Safely composing security protocols via tagging

Stéphanie Delaune

LSV, ENS Cachan & CNRS & INRIA project SECSI

March, 14, 2008

→ joint work with Véronique Cortier, Jérémie Delaitre, Myrto Arapinis and Steve Kremer
Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature, . . . . . )
Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy)
- use cryptographic primitives (e.g. encryption, signature, . . . . . . )

The network is unsecure!

Communications take place over a public network like the Internet.
Messages are abstracted by terms

- pairing \( \langle m_1, m_2 \rangle \),
- symmetric encryption \( \text{enc}(m, k) \) and public key encryption \( \text{enca}(m, \text{pub}(A)) \),
- signature \( \text{sign}(m, \text{priv}(A)) \).
Cryptographic protocols (symbolic approach)

Messages are abstracted by terms

- pairing \( \langle m_1, m_2 \rangle \),
- symmetric \( \text{enc}(m, k) \) and public key encryption \( \text{enca}(m, \text{pub}(A)) \),
- signature \( \text{sign}(m, \text{priv}(A)) \).

Presence of an idealized attacker

- may read, intercept and send messages,
- may build new messages following deduction rules (symbolic manipulation on terms).
Cryptographic protocols (symbolic approach)

Messages are abstracted by terms

- pairing $\langle m_1, m_2 \rangle$,
- symmetric $\text{enc}(m, k)$ and public key encryption $\text{enca}(m, \text{pub}(A))$,
- signature $\text{sign}(m, \text{priv}(A))$.

Presence of an idealized attacker

- may read, intercept and send messages,
- may build new messages following deduction rules (symbolic manipulation on terms).

Examples:

\[
\begin{align*}
\text{enc}(m, k) & \quad k \\
\text{enc}(m, k) & \quad m \\
\text{enca}(m, \text{pub}(a)) & \quad \text{priv}(a)
\end{align*}
\]
Motivations

Formal verification of security protocols

- Existing tools allow us to verify \textit{relatively small} protocols and sometimes only for a \textit{bounded number of sessions}
- Most often, we verify them in \textit{isolation}
  \hspace{1cm} \rightarrow \text{this is not sufficient}
Motivations

Formal verification of security protocols

- Existing tools allow us to verify \textit{relatively small} protocols and sometimes only for a \textit{bounded number of sessions}
- Most often, we verify them in \textit{isolation} → this is not sufficient

Example:

\[ P_1 : A \rightarrow B : \text{enca}(s, \text{pub}(B)) \]

Question: What about the secrecy of \( s \)?
Motivations

Formal verification of security protocols

- Existing tools allow us to verify relatively small protocols and sometimes only for a bounded number of sessions
- Most often, we verify them in isolation → this is not sufficient

Example:

\[ P_1 : \ A \rightarrow B : \ \text{enca}(s, \text{pub}(B)) \quad P_2 : \ A \rightarrow B : \ \text{enca}(N_a, \text{pub}(B)) \]
\[ B \rightarrow A : \ N_a \]

Question: What about the secrecy of \( s \)?
Motivations

Formal verification of security protocols

- Existing tools allow us to verify relatively small protocols and sometimes only for a bounded number of sessions
- Most often, we verify them in isolation → this is not sufficient

Our goal

investigate sufficient conditions to ensure that protocols can be safely used in an environment where:

1. other sessions of the same protocol may be executed;
2. other sessions of another protocol may be executed as well.

→ protocols may share identities and keys (e.g. public keys, long-term symmetric keys)
Outline of the talk

1. Introduction

2. Composition result I: “from one session to many”

3. Composition result II: “from one protocol to many”

4. Conclusion
Outline of the talk

1. Introduction
2. Composition result I: “from one session to many”
3. Composition result II: “from one protocol to many”
4. Conclusion
Our goal:
compose different sessions from the same protocol

→ well-known fact: an attack may involve an arbitrary number of sessions
Our goal:

compose different sessions from the same protocol

→ well-known fact: an attack may involve an arbitrary number of sessions

Solution

• a transformation which maps a protocol $P$ that is secure for a single session to a protocol $\overline{P}$ that is secure for an unbounded number of sessions.

