Composition Theorems for CryptoVerif and Application to TLS 1.3

Bruno Blanchet

INRIA Paris
Bruno.Blanchet@inria.fr

June 2018
Introduction

- **Composition** between
  - a key exchange protocol
  - a protocol that uses the key

- Results stated in the CryptoVerif framework:
  - protocol verifier in the 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]
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’17]

- The composition was stated only informally.
TLS 1.3: Structure of the composition

Handshake without pre-shared key

Handshake with pre-shared key

Record protocol

**Bruno Blanchet (INRIA)**

Composition for CryptoVerif

June 2018 4 / 28
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).
Notations

- CryptoVerif represents protocols using a **process calculus**.
- \( P, Q, S \): processes
- \( C \): **context** = process with one or several holes [ ]
- \( C[P_1, P_2] \): context \( C \) with \( P_1 \) in the first hole and \( P_2 \) in the second hole
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:

$$\text{event}(e_1(x)) \implies \text{event}(e_2(x))$$

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

\[ S_1: \]
\[ S_{\text{composed}}: \]

\[ S_2: \new k : T \]

(S\textsubscript{1} may run several sessions of A and B.)
Replicating $S_2$

Consider:

$$S_2 = c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots$$

We want to replicate $S_2$:

$$!^{i \leq n} c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots$$
Replicating $S_2$

Consider:

\[ S_2 = c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots \]

We want to replicate $S_2$:

\[ !^{i \leq n} c(); \ldots c_1(y[i] : T) \ldots \text{event } e(M) \ldots \]

Variables implicitly with indices of replication.
Replicating $S_2$

Consider:

$$S_2 = c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots$$

We want to replicate $S_2$:

$$!^{i \leq n} c[i]() ; \ldots c_1[i](y[i] : T) \ldots \text{event } e(i, M) \ldots$$

We could add indices to channels and events to distinguish the various sessions.
Replicating $S_2$

Consider:

$$S_2 = c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots$$

We want to replicate $S_2$:

$$!^{i \leq n} c[i]() ; \ldots c_1[i](y[i] : T) \ldots \text{event } e(i, M) \ldots$$

Problem: this is not preserved by composition.
In the key exchange, partnered sessions exchange the same messages, but may not have the same replication indices.
Also in the composed system.
Replicating $S_2$

Consider:

$$S_2 = c(); \ldots c_1(y : T) \ldots \textbf{event } e(M) \ldots$$

We want to replicate $S_2$:

$$!^{i \leq n} c[i](x : T_{sid}); \ldots c_1[i](y[i] : T) \ldots \textbf{event } e(x, M) \ldots$$

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:

$$S_2 = c(); P$$

$$P = \ldots c_1(y : T) \ldots \text{event } e(M) \ldots$$

We replicate $S_2$:

$$S_2! = \text{AddReplSid}(i \leq n, c', T_{\text{sid}}, S_2)$$

$$= !^{i \leq n} c'[i](x : T_{\text{sid}});$$

if that value of $x$ never used before then

$$\text{AddIdxSid}(i \leq n, x : T_{\text{sid}}, P)$$

$$\text{AddIdxSid}(i \leq n, x : T_{\text{sid}}, P) = \ldots c_1[i](y[i] : T) \ldots \text{event } e(x, M) \ldots$$

Never use the same session identifier twice.
Replicating $S_2$: transfer of security properties

**Theorem**

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

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 $\text{event}(e_1(y)) \implies \text{event}(e_2(y))$ with public variables $V$, then $Q!$ satisfies $\text{event}(e_1(x, y)) \implies \text{event}(e_2(x, y))$ with public variables $V$.

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

$S_1:\hspace{1cm} S_{\text{composed}}:\n$

\begin{align*}
S_1: & \quad A \quad B \\
& \downarrow k_A \quad \downarrow k_B \\
A & \quad B
\end{align*}

\begin{align*}
S_{\text{composed}}: & \quad A \quad B \\
& \downarrow k_A \quad \downarrow k_B \\
A & \quad B
\end{align*}

$S_2:\quad \text{AddReplMsg}\,
\textbf{new} \quad k : T$

$S_1$ may run several sessions of $A$ and $B.$
Main composition theorem

Theorem \((S_1 \text{ and } S_2!)\)

\[
S_1 = C[\text{event } e_A(\text{sid}(\overline{\text{msg}}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i]⟨M_A⟩; Q_{1A}, \text{event } e_B(\text{sid}(\overline{\text{msg}}_B), k_B); c_B[i']⟨M_B⟩; Q_{1B}]
\]

\[
S_2 = c_1(); \text{new } k : T; c_2(); (Q_{2A} | Q_{2B})
\]

\[
S_{2!} = \text{AddReplSid}(i \leq n, c'_1, T_{\text{sid}}, S_2)
\]

where

1. \(S_1 \text{ and } S_{2!}\) have no common variable, channel, event;
2. Other syntactic conditions: see the paper.
Main composition theorem

\begin{align*}
S_1 &= C \{ \text{event } e_A(\text{sid}(\tilde{msg}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i](M_A); Q_{1A}, \\
&\quad \text{event } e_B(\text{sid}(\tilde{msg}_B), k_B); c_B[i'](M_B); Q_{1B} \}
\end{align*}

\begin{align*}
S_2 &= c_1(); \text{new } k : T; c_2(); (Q_{2A} \mid Q_{2B}) \\
S_2! &= \text{AddReplSid}(i \leq n, c_1', T_{\text{sid}}, S_2)
\end{align*}

where

1. \(S_1\) and \(S_2!\) have no common variable, channel, event;
2. Other syntactic conditions: see the paper.
Main composition theorem

