a fresh channel. Let

$$S_{composed} = C[\mathbf{let}\ k = M\ \mathbf{in}\ \overline{c_1'}\langle\rangle; (Q_1 \mid Q_2)]$$

The most basic composition theorem

Theorem (First conclusion)

$$S_1 = C[\textbf{let } k = M \textbf{ in } \overline{c_1}\langle\rangle; Q_1]$$
$$S_2 = c_2(); \textbf{new } k : T; \overline{c_3}\langle\rangle; Q_2$$
$$S_{composed} = C[\textbf{let } k = M \textbf{ in } \overline{c_1'}\langle\rangle; (Q_1 \mid Q_2)]$$

1. *If S_1 preserves the secrecy of k with public variables V ($k \notin V$), then we can transfer security properties from S_2 to $S_{composed}$.*

   Let $S^\circ_{composed}$ be $S_{composed}$ with the events of S_1 removed.

   $$S^\circ_{composed} \approx^{V_1} C'[S_2]$$

   for some evaluation context C' acceptable for S_2 without public variables and for any $V_1 \subseteq V \cup (\mathrm{var}(S_1) \setminus \{k\})$.
   C' is independent of Q_2.
   Intuition: The secrecy of k allows us to replace k with a random key.

The most basic composition theorem

Theorem (Second conclusion)

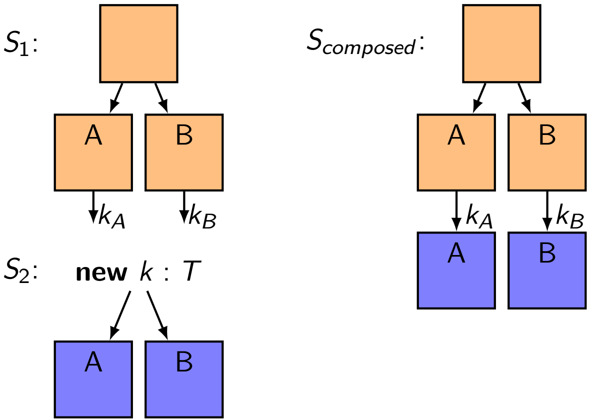
$$S_1 = C[\textbf{let } k = M \textbf{ in } \overline{c_1}\langle\rangle; Q_1]$$
$$S_2 = c_2(); \textbf{new } k : T; \overline{c_3}\langle\rangle; Q_2$$
$$S_{composed} = C[\textbf{let } k = M \textbf{ in } \overline{c_1'}\langle\rangle; (Q_1 \mid Q_2)]$$

2. *We can transfer security properties from S_1 to $S_{composed}$, provided they are proved with public variable k.*

$$S_{composed} \approx^{V'} C''[S_1]$$

for some evaluation context C'' acceptable for S_1 with public variable k and for any $V' \subseteq \mathrm{var}(S_{composed})$.
C'' contains the events of S_2.
C'' is independent of C and Q_1.

Main theorem



(S_1 may run several sessions of A and B.)

Replicating S_2

Consider:

$$S_2 = c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots \\ \textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We want to replicate S_2:

$$!^{\tilde{i} \leq \tilde{n}} c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots \\ \textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

Replicating S_2

Consider:

$$S_2 = c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots$$
$$\textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We want to replicate S_2:

$$!^{\widetilde{i} \le \widetilde{n}}\, c(); \dots c_1(y[\widetilde{i}] : T) \dots \textbf{event } e(M) \dots$$
$$\textbf{insert } T(M') \dots \textbf{get } T(z[\widetilde{i}]) \textbf{ suchthat} \dots$$

Variables implicitly with indices of replication.

Replicating S_2

Consider:

$$S_2 = c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots \\ \textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We want to replicate S_2:

$$!^{\tilde{i} \leq \tilde{n}} c[\tilde{i}](); \dots c_1[\tilde{i}](y[\tilde{i}] : T) \dots \textbf{event } e(\tilde{i}, M) \dots \\ \textbf{insert } T(\tilde{i}, M') \dots \textbf{get } T(= \tilde{i}, z[\tilde{i}]) \textbf{ suchthat} \dots$$

We could add indices to channels, events, and tables to distinguish the various sessions.