• side-effect: an effective strategy to design secure protocols
Let $P$ be a protocol with $\ell$ participants as given below:

\[
\begin{align*}
A_{i_1} & \to A_{j_1} : m_1 \\
A_{i_2} & \to A_{j_2} : m_2 \\
& \quad \vdots \\
A_{i_k} & \to A_{j_k} : m_k
\end{align*}
\]
Our transformation

The protocol $\overline{P}$ (with $\ell$ participants) is described below:

**Initialisation phase:** broadcast of fresh nonces

- $A_1 \rightarrow All : A_1, N_1$
- $A_2 \rightarrow All : A_2, N_2$
- ...
- $A_\ell \rightarrow All : A_\ell, N_\ell$
Our transformation

The protocol $\mathcal{P}$ (with $\ell$ participants) is described below:

**Initialisation phase**: broadcast of fresh nonces

$$
A_1 \rightarrow \text{All} : A_1, N_1 \\
A_2 \rightarrow \text{All} : A_2, N_2 \\
\vdots
\\
A_\ell \rightarrow \text{All} : A_\ell, N_\ell
$$

Every participant obtain a tag $= \langle A_1, N_1, A_2, N_2, \ldots, A_\ell, N_\ell \rangle$
Our transformation

The protocol $P$ (with $\ell$ participants) is described below:

**Initialisation phase:** broadcast of fresh nonces

$$A_1 \rightarrow All : A_1, N_1$$
$$A_2 \rightarrow All : A_2, N_2$$
$$\vdots$$
$$A_\ell \rightarrow All : A_\ell, N_\ell$$

Every participant obtain a $\text{tag} = \langle A_1, N_1, A_2, N_2, \ldots, A_\ell, N_\ell \rangle$

**Main phase:**

where the function $\overline{m}$ is defined by:

$$A_{i_1} \rightarrow A_{j_1} : \overline{m_1}$$
$$A_{i_2} \rightarrow A_{j_2} : \overline{m_2}$$
$$\vdots$$
$$A_{i_k} \rightarrow A_{j_k} : \overline{m_k}$$

$$\begin{cases}
\langle u_1, u_2 \rangle \rightarrow \langle \overline{u_1}, \overline{u_2} \rangle \\
\text{f}(u_1, u_2) \rightarrow \text{f}(\langle \text{tag}, \overline{u_1} \rangle, \overline{u_2}) \\
\overline{u} \rightarrow u
\end{cases}$$

when $f \in \{\text{enc, enca, sign}\}$

otherwise
Theorem

Let $P$ be a protocol with no critical long-term keys in plaintext position.

If $P$ preserves the secrecy of $s$ for a single honest session of each role then $\overline{P}$ preserves the secrecy of $s$ for an unbounded number of sessions.

- critical long-term keys do not appear in plaintext
  → this can be easily checked on the finite specification of the protocol
  → often satisfied since it is considered as a prudent practice

- single honest session of each role
  → i.e. one an instance of each role (in general 2 or 3);
  → participants engaged in this session are honest.
Composition result “from one session to many”

Theorem

Let $P$ be a protocol with no critical long-term keys in plaintext position.

If $P$ preserves the secrecy of $s$ for a single honest session of each role then $\overline{P}$ preserves the secrecy of $s$ for an unbounded number of sessions.

- **critical long-term keys** do not appear in plaintext
  - this can be easily checked on the finite specification of the protocol
  - often satisfied since it is considered as a prudent practice

- **single honest session** of each role
  - i.e. one an instance of each role (in general 2 or 3);
  - participants engaged in this session are honest.

**Exemple:** Needham-Schroeder public key protocol
- the Lowe’s famous man-in-the-middle attack is prevented
Related work

Computational models

Several compilers already exist in the area of cryptographic design, e.g.

