Secure composition of security protocols

Workshop on the 20th Anniversary of LSV, May 11th, 2017

Véronique Cortier

LORIA - CNRS, INRIA, Université de Lorraine
Context: cryptographic protocols

Cryptographic protocols are widely used in everyday life.

→ They aim at securing communications over public or insecure networks.
How does this work?

**Protocol**: describes how each participant should behave in order to get e.g. a common key.

**Cryptography**: makes use of cryptographic primitives

- encryption
- signature
- hash
- ...
Example: HTTPS

- Various implementations: OpenSSL, SecureTransport, JSSE, ...
- Lots of bugs and attacks, with fixes every month

Go to fail

Missing checks (MACs, signatures, ...)

Memory overflow
What the issue?

Dishonest users may:

- read or even intercept sent messages,
- forge and send messages,
- be legitimate participants of some protocols.
Need for formal guarantees

Many flaws are discovered afterwards
- Difficult and costly to fix
- Old (insecure) versions remain
Need for formal guarantees

Many flaws are discovered afterwards
- Difficult and costly to fix
- Old (insecure) versions remain

Formal methods can help:
- Provide rigorous models and proofs
- Automatically find flaws
Messages

Messages are abstracted by terms.

Agents: \(a, b, \ldots\)  
Nonces: \(n_1, n_2, \ldots\)

Keys: \(k_1, k_2, \ldots\)

Cyphertext: \(\text{enc}(m, k)\)  
Concatenation: \(\text{pair}(m_1, m_2)\)

Example: The message \(\{A, N_a\}_K\) is represented by:

\[
\text{enc}(\text{pair}(A, N_a), K)
\]

Intuition: only the structure of the message is kept.
Equational theory

\[ \text{dec}(\text{enc}(x, y), y) = x \]

\[ \pi_1(\langle x, y \rangle) = x \]

\[ \pi_2(\langle x, y \rangle) = y \]

\[ \text{deca}(\text{enca}(x, \text{pub}(y)), y) = x \]

**Definition (Equational theory)**

An equational theory \( \equiv_E \) is a relation on terms

- closed under substitutions
  \[ u \equiv_E v \Rightarrow u\sigma \equiv_E v\sigma \]

- closed by context
  \[ u_1 \equiv_E v_1, \ldots, u_n \equiv_E v_n \Rightarrow f(u_1, \ldots, u_n) \equiv_E f(v_1, \ldots, v_n) \]
Other examples of theories

**EXclusive Or**

\[
x \oplus (y \oplus z) = (x \oplus y) \oplus z \\
x \oplus x = 0 \\
x \oplus 0 = x
\]

**Diffie-Hellmann**

\[
\exp(\exp(z, x), y) = \exp(\exp(z, y), x)
\]
A simple protocol

\[ \langle \text{Bob, } k \rangle \]

\[ \langle \text{Alice, } \text{enc}(s, k) \rangle \]
A simple protocol

⟨Bob, k⟩ → ⟨Alice, enc(s, k)⟩

Question?

Does the protocol $P$ leak the secret $s$?
A simple protocol

Answer: Of course, Yes!

\[ \langle \text{Alice, enc}(s, k) \rangle \rightarrow \langle \text{Bob, k} \rangle \]

\[ \langle \text{Bob, k} \rangle \rightarrow \langle \text{Alice, enc}(s, k) \rangle \]

Notation: \( \langle \text{Alice, enc}(s, k) \rangle, \langle \text{Bob, k} \rangle \vdash s \)
Model for protocols

Protocols modelled in a process calculus, e.g. the applied pi calculus

\[ P ::= \]

\[ 0 \]

\[ \text{in}(c, x).P \quad \text{input} \]

\[ \text{out}(c, t).P \quad \text{output} \]

\[ \text{if } t_1 = t_2 \text{ then } P \text{ else } Q \quad \text{conditional} \]

\[ P \parallel Q \quad \text{parallel} \]

\[ !P \quad \text{replication} \]

\[ \text{new } n.P \quad \text{restriction} \]

Specificities:

- messages are terms (not just names as in the pi calculus)
- equality in conditionals interpreted modulo an equational theory
Example: Needham-Schroeder protocol

\[ A \rightarrow B : \{ A, N_a \}_{\text{pub}(B)} \]
\[ B \rightarrow A : \{ N_a, N_b \}_{\text{pub}(A)} \]
\[ A \rightarrow B : \{ N_b \}_{\text{pub}(B)} \]

\( N_a \) random number generated by \( A \).
\( N_b \) random number generated by \( B \).
Example: Needham-Schroeder protocol

\[ A \rightarrow B : \{A, N_a\}_{\text{pub}(B)} \]
\[ B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \]
\[ A \rightarrow B : \{N_b\}_{\text{pub}(B)} \]

- \(N_a\) random number generated by \(A\).
- \(N_b\) random number generated by \(B\).

