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]
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.
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:

\[
C ::= \\
[] \quad \text{hole} \\
\text{newChannel } c; C \quad \text{channel restriction} \\
Q | C \quad \text{parallel composition} \\
C | Q \quad \text{parallel composition}
\]
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:

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

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

\[ S_1: \quad k \text{ (secret)} \]

\[ S_2: \quad \text{new} \quad k : T \]

\[ S_{\text{composed}}: \quad k \]

Bruno Blanchet (INRIA)
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

\[
S_1 = C[\text{let } k = M \text{ in } \overline{c_1}\langle\rangle; Q_1]
\]

\[
S_2 = c_2(); \text{new } k : T; \overline{c_3}\langle\rangle; Q_2
\]

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

\[
S_{\text{composed}} = C[\text{let } k = M \text{ in } \overline{c'_1}\langle\rangle; (Q_1 | Q_2)]
\]
The most basic composition theorem

Theorem (First conclusion)

\[ S_1 = C[\text{let } k = M \text{ in } \overline{c_1}(); Q_1] \]
\[ S_2 = c_2(); \text{new } k : T; \overline{c_3}(); Q_2 \]
\[ S_{\text{composed}} = C[\text{let } k = M \text{ in } \overline{c_1}(); (Q_1 \mid Q_2)] \]

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 \( S_{\text{composed}} \).

Let \( S_{\text{composed}}^\circ \) be \( S_{\text{composed}} \) with the events of \( S_1 \) removed.

\[ S_{\text{composed}}^\circ \approx_{V_1} C'[S_2] \]

for some evaluation context \( C' \) acceptable for \( S_2 \) without public variables and for any \( V_1 \subseteq V \cup (\text{var}(S_1) \setminus \{k\}) \).

\( C' \) 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)**

\[
S_1 = C[\text{let } k = M \text{ in } \overline{c_1}\langle \rangle; Q_1]
\]

\[
S_2 = c_2(); \text{new } k : T; \overline{c_3}\langle \rangle; Q_2
\]

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

2. We can transfer security properties from \(S_1\) to \(S_{\text{composed}}\), provided they are proved with public variable \(k\).

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

for some evaluation context \(C''\) acceptable for \(S_1\) with public variable \(k\) and for any \(V' \subseteq \text{var}(S_{\text{composed}})\).

\(C''\) contains the events of \(S_2\).

\(C''\) is independent of \(C\) and \(Q_1\).
Main theorem

\( S_1: \)

\[ \text{A} \quad \text{B} \]

\[ k_A \quad k_B \]

\( S_2: \textbf{new} \ k : T \)

\[ \text{A} \quad \text{B} \]

\( S_{\text{composed}}: \)

\[ \text{A} \quad \text{B} \]

\[ k_A \quad k_B \]

\((S_1 \text{ may run several sessions of } A \text{ and } B.\)
Consider:

\[ S_2 = \text{c() \ldots c}_1(y : T) \ldots \text{event } e(M) \ldots \]

\[ \text{insert } T(M') \ldots \text{get } T(z) \text{ such that} \ldots \]

We want to replicate \( S_2 \):

\[ \text{!}\tilde{i} \leq \tilde{n} \text{ c() \ldots c}_1(y : T) \ldots \text{event } e(M) \ldots \]

\[ \text{insert } T(M') \ldots \text{get } T(z) \text{ such that} \ldots \]
Replicating $S_2$

Consider:

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

$$\textbf{insert } T(M') \ldots \textbf{get } T(z) \textbf{ such that } \ldots$$

We want to replicate $S_2$:

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

$$\textbf{insert } T(M') \ldots \textbf{get } T(z[i]) \textbf{ such that } \ldots$$

Variables implicitly with indices of replication.
Replicating $S_2$

Consider:

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

$$\textbf{insert } T(M') \ldots \textbf{get } T(z) \textbf{ suchthat } \ldots$$

We want to replicate $S_2$:

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

$$\textbf{insert } T(\tilde{i}, M') \ldots \textbf{get } T(= \tilde{i}, z[\tilde{i}]) \textbf{ suchthat } \ldots$$

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

Consider:

\[
S_2 = c() \ldots c_1(y : T) \ldots \text{event } e(M) \ldots \\
\text{insert } T(M') \ldots \text{get } T(z) \text{ suchthat } \ldots 
\]

We want to replicate $S_2$:

\[
!i \leq \tilde{n} \ c[\tilde{i}]() \ldots c_1[\tilde{i}](y[\tilde{i}] : T) \ldots \text{event } e(\tilde{i}, M) \ldots \\
\text{insert } T(\tilde{i}, M') \ldots \text{get } T(= \tilde{i}, z[\tilde{i}]) \text{ suchthat } \ldots 
\]

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:

\[ S_2 = c(); \ldots c_1(y : T) \ldots \text{event } e(M) \ldots \]
\[ \text{insert } T(M') \ldots \text{get } T(z) \text{ such that } \ldots \]

We want to replicate $S_2$:

\[ !\overset{i \leq n}{\bar{c}[i]}(x : T_{\text{sid}}); \ldots c_1[\bar{i}](y[\bar{i}] : T) \ldots \text{event } e(x, M) \ldots \]
\[ \text{insert } T(x, M') \ldots \text{get } T(= x, z[\bar{i}]) \text{ such that } \ldots \]