- *Scalable protocols for authenticated group key exchange* [Katz & Yung, 03]

Symbolic models

- *Synthesizing secure protocols* [Cortier et al., 07]
- *How to guarantee secrecy for cryptographic protocols* [Beauquier & Gauche, 07]

→ the transformations make heavy use of cryptography
Outline of the talk

1. Introduction

2. Composition result I: “from one session to many”

3. Composition result II: “from one protocol to many”

4. Conclusion
Summary: “from one protocol to many”

Our goal:
compose sessions coming from different protocols

Solution
we propose sufficient and rather tight conditions for a protocol to be safely used in an environment where other protocols may be executed as well;
Summary: “from one protocol to many”

Our goal:
compose sessions coming from different protocols

Solution
we propose sufficient and rather tight conditions for a protocol to be safely used in an environment where other protocols may be executed as well;

Example: (given in introduction)

\[ P_1 : A \rightarrow B : \text{enca}(s, \text{pub}(B)) \]
\[ P_2 : A \rightarrow B : \text{enca}(N_a, \text{pub}(B)) \]
\[ B \rightarrow A : N_a \]

\[ \rightarrow \] protocols may share identities and keys (e.g. public keys, long-term symmetric keys)
Main condition - Tagging

Well-tagged protocol

Each protocol is given an **identifier** (*e.g.* the protocol’s name). This identifier has to appear in any **encrypted** and **signed** message.

→ this **tagging policy** will avoid interaction between two different protocols.
Well-tagged protocol

Each protocol is given an identifier (e.g. the protocol’s name). This identifier has to appear in any encrypted and signed message.

→ this tagging policy will avoid interaction between two different protocols.

Example: \( P_1 \) is 1-tagged whereas \( P_2 \) is 2-tagged

Protocol \( P_1 \)

\[ A \rightarrow B : \text{enca}(\langle 1, s \rangle, \text{pub}(B)) \]

Protocol \( P_2 \)

\[ A \rightarrow B : \text{enca}(\langle 2, N_a \rangle, \text{pub}(B)) \]

\[ B \rightarrow A : N_a \]
Theorem

Let $P_1$ and $P_2$ be two well-tagged protocols such that

- no critical long-term keys appear in plaintext position neither in $P_1$ nor in $P_2$,
- $P_1$ is $\alpha$-tagged and $P_2$ is $\beta$-tagged with $\alpha \neq \beta$.

If $P_1$ preserves the secrecy of $s$ then $P_1 \parallel P_2$ preserves the secrecy of $s$. 
Theorem

Let $P_1$ and $P_2$ be two well-tagged protocols such that

- no critical long-term keys appear in plaintext position neither in $P_1$ nor in $P_2$,
- $P_1$ is $\alpha$-tagged and $P_2$ is $\beta$-tagged with $\alpha \neq \beta$.

If $P_1$ preserves the secrecy of $s$ then $P_1 | P_2$ preserves the secrecy of $s$.

Extensions that have been already done:

1. well-tagged condition can be relaxed: disjoint encryption is actually sufficient;
2. composition result holds for a class of security properties (secrecy, authentication, . . .)
The idea of adding an identifier is not novel:

*Principle 10 in the prudent engineering paper*

[Abadi & Needham, 95]
Related work

The idea of adding an identifier is not novel:

*Principle 10 in the prudent engineering paper*

[Abadi & Needham, 95]

There are also some formal results about this problem:

- *Protocol independence through disjoint encryption*  
  [Guttman & Thayer, 00]  
  → their condition has to hold on any valid execution of the protocol

- *Sufficient conditions for composing security protocols*  
  [Andova et al., 07]  
  → they have to assume typing hypothesis, they can not deal with protocols with ciphertext forwarding
Outline of the talk

1. Introduction

2. Composition result I: “from one session to many”

3. Composition result II: “from one protocol to many”

4. Conclusion
Conclusion

Two composition results

1. one that is useful to compose sessions coming from the same protocol
   \[\rightarrow\] this can be obtained with dynamic tags

2. one that can be used to compose protocols that satisfy disjoint encryption
   \[\rightarrow\] this can be obtained with static tags

\[\rightarrow\] to combine both results, use \(tag = \langle id_\alpha, A_1, N_1, \ldots, A_\ell, N_\ell\rangle\).
Conclusion

Two composition results

1. one that is useful to compose sessions coming from the same protocol
   → this can be obtained with dynamic tags
2. one that can be used to compose protocols that satisfy disjoint encryption
   → this can be obtained with static tags

→ to combine both results, use $\text{tag} = \langle id_\alpha, A_1, N_1, \ldots, A_\ell, N_\ell \rangle$.

Future Work

- obtain a more fine-grained characterization of a decidable class (for an unbounded number of sessions and a class security properties)
- other kind of security properties (e.g. equivalence-based properties)
- other kind of composition (e.g. sequence)