We need to model two processes:
- one corresponding to the role of \(A\)
- one corresponding to the role of \(B\)
Protocols

Role of A

\[ A \rightarrow B : \{ A, N_a \}_{\text{pub}(B)} \]
\[ B \rightarrow A : \{ N_a, N_b \}_{\text{pub}(A)} \]
\[ A \rightarrow B : \{ N_b \}_{\text{pub}(B)} \]

\[ P_A(\text{priv}_A, \text{pub}_B, A, B) := \]
\[ \nu N_A. \]
\[ \text{out}(c, \text{enca}(\text{pair}(A, N_A), \text{pub}_B)). \]
Protocols

Role of A

\[

tag{1} A \rightarrow B : \{A, N_a\}_{\text{pub}(B)} \\
tag{2} B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \\
tag{3} A \rightarrow B : \{N_b\}_{\text{pub}(B)}
\]

\[
P_A(\text{priv}_A, \text{pub}_B, A, B) := \\
\nu N_A. \tag{1} \\
\nu\text{out}(c, \text{enca}(\text{pair}(A, N_A), \text{pub}_B)). \tag{2} \\
\nu\text{in}(c, x). \tag{3} \\
\quad \text{let } z = \text{deca}(x, \text{priv}_A) \text{ in} \tag{4} \\
\quad \quad \text{if } N_A = \pi_1(z) \text{ then let } y = \pi_2(z) \text{ in} \tag{5} \\
\quad \quad \quad \text{out}(c, \text{enca}(y, \text{pub}_B)) \tag{6}
\]
Complete process

Then, the complete process representing the Needham-Schroeder protocol is:

$$P :=$$

$$!P_A(\text{priv}_A, \text{pub(\text{priv}_B)}, A, B)$$

$$!P_B(\text{priv}_B, \text{pub(\text{priv}_A)}, A, B)$$
Then, the complete process representing the Needham-Schroeder protocol is:

\[
P := \\
\nu \text{priv}_A.\nu \text{priv}_B. \\
\text{out}(c, \text{pub}(\text{priv}_A)).\text{out}(c, \text{pub}(\text{priv}_B)). \\
!P_A(\text{priv}_A, \text{pub}(\text{priv}_B), A, B) \\
!P_B(\text{priv}_B, \text{pub}(\text{priv}_A), A, B)
\]
Complete process

Then, the complete process representing the Needham-Schroeder protocol is:

\[
P :=
\]
\[
\nu_{\text{priv}_A} \nu_{\text{priv}_B} \nu_{\text{priv}_C}.
\]
\[
\text{out}(c, \text{pub}(\text{priv}_A)) \cdot \text{out}(c, \text{pub}(\text{priv}_B)) \cdot \text{out}(c, \text{priv}_C).
\]
\[
!P_A(\text{priv}_A, \text{pub}(\text{priv}_B), A, B)
\]
\[
!P_B(\text{priv}_B, \text{pub}(\text{priv}_A), A, B)
\]
\[
!P_A(\text{priv}_A, \text{pub}(\text{priv}_C), A, C)
\]
\[
!P_B(\text{priv}_B, \text{pub}(\text{priv}_C), A, C)
\]
Complete process