Replicating S_2

Consider:

$$S_2 = c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots \\ \textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We want to replicate S_2:

$$!^{\tilde{i} \leq \tilde{n}}\, c[\tilde{i}](); \dots c_1[\tilde{i}](y[\tilde{i}] : T) \dots \textbf{event } e(\tilde{i}, M) \dots \\ \textbf{insert } T(\tilde{i}, M') \dots \textbf{get } T(= \tilde{i}, z[\tilde{i}]) \textbf{ suchthat} \dots$$

Problem: this is not preserved by composition.
In the key exchange, partenered sessions exchange the same messages, but may not have the same replication indices.
Also in the composed system.

Replicating S_2

Consider:

$$S_2 = c(); \dots c_1(y : T) \dots \textbf{event } e(M) \dots \\ \textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We want to replicate S_2:

$$!^{\tilde{i} \leq \tilde{n}} c[\tilde{i}](x : T_{\mathrm{sid}}); \dots c_1[\tilde{i}](y[\tilde{i}] : T) \dots \textbf{event } e(x, M) \dots \\ \textbf{insert } T(x, M') \dots \textbf{get } T(= x, z[\tilde{i}]) \textbf{ suchthat} \dots$$

Partnered sessions can be determined by a session identifier computed from the messages in the protocol.
The protocol that uses the key receives the session identifier in a variable x.

Replicating S_2

Consider:

$$S_2 = c(); P$$
$$P = \dots c_1(y : T) \dots \textbf{event } e(M) \dots$$
$$\textbf{insert } T(M') \dots \textbf{get } T(z) \textbf{ suchthat} \dots$$

We replicate S_2:

$$S_{2!} = \mathrm{AddReplSid}(\tilde{i} \leq \tilde{n}, c', T_{\mathrm{sid}}, S_2) = !^{\tilde{i} \leq \tilde{n}}\, c'[\tilde{i}](x : T_{\mathrm{sid}});$$
$$\textbf{find } \tilde{u} = \tilde{i}' \leq \tilde{n} \textbf{ suchthat defined}(x[\tilde{i}'], x'[\tilde{i}'])$$
$$\wedge\, x = x[\tilde{i}'] \textbf{ then yield else}$$
$$\textbf{let } x' = \mathrm{cst} \textbf{ in } \mathrm{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\mathrm{sid}}, P)$$
$$\mathrm{AddIdxSid}(\tilde{i} \leq \tilde{n}, x : T_{\mathrm{sid}}, P) = \dots c_1[\tilde{i}](y[\tilde{i}] : T) \dots \textbf{event } e(x, M) \dots$$
$$\textbf{insert } T(x, M') \dots \textbf{get } T(= x, z[\tilde{i}]) \textbf{ suchthat} \dots$$

Never use the same session identifier twice.

Replicating S_2: transfer of security properties

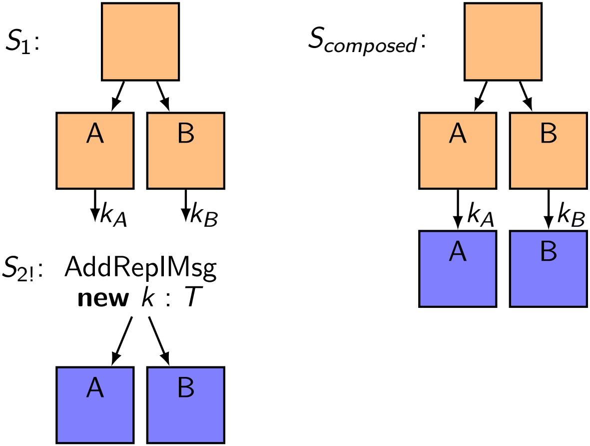
Theorem

Let $Q_! = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c', T_{\mathrm{sid}}, Q)$
and $Q'_! = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c', T_{\mathrm{sid}}, Q')$.