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:

\[ S_2 = c(); P \]
\[ P = \ldots c_1(y : T) \ldots \text{event } e(M) \ldots \]
\[ \text{insert } T(M') \ldots \text{get } T(z) \text{ suchthat } \ldots \]

We replicate $S_2$:

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

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

**Theorem**

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

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

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

\( S_1 : \)  

\( \text{AddReplMsg new } k : T \)

\( S_{\text{composed}} : \)

\( (S_1 \text{ may run several sessions of } A \text{ and } B. ) \)
Main composition theorem

**Theorem (S$_1$ and S$_2!$)**

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

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

\[
S_2! = \text{AddReplSid}(\tilde{i} \leq \tilde{n}, c_1', T_{\text{sid}}, S_2)
\]

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 \textbf{newChannel};
5. and there is no \textbf{defined} condition in S$_2$. 
C is a context with two holes, with replications \( !^{i \leq \tilde{n}} \) above the first hole and \( !^{i' \leq \tilde{n}'} \) above the second hole

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

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 \textbf{newChannel};
5. and there is no \textbf{defined} condition in S_2.
Main composition theorem

Theorem ($S_1$ and $S_2!$)

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

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 \textbf{newChannel};
5. and there is no \textbf{defined} condition in $S_2$. 

Bruno Blanchet (INRIA)
Main composition theorem

Theorem ($S_1$ and $S_2!$)

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

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

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

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

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 $S_2$.

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

Bruno Blanchet (INRIA)
Theorem \((S_1, S_2)\)

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

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

\[
S_2 = c_1(); \text{new } k : T; c'_2(); (Q_{2A} \mid Q_{2B})
\]

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

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 \textbf{newChannel};
5. and there is no \textbf{defined} condition in \(S_2.\)
Main composition theorem

Theorem \( (S_1, S_2!) \)

\[
S_1 = C[event e_A(sid(msg_A), k_A, l); let k_A = k'_A in c_A[i](M_A); Q_1A, \\
\text{event } e_B(sid(msg_B), k_B); c_B[i'](M_B); Q_1B] \\
S_2 = c_1(); \text{new } k : T; c_2(); (Q_2A | Q_2B) \\
S_2! = \text{AddReplSid}(i \leq \tilde{n}, c_1', T_{\text{sid}}, S_2)
\]

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 \text{newChannel};
5. and there is no defined condition in \( S_2 \).

\( \tilde{msg}_B \) is a sequence of variables input or output by \( C \) above the second hole.
Main composition theorem

Theorem ($S_1$ and $S_2!$)

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

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 $\text{newChannel}$;
5. and there is no defined condition in $S_2$. 
Main composition theorem

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

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

\[
S_{\text{composed}} = C[\text{event } e_A(\text{sid}(\overline{\text{msg}_A}), k_A, \tilde{i}); c'_A[\tilde{i}][M_A];
(Q_{1A} \mid Q'_{2A}\{k_A/k, \text{sid}(\overline{\text{msg}_A})/x\})),
\]

\[
\text{event } e_B(\text{sid}(\overline{\text{msg}_B}), k_B); c'_B[\tilde{i}'][M_B];
(Q_{1B} \mid Q'_{2B}\{k_B/k, \text{sid}(\overline{\text{msg}_B})/x\}])
\]
Main composition theorem

Theorem (First conclusion)

1. 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 $B$:
     \[
     \text{inj-event}(e_B(sid, k)) \implies \text{inj-event}(e_A(sid, k, \tilde{i}))
     \]
     with public variables $V \cup \{k'_A\}$,
   - single $e_A$ for each session identifier:
     \[
     \text{event}(e_A(sid, k_1, \tilde{i}_1)) \land \text{event}(e_A(sid, k_2, \tilde{i}_2)) \implies \tilde{i}_1 = \tilde{i}_2
     \]
     with public variables $V \cup \{k'_A\}$,

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

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

\[
S_{\text{composed}}^\circ \nRightarrow_f V_1, V_2 S_2!
\]

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

Theorem (Second conclusion)

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

$$S_{\text{composed}} \approx_{0}^{V'} C'[S_1]$$

for some evaluation context $C'$ acceptable for $S_1$ with public variables $k'_A, k_B$ and any $V' \subseteq \text{var}(S_{\text{composed}}) \setminus \{k'_A\}$.

$C'$ 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

- Handshake without pre-shared key
  - cats
  - sats
  - ems
  - resumption_secret

- Handshake with pre-shared key
  - cats
  - sats
  - ems
  - cets

- Record protocol
  - updated ts
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

Handshake without pre-shared key

cats sats ems resumption_secret

Handshake with pre-shared key

cats sats ems cets

Record protocol

updated ts
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 \textit{cats}, \textit{sats}, and with weaker properties \textit{cets}.

4. We replicate and compose the handshake with pre-shared key with itself recursively, with key \textit{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 \textit{cats} and \textit{sats}.

6. We compose the obtained handshake without pre-shared key with the obtained handshake with pre-shared key, with key \textit{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.