Then, the complete process representing the Needham-Schroeder protocol is:

\[ P := \]
\[ \nu_{\text{priv}_A} \cdot \nu_{\text{priv}_B} \cdot \nu_{\text{priv}_C}. \]
\[ \text{out}(c, \text{pub}(\text{priv}_A)) \cdot \text{out}(c, \text{pub}(\text{priv}_B)) \cdot \text{out}(c, \text{priv}_C). \]
\[ !P_A(\text{priv}_A, \text{pub}(\text{priv}_B), A, B) \]
\[ !P_B(\text{priv}_B, \text{pub}(\text{priv}_A), A, B) \]
\[ !P_A(\text{priv}_A, \text{pub}(\text{priv}_C), A, C) \]
\[ !P_B(\text{priv}_B, \text{pub}(\text{priv}_C), A, C) \]

and also

\[ !(P_A(\text{priv}_B, \text{pub}(\text{priv}_A), B, A)) \]
\[ !(P_A(\text{priv}_B, \text{pub}(\text{priv}_C), B, C)) \]
\[ \ldots \]
Security properties

\[ \forall A \quad P \mid A \models \phi \]

\(\phi\) : trace property

“something bad never occurs on any execution trace of \(P\)”
Formal Model

Composition

Secure channels

Conclusion

Properties

Security properties

\[ \forall A \quad P | A \models \phi \]

\[ \phi : \text{trace property} \]

“something bad never occurs on any execution trace of \( P \)”

Examples:

- **secrecy** of \( s \): \[ \forall A \quad P | A \vdash s \]
- **authentication**: we need a small logic
How to check security properties?

∀A  P | A |= \phi

A lot of existing results for trace properties:

- several procedures to deal with a variety of cryptographic primitives, e.g. encryption, signature, exclusive or, 
  ...
- several automatic tools
  e.g., ProVerif, AVISPA, Scyther, Tamarin ...
A fairy tale?

A success story

Just prove your protocol secure using formal models and tools.

That’s it?
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.
I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.

I also need to authenticate the bank.

App designer

Security expert
I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.

I also need to authenticate the bank.

Let’s use a PKI infrastructure.
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.

I also need to authenticate the bank.

Let’s use a PKI infrastructure.

So I can pick any standard stuff and put everything together?
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.

I also need to authenticate the bank.

Let’s use a PKI infrastructure.

So I can pick any standard stuff and put everything together?

Hum, wait a minute...
I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.

I also need to authenticate the bank.

Let’s use a PKI infrastructure.

So I can pick any standard stuff and put everything together?

Hum, wait a minute...

→ Can we safely compose protocols?
Can we safely compose protocols?

**Option 1**
Put together the building blocks and analyze the resulting protocol
→ Tools do not scale very well.
→ Not modular (what if I change one of the subprotocols?)
Can we safely compose protocols?

Option 1
Put together the building blocks and analyze the resulting protocol
→ Tools do not scale very well.
→ Not modular (what if I change one of the subprotocols?)

Option 2
Secure composition of protocols
warm up: parallel composition with shared keys

Joint work with Stéphanie Delaune

Example: shared long-term keys, or public keys
Parallel composition

Does not work in general!

Protocols do not compose well as soon as they share data.

Protocol 1

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

Question

Does \( s \) remain confidential?
Parallel composition

Does not work in general!

Protocols do not compose well as soon as they share data.

Protocol 1

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

Protocol 2

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

Question

Does \( s \) remain confidential?
Parallel composition

Condition 1 - Tagging

condition 1 (well-tagged protocols)

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 avoids interaction between two different protocols.
Condition 1 - Tagging

**condition 1 (well-tagged protocols)**

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** avoids 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 $P_2$**

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

\[ B \rightarrow A : N_a \]
Condition 2 - Shared key are not revealed

Protocol $P_1$

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

Protocol $P_2$

$B \rightarrow A : \text{priv}(B)$
Condition 2 - Shared key are not revealed

Protocol $P_1$

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

Protocol $P_2$

\[ B \rightarrow A : \text{priv}(B) \]

Condition 2 (shared key are not revealed)

Let $SK$ be the set of shared keys, that is constants and long-term keys used in $P_1$ and $P_2$, not publicly known.

\[ \forall k \in SK \quad P_1 \not\vdash k, \quad P_2 \not\vdash k. \]

Example: We have that $SK = \{\text{priv}(B)\}$.

\[ \rightarrow \text{Condition 2 (shared key are not revealed) is not satisfied by } P_2. \]
Parallel composition theorem

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

1. $P_1$ is $\alpha$-tagged and $P_2$ is $\beta$-tagged with $\alpha \neq \beta$,
2. shared keys are not revealed.