1. *If* Q *and* Q' *do not contain events and* $Q \approx^V Q'$, *then* $Q_! \approx^V Q'_!$.
2. *If* Q *preserves the secrecy of* y *with public variables* V, *then so does* $Q_!$.
3. *If* Q *satisfies* $\textbf{event}(e_1(y)) \Longrightarrow \textbf{event}(e_2(y))$ *with public variables* V, *then* $Q_!$ *satisfies* $\textbf{event}(e_1(x, y)) \Longrightarrow \textbf{event}(e_2(x, y))$ *with public variables* V.

   (Add a variable session identifier at the beginning of each event.)

Main composition theorem



(S_1 may run several sessions of A and B.)

Main composition theorem

Theorem (S_1 and $S_{2!}$)

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},$$
$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]$$
$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$
$$S_{2!} = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in S_2.*

C is a context with two holes, with replications $!^{\tilde{i} \leq \tilde{n}}$ above the first hole and $!^{\tilde{i}' \leq \tilde{n}'}$ above the second hole

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\tilde{i}]}\langle M_A\rangle; Q_{1A},$$

$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\tilde{i}']}\langle M_B\rangle; Q_{1B}]$$

$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$

$$S_{2!} = \mathrm{AddReplSid}(\tilde{i} \leq \tilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in S_2.*

Main composition theorem

Theorem (S_1 and $S_{2!}$)

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},$$
$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]$$
$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$
$$S_{2!} = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in S_2.*

Main con

sid is a function that takes a sequence of messages and returns a session identifier of type T_{sid}

Theorem

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},$$

$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]$$

$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$

$$S_{2!} = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in S_2.*

Main compos…

$\widetilde{msg}_A$ *is a sequence of variables defined in* C *above the first hole and input or output by* C *above the first hole or by the output* $\overline{c_A[\widetilde{i}]}\langle M_A\rangle$

Theorem (S_1 a…

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},$$

$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]$$

$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$

$$S_{2!} = \mathrm{AddReplSid}(\widetilde{i} \le \widetilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. C, Q_{1A}, Q_{1B}, Q_{2A}, *and* Q_{2B} *make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. $c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ *do not occur elsewhere in* $S_1, S_{2!}$*;*
3. S_1 *and* $S_{2!}$ *have no common variable, channel, event, table;*
4. S_1 *and* $S_{2!}$ *do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in* S_2.

Main composition theorem

Theorem (S_1 …

$\widetilde{msg}_B$ is a sequence of variables input or output by C above the second hole

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \tilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\tilde{i}]}\langle M_A\rangle; Q_{1A},$$

$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\tilde{i}']}\langle M_B\rangle; Q_{1B}]$$

$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$

$$S_{2!} = \mathrm{AddReplSid}(\tilde{i} \le \tilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain **newChannel**;*
5. *and there is no **defined** condition in S_2.*

Main composition theorem

Theorem (S_1 and $S_{2!}$)

$$S_1 = C[\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \textbf{let } k'_A = k_A \textbf{ in } \overline{c_A[\widetilde{i}]}\langle M_A\rangle; Q_{1A},$$

$$\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c_B[\widetilde{i'}]}\langle M_B\rangle; Q_{1B}]$$

$$S_2 = c_1(); \textbf{new } k : T; \overline{c_2}\langle\rangle; (Q_{2A} \mid Q_{2B})$$

$$S_{2!} = \mathrm{AddReplSid}(\widetilde{i} \leq \widetilde{n}, c'_1, T_{\mathrm{sid}}, S_2)$$

where

1. *C, Q_{1A}, Q_{1B}, Q_{2A}, and Q_{2B} make all their inputs and outputs on pairwise distinct channels with indices the current replication indices;*
2. *$c_A, c_B, c_1, c'_1, c_2, k'_A, e_A, e_B$ do not occur elsewhere in $S_1, S_{2!}$;*
3. *S_1 and $S_{2!}$ have no common variable, channel, event, table;*
4. *S_1 and $S_{2!}$ do not contain* **newChannel***;*
5. *and there is no* **defined** *condition in S_2.*

Main composition theorem

Theorem ($S_{composed}$)

