CSF 2021
Vertical Composition and Sound Payload Abstraction for Stateful Protocols

Sébastien Gondron and Sebastian Mödersheim

June 23, 2021
Motivations

Login application

running over a secure channel:
Vertical Composition

Is such a composition secure?

• Can the channel be replaced by a different one?
• Can the application be replaced by a different one?

Sébastien Gondron and Sebastian Mödersheim
Is such a composition secure?

- Can the channel be replaced by a different one?
- Can the application be replaced by a different one?
Composition of protocols with shared states is hard to get right.

Given:
- an application App,
- a channel Ch protocols,
- they are secure in isolation,
- and some conditions (???),

is their vertical composition also secure?
Composition of protocols with shared states is hard to get right.

**Vertical Composition**

Given:

- an application \( \text{App} \),
- a channel \( \text{Ch} \) protocols,
- they are secure in isolation,
- and some conditions (???)

is their **vertical composition** \( \begin{array}{c}
\text{App} \\
\hline
\text{Ch}
\end{array} \) also secure?
Can we solve vertical composition of stateful protocols?

What about a parallel composition\(^1\)?

- we consider a channel protocol $Ch$ and an application protocol $App$,
- they run in parallel and share sets as an interface, called inbox and outbox.

---

Can we solve vertical composition of stateful protocols?

What about a parallel composition\(^1\)?

- we consider a channel protocol $Ch$ and an application protocol $App$,
- they run in parallel and share sets as an interface, called inbox and outbox.

\(^{1}\)Hess, Mödersheim, and Brucker, “Stateful Protocol Composition”.
Can we solve vertical composition of stateful protocols?

What about a parallel composition\(^1\)?

- we consider a channel protocol $\text{Ch}$ and an application protocol $\text{App}$,
- they run in parallel and share sets as an interface, called inbox and outbox.

---

\(^1\)Hess, Mödersheim, and Brucker, “Stateful Protocol Composition”.
Can we solve vertical composition of stateful protocols?

What about a parallel composition\(^1\)?

- we consider a channel protocol Ch and an application protocol App,
- they run in parallel and share sets as an interface, called inbox and outbox.

\(^1\)Hess, Mödersheim, and Brucker, “Stateful Protocol Composition”. 
Verifying \( \text{Ch}^* \parallel \text{App} \) means that the application is secure and has no attack as long as:

- the channel does not manipulate the inbox and outbox sets in any other way than described in \( \text{Ch}^* \), and
- the channel does not leak any messages except those explicitly declassified in \( \text{Ch}^* \).
Channel Idealization

Verifying $\text{Ch}^* \parallel \text{App}$ means that the application is secure and has no attack as long as:

- the channel does not manipulate the inbox and outbox sets in any other way than described in $\text{Ch}^*$, and
- the channel does not leak any messages except those explicitly declassified in $\text{Ch}^*$.
Channel Idealization

Verifying $\text{Ch}^* \parallel \text{App}$ means that the application is secure and has no attack as long as:

- the channel does not manipulate the inbox and outbox sets in any other way than described in $\text{Ch}^*$, and
- the channel does not leak any messages except those explicitly declassified in $\text{Ch}^*$.

First success: any channel $\text{Ch}'$ with $\text{Ch}'^* = \text{Ch}^*$ works!
This is not enough!

Let us take stock!
This is not enough!

Let us take stock!

We still need to solve the other problem: $\text{Ch} \parallel \text{App}^*$. 
Abstracting the Payload

A concrete execution of \textit{Ch} \parallel \textit{App}^* has the concrete messages from the application:

- in the outbox and inbox sets, and
- as subterms of the messages that the channel transmits.
Abstracting the Payload

A concrete execution of $\text{Ch} \parallel \text{App}^*$ has the concrete messages from the application:

- in the outbox and inbox sets, and
- as subterms of the messages that the channel transmits.
Abstracting the Payload

A concrete execution of \( \text{Ch} \parallel \text{App}^* \) has the concrete messages from the application:

- in the outbox and inbox sets, and
- as subterms of the messages that the channel transmits.

But it should be

- simpler: we do not want the complexity of the messages of App, and
- more general: we do not want to verify the channel again when considering a different application.
Abstracting the Payload

The main idea is to transform Ch into an abstract channel $Ch^\#:$

- we remove outbox and inbox interface, and
- we replace payload variable with abstract constant
Abstracting the Payload

The main idea is to transform Ch into an abstract channel Ch♯:

- we remove outbox and inbox interface, and
- we replace payload variable with abstract constant

\[
\text{sk}(A, B) \rightarrow \text{abstract} \downarrow
\]

\[
\text{sk}(A, B) \rightarrow \text{abstract} \uparrow
\]

\(\text{abstract}\) can be

- known to the intruder or not, and
- fresh or reused.
To prove the security of $\text{App}_{\text{Ch}}$, it is enough to prove the security of $\text{Ch}^\dagger$ and of $\text{Ch}^* \parallel \text{App}$ (given that App and Ch respects a number of syntactic conditions such as disjointness).