\[
\forall k \in SK \quad P_1 \not\vdash k, \quad P_2 \not\vdash k
\]
Parallel composition theorem

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

1. $P_1$ is $\alpha$-tagged and $P_2$ is $\beta$-tagged with $\alpha \neq \beta$,
2. shared keys are not revealed.

$$\forall k \in SK \quad P_1 \not\triangleright k, \ P_2 \not\triangleright k$$

Parallel composition [FSTTCS’07+ACD-POST 2015]

$\nu k. P_1$ preserves the secrecy of $s$ if and only if $\nu k. (P_1 | P_2)$ preserves the secrecy of $s$. 
More general protocol composition

Joint work with Stefan Ciobaca

\[ P \models \phi \]

\[ Q \models \psi \]

\[ (P; Q) \models ? \]
More general protocol composition

Joint work with Stefan Ciobaca

\[ \nu k. \left( \begin{array}{ccc} P & Q & P & Q & P & Q \end{array} \right) = ? \]
Example: key establishment

- $P = P_1.(z_1 = t_1) \mid P_2.(z_2 = t_2)$ establishes a session key (e.g. Diffie-Hellman protocol).

- $Q = \text{out}(	ext{enc}(s, z_1))$ uses the key to encrypt.

Is $P.Q$ secure?
Assumptions

Of course, we need:

**Condition 1**  No interference between primitives

1. tagging standard primitives
2. disjoint primitives

**Condition 2**  Shared keys are not revealed

This is not sufficient!
Key freshness

Protocol $P$
establishes two keys $k_1, k_2$

Protocol $Q$
uses $k_1, k_2$

\[ \nu s. [x_{k_1} = x_{k_2}]. \text{out}(s) \]

$Q$ with idealized key distribution is secure:

\[ \nu k_1. \nu k_2. (x_{k_1} := k_1). (x_{k_2} := k_2). Q \vdash s \]
Sequential composition

Key freshness

Protocol $P$
establishes two keys $k_1, k_2$

Protocol $Q$
uses $k_1, k_2$

\[ \nu s. [x_{k_1} = x_{k_2}]. \text{out}(s) \]

$Q$ with idealized key distribution is secure:

\[ \nu k_1. \nu k_2. (x_{k_1} := k_1). (x_{k_2} := k_2). Q \vdash s \]

What about $Q$ composed with $P$?

\[ \nu k_1. \nu k_2. P.Q \vdash s \]
Key freshness

Protocol \( P \)
establishes two keys \( k_1, k_2 \)

\[
x_{k_1} := k_1
\]
\[
x_{k_2} := x_{k_1}
\]

Protocol \( Q \)
uses \( k_1, k_2 \)

\[
\nu s. [x_{k_1} = x_{k_2}] . \text{out}(s)
\]

\( Q \) with idealized key distribution is secure:

\[
\nu k_1 . \nu k_2 . (x_{k_1} := k_1) . (x_{k_2} := k_2) . Q \vdash s
\]

What about \( Q \) composed with \( P \)?

\[
\nu k_1 . \nu k_2 . P . Q \vdash s
\]

→ Composing protocols may create unexpected equalities.
Sequential composition

Theorem (simplified) [CSF 2010]

Assume

- $P$ and $Q$ tagged,
- $P \not\vdash s$ and $Q \not\vdash s$,
- $P$ and $Q$ do not reveal shared keys,
- $P$ and $Q$ instantiate two distinct keys to two distinct values.

Then $P.Q \not\vdash s$. 
Application: Key exchange protocol

Assume

\[ P = \nu \tilde{n} \cdot (P_1 | P_2) \not\vdash x_k, y_k \]

\[ Q = \nu k \cdot (x_k := k \cdot Q_1 | y_k := k \cdot Q_2) \not\vdash x_k, y_k, x_s \]
Application: Key exchange protocol

Assume

\[ P = \nu \, \tilde{n} \cdot (P_1 \mid P_2) \not\sqsupseteq x_k, y_k \]

\[ Q = \nu \, k \cdot (x_k := k \cdot Q_1 \mid y_k := k \cdot Q_2) \not\sqsupseteq x_k, y_k, x_s \]

By the composition theorem:

\[ W = \nu \, \tilde{n} \cdot (P_1 \cdot Q_1 \mid P_2 \cdot Q_2) \not\sqsupseteq x_s \]

Application: Diffie-Hellman can be used as a key exchanged protocol.
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Formal Model</th>
<th>Composition</th>
<th>Secure channels</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

This was about composition of *symmetric* key establishment protocols.