Let $Q'_{2A} = \mathrm{AddIdxSid}(\widetilde{i} \leq \widetilde{n}, x : T_{\mathrm{sid}}, Q_{2A})$
and $Q'_{2B} = \mathrm{AddIdxSid}(\widetilde{i'} \leq \widetilde{n}', x : T_{\mathrm{sid}}, Q_{2B})$.
Let c'_A, c'_B *be fresh channels. Let*

$$\begin{aligned} S_{composed} = C[&\textbf{event } e_A(\mathrm{sid}(\widetilde{msg}_A), k_A, \widetilde{i}); \overline{c'_A[\widetilde{i}]}\langle M_A\rangle; \\ &\quad (Q_{1A} \mid Q'_{2A}\{k_A/k, \mathrm{sid}(\widetilde{msg}_A)/x\}), \\ &\textbf{event } e_B(\mathrm{sid}(\widetilde{msg}_B), k_B); \overline{c'_B[\widetilde{i'}]}\langle M_B\rangle; \\ &\quad (Q_{1B} \mid Q'_{2B}\{k_B/k, \mathrm{sid}(\widetilde{msg}_B)/x\})] \end{aligned}$$

Main composition theorem

Theorem (First conclusion)

1. *If S_1 satisfies*
   - *secrecy of k'_A with public variables V ($V \subseteq \mathrm{var}(S_1) \setminus \{k_A, k'_A\}$),*
   - *injective authentication of A to B:*
     $\textbf{inj-event}(e_B(sid, k)) \Longrightarrow \textbf{inj-event}(e_A(sid, k, \widetilde{i}))$
     with public variables $V \cup \{k'_A\}$,
   - *single e_A for each session identifier:*
     $\textbf{event}(e_A(sid, k_1, \widetilde{i}_1)) \wedge \textbf{event}(e_A(sid, k_2, \widetilde{i}_2)) \Longrightarrow \widetilde{i}_1 = \widetilde{i}_2$
     with public variables $V \cup \{k'_A\}$,

   then we can transfer security properties from $S_{2!}$ to $S_{composed}$.

   Let $S^\circ_{composed}$ be $S_{composed}$ with the events of S_1 removed.

$$S^\circ_{composed} \stackrel{V_1,V_2}{\overset{\sim}{\rightarrow}_f} S_{2!}$$

for some f, any $V_1 \subseteq V \cup (\mathrm{var}(S_2) \setminus \{k\})$, and $V_2 = V_1 \cap \mathrm{var}(S_2)$.

Main composition theorem

Theorem (Second conclusion)

2. *We can transfer security properties from S_1 to $S_{composed}$, provided they are proved with public variables k'_A, k_B.*

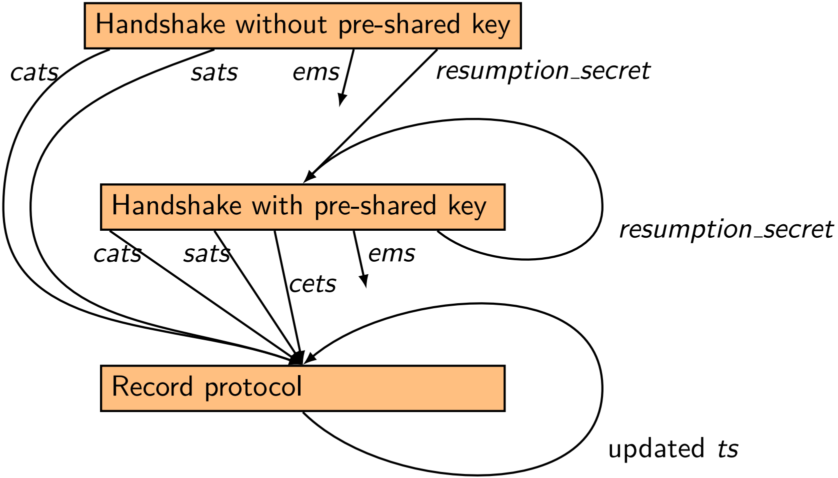
$$S_{composed} \approx_0^{V'} C'[S_1]$$

for some evaluation context C' acceptable for S_1 with public variables k'_A, k_B and any $V' \subseteq \mathrm{var}(S_{composed}) \setminus \{k'_A\}$.
C' contains the events of $S_{2!}$.
C' is independent of Q_{1A} and Q_{1B}.