**Theorem \((S_1 \text{ and } S_{2!})\)**

\[
S_1 = C[\text{event } e_A(\text{sid}(\overline{\text{msg}}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i](M_A); Q_{1A}, \\
\text{event } e_B(\text{sid}(\overline{\text{msg}}_B), k_B); c_B[i'](M_B); Q_{1B}]
\]

\[
S_2 = c_1(); \text{new } k : T; c_2(); (Q_{2A} \mid Q_{2B})
\]

\[
S_{2!} = \text{AddReplSid}(i \leq n, c'_1, T_{\text{sid}}, S_2)
\]

\text{where}

1. \(S_1 \text{ and } S_{2!} \) have no common variable, channel, event;
2. *Other syntactic conditions: see the paper.*
Main composition theorem

Theorem \((S_1 \text{ and } S_2!)
\)

\[ S_1 = C[\text{event } e_A(\text{sid}(\overline{msg}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i]\langle M_A \rangle; Q_{1A}, \text{event } e_B(\text{sid}(\overline{msg}_B), k_B); c_B[i']\langle M_B \rangle; Q_{1B}] \]

\[ S_2 = c_1(); \text{new } k : T; c_2(); (Q_{2A} | Q_{2B}) \]

\[ S_2! = \text{AddReplSid}(i \leq n, c'_1, T_{\text{sid}}, S_2) \]

where

1. \( S_1 \) and \( S_2! \) have no common variable, channel, event;
2. Other syntactic conditions: see the paper.

\( \text{sid is a function that takes a sequence of messages and returns a session identifier} \)
Main composition theorem

Theorem \((S_1 \text{ and } S_2!)\)

\[
S_1 = C[\text{event } e_A(\text{sid}(\sim \text{msg}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i] \langle M_A \rangle; Q_{1A}, \\
\text{event } e_B(\text{sid}(\sim \text{msg}_B), k_B); c_B[i'] \langle M_B \rangle; Q_{1B}]
\]

\[
S_2 = c_1(); \text{new } k : T; c_2(); (Q_{2A} \mid Q_{2B})
\]

\[
S_2! = \text{AddReplSid}(i \leq n, c'_1, T_{\text{sid}}, S_2)
\]

where

1. \(S_1 \text{ and } S_2!\) have no common variable, channel, event;
2. Other syntactic conditions: see the paper.

\(\sim \text{msg}_A\) is a sequence of variables input or output by \(C\) above the first hole.
Main composition theorem

Theorem ($S_1$ and $S_2!$)

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

$\text{event } e_B(\text{sid}(\tilde{\text{msg}}_B), k_B); c_B[i'](M_B); Q_{1B}]$

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

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

where

1. $S_1$ and $S_{2!}$ have no common variable, channel, event;
2. Other syntactic conditions: see the paper.
Main composition theorem

**Theorem (S₁ and S₂!)**

\[
S₁ = C[\text{event } e_A(\text{sid}(\tilde{msg}_A), k_A, i); \text{let } k'_A = k_A \text{ in } c_A[i](M_A); Q₁_A, \\
\text{event } e_B(\text{sid}(\tilde{msg}_B), k_B); c_B[i'](M_B); Q₁_B]
\]

\[
S₂ = c_1(); \text{new } k : T; c₂(); (Q₂_A \mid Q₂_B)
\]

\[
S₂! = \text{AddReplSid}(i \leq n, c'₁, T\text{sid}, S₂)
\]

where

1. \( S₁ \) and \( S₂! \) have no common variable, channel, event;
2. Other syntactic conditions: see the paper.
Main composition theorem

Theorem ($S_{\text{composed}}$)

Let $Q'_2A = \text{AddIdxSid}(i \leq n, x : T_{\text{sid}}, Q_{2A})$
and $Q'_2B = \text{AddIdxSid}(i' \leq n', x : T_{\text{sid}}, Q_{2B})$.

\[
S_{\text{composed}} = C[\text{event } e_A(\text{sid}(...), k_A, i); c_A[i](M_A); (Q_{1A} \mid Q'_2A{\{k_A/}k, \text{sid}(\text{msg}_A)/x})],
\text{event } e_B(\text{sid}(\text{msg}_B), k_B); c_B[i'](M_B); (Q_{1B} \mid Q'_2B{\{k_B/}k, \text{sid}(\text{msg}_B)/x})]
\]
Main composition theorem

Theorem (First conclusion)

If $S_1$ satisfies

- secrecy of $k'_A$ with public variables $V$ ($V \subseteq \text{var}(S_1) \setminus \{k_A, k'_A\}$),
- injective authentication of $A$ to $B$:
  \[\text{inj-event}(e_B(sid, k)) \implies \text{inj-event}(e_A(sid, k, i))\]
  with public variables $V \cup \{k'_A\}$,
- single $e_A$ for each session identifier:
  \[\text{event}(e_A(sid, k_1, i_1)) \land \text{event}(e_A(sid, k_2, i_2)) \implies i_1 = i_2\]
  with public variables $V \cup \{k'_A\}$,

then we can transfer security properties from $S_2!$ to $S_{\text{composed}}$.

Up to renumbering of variable indices,

$S_{\text{composed}}$ with the events of $S_1$ removed

is indistinguishable with public variables $V \cup (\text{var}(S_2) \setminus \{k\})$

from an adversary interacting with $S_2!$. 
We can transfer security properties from $S_1$ to $S_{\text{composed}}$, provided they are proved with public variables $k'_A, k_B$.

$S_{\text{composed}}$

is indistinguishable with public variables \( \text{var}(S_{\text{composed}}) \setminus \{k'_A\} \)

from an adversary interacting with $S_1$ with access to $k'_A, k_B$. 
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.
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