What about public key infrastructure (PKI)?
Secure composition of PKIs

The famous Needham-Schroeder protocol

\[
\begin{align*}
A & \rightarrow B : \quad \{ A, N_a \}_{\text{pub}(B)} \\
B & \rightarrow A : \quad \{ N_a, N_b \}_{\text{pub}(A)} \\
A & \rightarrow B : \quad \{ N_b \}_{\text{pub}(B)}
\end{align*}
\]

Can we use the Needham-Schroeder protocol with any Public Key Infrastructure (PKI)?
What is a secure PKI?

Typically properties assumed from a PKI: $\phi_{\text{PKI}}$

- Each honest agent has a unique public/private key pair.
- Honest agents have pairwise distinct private keys.
- Keys are consistently distributed, that is, honest agents know each other public and verification keys.
- Decryption keys of honest agents are private.
What is a secure PKI?

Typically properties assumed from a PKI: $\phi_{\text{PKI}}$

- Each honest agent has a unique public/private key pair.
- Honest agents have pairwise distinct private keys.
- Keys are consistently distributed, that is, honest agents know each other public and verification keys.
- Decryption keys of honest agents are private.

This may not be enough!
What can go wrong?

A simple protocol

\[ A \rightarrow B : [pkB, M]_{skA} \]
What can go wrong?

A simple protocol

\[ A \rightarrow B : [pkB, M]_{skA} \]

Then C registers \( pkC = pkB \), \( w \) for his public key and waits for a message from A.

\[ A \rightarrow C : [pkC, M]_{skA} \]
What can go wrong?

A simple protocol

\[ A \rightarrow B : [pkB, M]_{skA} \]

Then \( C \) registers \( pkC = pkB, w \) for his public key and waits for a message from \( A \).

\[ A \rightarrow C : [pkC, M]_{skA} \]

\( C(A) \rightarrow B : [pkB, w, M]_{skA} \) since \( pkC = pkB, w \)

\( C \) can now impersonate \( A \) w.r.t. \( B \).
What can go wrong? - continued

This (contrived) protocol is insecure as soon as the attacker registers the same public key pkC for both C and D.

In brief, standard PKIs do not provide guarantees on dishonest keys.
What can go wrong? - continued

This (contrived) protocol is insecure as soon as the attacker registers the same public key \( pkC \) for both \( C \) and \( D \).

In brief, standard PKIs do not provide guarantees on dishonest keys.
Option 1: you don’t know whether the underlying PKI provides guarantees on dishonest keys.
How to compose with a PKI? Option 1

Option 1: you don’t know whether the underlying PKI provides guarantees on dishonest keys.
→ analyse permissive $Q$ instead of the standard, idealized $Q$. 
How to compose with a PKI? Option 1

**Option 1**: you don’t know whether the underlying PKI provides guarantees on dishonest keys.

→ analyse *permissive* \( Q \) instead of the standard, idealized \( Q \).

**Needham-Schroeder**

\[
P := \\
\nu \text{priv}_A.\nu \text{priv}_B.\nu \text{priv}_C. \\
\text{out}(c, \text{pub(\text{priv}_A))).\text{out}(c, \text{pub(\text{priv}_B))).\text{out}(c, \text{priv}_C). \\
!(\nu \text{N}_A.\text{P}_A(\text{priv}_A, \text{pub(\text{priv}_B)}, A, B, N_A)) \\
!(\nu \text{N}_B.\text{P}_B(\text{priv}_B, \text{pub(\text{priv}_A)}, A, B, N_B)) \\
!(\nu \text{N}_A.\text{P}_A(\text{priv}_A, \text{pub(\text{priv}_C)}, A, C, N_A)) \\
!(\nu \text{N}_B.\text{P}_B(\text{priv}_B, \text{pub(\text{priv}_C)}, A, C, N_B))
\]
How to compose with a PKI? Option 1

**Option 1**: you don’t know whether the underlying PKI provides guarantees on dishonest keys.

→ analyse *permissive* $Q$ instead of the standard, idealized $Q$.