Further results in the paper

- Exact security.
- New: Shared hash oracles between the key exchange and the protocol that uses the key.
- New: Variant with non-injective authentication.
- New: Variant for modified ClientHello messages.

TLS 1.3: Structure of the composition



Security of the handshake without pre-shared key

- Mutual injective authentication.
- Key secrecy: the keys
    - *cats*, *ems*, *resumption_secret* client side,
    - *sats* server side

  are secret.
- Unique accept event for each session identifier.

Security of the handshake with pre-shared key

Same properties as for the initial handshake, but

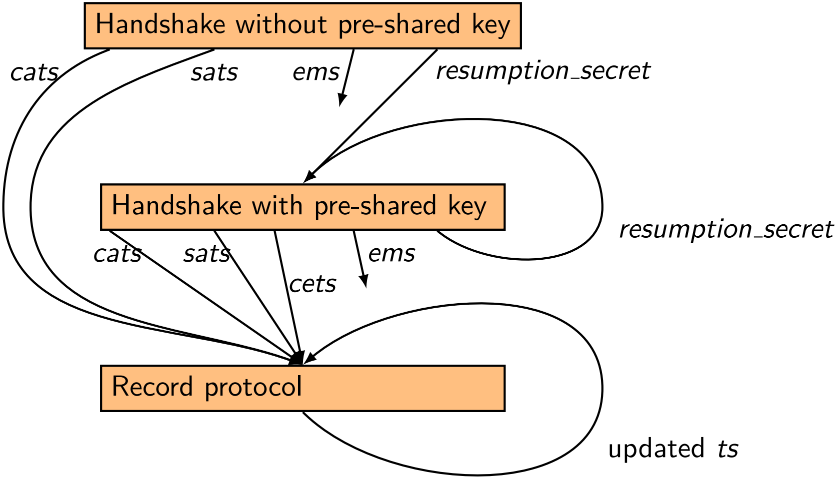
- No compromise of PSK (*resumption_secret*).
  - Limitation of CryptoVerif: cannot prove forward secrecy wrt. to the compromise of PSK for PSK-DHE.
- Weaker properties for 0-RTT:
  - The keys *cets* client side are secret.
  - If the ClientHello message received by the server has been sent by the client, then we have non-injective authentication of client to server: this session matches a session of the client with same key *cets*.
  - Otherwise,
    - If the ClientHello message has been received before, then the key *cets* computed by the server is the same as in the previous session with the same ClientHello message.
    - Otherwise, the key *cets* computed by the server is secret, independent from other keys.

Security of the record protocol

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

- Key secrecy: The updated traffic secret is secret.
- 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.
- Injective message authentication: Every time a message m is decrypted by the receiver with a counter c, the message m has been encrypted and sent by an honest sender with the same counter c.

Composition



Composition

1. We compose the record protocol with itself recursively.
    - We obtain security of the record protocol with an unbounded number of key updates.
2. We replicate that record protocol.
3. We compose the handshake with pre-shared key with the obtained record protocol, with keys *cats*, *sats*, and with weaker properties *cets*.
4. We replicate and compose the handshake with pre-shared key with itself recursively, with key *resumption_secret*.
    - We obtain security for an unbounded number of handshakes with pre-shared key.
5. We compose the handshake without pre-shared key with the record protocol, with keys *cats* and *sats*.
6. We compose the obtained handshake without pre-shared key with the obtained handshake with pre-shared key, with key *resumption_secret*.
    - We obtain security for TLS 1.3 draft 18.

Conclusion

- Composition theorems for CryptoVerif
  - computational
  - easy to apply when the protocol pieces are proved secure in CryptoVerif
  - flexible: hash oracles, injective and non-injective authentication
- Application to TLS 1.3
  - important protocol
  - would be out of scope of CryptoVerif without composition because of loops
- Applicable to other protocols

Future directions

- Composition theorems could be proved for other tools, such as EasyCrypt.
- We could automate the verification of the assumptions of our theorems and the computation of the composed protocol.
  - Automating the TLS case study would be more difficult (recursive composition).
- We could consider composition with a key exchange protocol that already uses the key.