**Permissive Needham-Schroeder**

$$P :=$$

$$\nu \text{priv}_A.\nu \text{priv}_B.$$  
$$\text{out}(c, \text{pub}(\text{priv}_A)).\text{out}(c, \text{pub}(\text{priv}_B)).$$  
$$!(\nu N_A.P_A(\text{priv}_A, \text{pub}(\text{priv}_B), A, B, N_A))$$  
$$!(\nu N_B.P_B(\text{priv}_B, \text{pub}(\text{priv}_A), A, B, N_B))$$  
$$!\text{in}(c, y).(\nu N_A.P_A(\text{priv}_A, y, A, C, N_A))$$  
$$!\text{in}(c, y).(\nu N_B.P_B(\text{priv}_B, y, A, C, N_B))$$
Option 2: be more demanding on your PKI $\phi_{\text{ideal}}$
Option 2: be more demanding on your PKI $\phi_{\text{ideal}}$

- Each honest and dishonest agent has a unique public/private key pair.
- Honest and dishonest agents have pairwise distinct private keys.

Moreover, public key should be tagged when used in plaintext.

$$A \rightarrow B : [\text{tag}_{pk}(pkB), M]_{skA}$$
Joint work with Vincent Cheval and Bogdan Warinschi [CSF’17]

$P$ and $Q$

- may use arbitrary primitives (including e.g. Exclusive Or and modular exponentiation) but
- only share standard primitives (encryption, signature, hash, concatenation).

<table>
<thead>
<tr>
<th>$Q$ secure</th>
<th>$P$ secure PKI</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_{perm}$</td>
<td>$\phi_{PKI}$</td>
<td>Tagged Processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disjoint keys</td>
</tr>
<tr>
<td>$Q_{ideal}$</td>
<td>$\phi_{PKI} \land \phi_{ideal}$</td>
<td>Tagged Processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disjoint keys</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tagged public keys</td>
</tr>
</tbody>
</table>

Relaxed assumptions: disjoint keys.
For example, $P$ uses a key $pk(k)$ to encrypt and $Q$ uses $k$ for signature but not for encryption (and conversely).
An imaginary discussion

I wish to design a cool app for mobile payment. I need clients to authenticate.

Sure, use some password-based protocol.

I need secure communications between the mobile and the bank.

No problem, use some secure channel.
Refinement of secure channels

Protocols often assume an underlying secure channel.

We consider 4 types of channels.

- **Authenticated**: the attacker cannot write (e.g. signed messages, password-based authentication)
- **Confidential**: the attacker cannot read (e.g. public or symmetric encryption)
- **Secure**: cannot write, cannot read (e.g. encrypt-then-sign, TLS)
- **Public**: can write, can read
Example: 3D Secure

3D Secure: online payment protocol

- involves four users (client, merchant, bank, Visa)
- communications between any two users are assumed to be secure without further implementation details
- typically uses TLS sessions in practice
Example: 3D Secure

3D Secure: online payment protocol

- involves four users (client, merchant, bank, Visa)
- communications between any two users are assumed to be secure **without further implementation details**
- typically uses TLS sessions in practice

Verifying 3D secure involves the analysis of **TLS + 3D Secure**
Composition framework

Protocol $P$
establishes crypto material: $k_1, k_2, ...$
- session keys
- MAC keys
- signature certificates

Protocol $Q$
assumes secure channels

$A \rightarrow B : m_1$
$B \rightarrow_{auth} A : m_2$
$A \rightarrow_{conf} B : m_3$
$B \rightarrow_{secure} A : m_4$
$A \rightarrow_{secure} B : m_5$
Composition framework

Protocol $P$
- establishes crypto material: $k_1, k_2, \ldots$
  - session keys
  - MAC keys
  - signature certificates

Protocol $Q$
- assumes secure channels

Composed protocol $P \cdot Q$
- runs $P$

$A \rightarrow B : m_1$
$B \rightarrow A : E_{auth}(m_2; k_1, k_2, \ldots)$
$A \rightarrow B : E_{conf}(m_3; k_1, k_2, \ldots)$
$B \rightarrow A : E_{secure}(m_4; k_1, k_2, \ldots)$
$A \rightarrow B : E_{secure}(m_5; k_1, k_2, \ldots)$
Encapsulations

Three types of protecting encapsulations:

**authentic**: cannot write
- Example: \( E_{\text{sign}} = \text{sign}(x, x_1) \)
- \( E_{\text{mac}} = \langle x, h(x, x_1) \rangle \)

**confidential**: cannot read
- Example: \( E_{\text{ench}} = \text{enca}(x, \text{pub}(x_1)) \)

**secure**: cannot write, cannot read
- Example: \( E_{\text{TLS}} = \text{enc}(x, x_1) \)
- \( E_{\text{signcrypt}} = \text{sign}(\text{enca}(x, \text{pub}(x_1)), x_2) \)
- \( E_{\text{BAC}} = \langle \text{enc}(x, x_1), \text{mac}(\text{enc}(x, x_1), x_2) \rangle \)
Secure refinement of channels

[FSTTCS 2015, Joint work with V. Cheval and E. Le Morvan]

**Theorem (informal)**

Let $P$ be a secure channel establishment protocol. Let $Q$ be a protocol that preserves secrecy. Let $\mathcal{E} = \{\mathcal{E}_i\}_{i \in I}$ be a set of secure/authenticated/confidential encapsulations. Assume that:

- $P$, $Q$, $\mathcal{E}$ are tagged (with different tags);
- $P$ and $Q$ do not share data (only variables).

Then $P \cdot \mathcal{E} \cdot Q$ preserves secrecy.
Electronic passport protocol

- Contain RFID chip that stores personal information
- Data is optically read at the airport
- Two subprotocols for authentication:
  - Passive Authentication (PA)
  - Active Authentication (AA)
- They require a fresh secure channel
  → need for fresh session keys
  → provided by the Basic Access Control protocol (BAC)
- It is a composition pattern!
Experiments with the passport

- Experiments with the tool CL-ATSE
- Tagged version of BAC and Passive Authentication

<table>
<thead>
<tr>
<th>protocols</th>
<th>BAC&amp; PA</th>
<th>BAC&amp; AA</th>
<th>BAC&amp; PA&amp; AA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolated or Composed</td>
<td>I</td>
<td>C</td>
<td>I</td>
</tr>
<tr>
<td>number of sessions</td>
<td>1</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6.2</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>9133</td>
<td>-</td>
</tr>
</tbody>
</table>
Experiments with 3D secure and TLS

- Experiments with the tool CL-ATSE
- Tagged version of 3D Secure and TLS (Basic TLS handshake, in RSA mode)

<table>
<thead>
<tr>
<th>protocols</th>
<th>Analysis time (in sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TLS &amp; 3D Secure</td>
</tr>
<tr>
<td>Isolated or Composed</td>
<td>I</td>
</tr>
<tr>
<td>number of sessions</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>1350</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
The idea of adding an identifier is not novel:

- **Principle 10** in the prudent engineering paper, [Abadi & Needham, 1995]
- More formally introduced by Guttman & Thayer [CSFW 2000]
The idea of adding an identifier is not novel:

- **Principle 10** in the prudent engineering paper, [Abadi & Needham, 1995]
- More formally introduced by Guttman & Thayer [CSFW 2000]

**Cryptographic Protocol Composition via the Authentication Tests** [Guttmann, 2009]

→ Proposition of a general criteria for composing protocols

- The criteria has to be checked for any considered pair of protocols
- allow more fine grain composition

**Secure Pseudonymous Channels** [Mödersheim & Vigano, 2009]

- Explain how to implement a variety of channels: authenticated, confidential, ...
- “Refinement” oriented

→ continued with *vertical composition* [Gross & Mödersheim, 2011]
Related Work - continued

Protocol composition logic [Mitchel et al.]
- A logic for modularly proving security of protocols
- A large variety of results for composing protocols

Universal composability [Canetti et al.]
- Allow to compose protocols with primitives, not protocols with themselves
- Some joint state theorems, assuming strong typing assumptions

Ideal encryption functionality [Küsters & Tuengerthal]
- A composition theorem for key exchange protocols.
Lessons learned:
- composition is tricky
- “common sense” properties typically do not compose in general

Future work
- lack of a general, unifying composition result
- other channels: anonymous, more dynamic
- equivalence properties (done for parallel and sequential composition by [Arapinis, Cheval, Delaune, POST 2015])
- from symbolic to computational models