Composition in the BC model

Hubert Comon, Charlie Jacomme, Guillaume Scerri

January 23th, 2020

Introduction

Practical Context

Who am I ?
A third year PhD Student, working in Paris and Nancy, supervised by:

- Hubert Comon-Lundh - LSV
- Steve Kremer - LORIA

The goal

We want security !
We want formal proofs of security, in the computational model.

The goal

We want security !
We want formal proofs of security, in the computational model.

But:

- There is few automation;
- proofs are long and error-prone;
- there is no modularity;
- and proofs size grows w.r.t to the size of the protocol.

Our contributions

The composition framework

- Allows to split the security of an unbounded number of sessions of a compound protocol into smaller finite goals;

Our contributions

The composition framework

- Allows to split the security of an unbounded number of sessions of a compound protocol into smaller finite goals;
- allows to consider protocols with state passing and long term shared secrets;

Our contributions

The composition framework

- Allows to split the security of an unbounded number of sessions of a compound protocol into smaller finite goals;
- allows to consider protocols with state passing and long term shared secrets;
- naturally translates to the BC model, and allows for the first time to perform proofs for an unbounded number of sessions in this model.

The BC model ?

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

| Checks the signature

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

| Checks the signature

$$\xleftarrow{<\text{“}ok\text{”}, r>}$$

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

| Checks the signature

$$\xleftarrow{<\text{``}ok\text{''}, r>}$$

In BC

Protocols are modelled with sequences of terms:

$$\phi_0 := \ \texttt{sign}(r, sk_A)$$

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

Names represent uniformly sampled bitstrings of length η

ks the signature

In BC

Protocols are modelled with sequences of terms:

$$\phi_0 := \texttt{sign}(r, sk_A)$$

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\texttt{sign}(r, skA)} B$$

Names represent uniformly sampled bitstrings of length η

ks the signature

In BC

Protocols are modelled with sequences of terms:

$$\phi_0 := \texttt{sign}(r, sk_A)$$

$$\phi_1 := \phi_0, \quad \begin{array}{l} \textbf{if } (\texttt{checksign}(g_0(\phi_0), pk(sk_A))) \textbf{ then} \\ \quad < \text{"}ok\text{"}, \texttt{getmess}(g_0(\phi_0))) > \end{array}$$

A quick introduction to the BC model

A protocol

$$A \xrightarrow{\mathtt{sign}(r, skA)} B$$

Names represent uniformly sampled bitstrings of length η

ks the signature

In BC

Protocols are modelled with sequences of terms:

$$\phi_0 := \mathtt{sign}(r, sk_A)$$

$$\phi_1 := \phi_0, \quad \begin{array}{l} \textbf{if} \ (\mathtt{checksign}(g_0(\phi_0), pk(sk_A))) \ \textbf{then} \\ \quad < \text{“}ok\text{”}, \ \mathtt{getmess}(g_0(\phi_0))) > \end{array}$$

Attacker inputs represented with non instantiated function symbol

A quick introduction to the BC model

How to reason on terms ?
A first order logic built over a predicate:

$$t_1 \sim t_2$$

A quick introduction to the BC model

How to reason on terms ?

A first order logic built over a predicate:

$$t_1 \sim t_2$$

For all η, for all interpretations of free function symbols by PPT, any attacker can only distinguish between t_1 and t_2 with negligible probability.

How to make proofs
Logical rules allow to reason about $\sim$:

A quick introduction to the BC model

How to make proofs
Logical rules allow to reason about $\sim$:

- for any term t, $t \sim t$

A quick introduction to the BC model

How to make proofs

Logical rules allow to reason about $\sim$:

- for any term t, $t \sim t$
- for any function symbol f and terms $t_1, \ldots, t_n, t'_1, \ldots, t'_n$,

$$t_1, \ldots, t_n \sim t'_1, \ldots, t'_n \Rightarrow f(t_1, \ldots, t_n) \sim f(t'_1, \ldots, t'_n)$$

How to make proofs
Logical rules allow to reason about $\sim$:

- for any term t, $t \sim t$
- for any function symbol f and terms $t_1, \ldots, t_n, t'_1, \ldots, t'_n$,

$$t_1, \ldots, t_n \sim t'_1, \ldots, t'_n \Rightarrow f(t_1, \ldots, t_n) \sim f(t'_1, \ldots, t'_n)$$

- transitivity, branching over conditionals, ...

A quick introduction to the BC model

EUF-CMA
For all terms t such that sk only appears in key position:

$$\mathtt{checksign}(t, pk(sk))) \Rightarrow \bigvee_{\mathsf{sign}(x,sk) \in \mathtt{St}(t)} t \doteq \mathsf{sign}(x, sk)$$

A quick introduction to the BC model

EUF-CMA
For all terms t such that sk only appears in key position:

$$\begin{aligned}&\mathtt{checksign}(t, pk(sk))) \Rightarrow \\ &\quad \bigvee_{\mathsf{sign}(x,sk)\in\, \mathtt{St}(t)} t \doteq \mathsf{sign}(x, sk) \\ &\sim \mathit{true}\end{aligned}$$

A quick introduction to the BC model

A reminder of our protocol

$$\phi_0 := \texttt{sign}(r, sk_A)$$

$$\phi_1 := \phi_0, \begin{array}{l}\textbf{if } (\texttt{checksign}(g_0(\phi_0), pk(sk_A))) \textbf{ then} \\ \quad < \text{“}ok\text{”}, \texttt{getmess}(g_0(\phi_0))) >)\end{array}$$

A quick introduction to the BC model

A reminder of our protocol

$$\phi_0 := \ \mathtt{sign}(r, sk_A)$$

$$\phi_1 := \phi_0, \begin{array}{l} \textbf{if} \ (\ \mathtt{checksign}(g_0(\phi_0), pk(sk_A))) \ \textbf{then} \\ \quad < \text{“}ok\text{”}, \ \mathtt{getmess}(g_0(\phi_0))) >) \end{array}$$

A security property

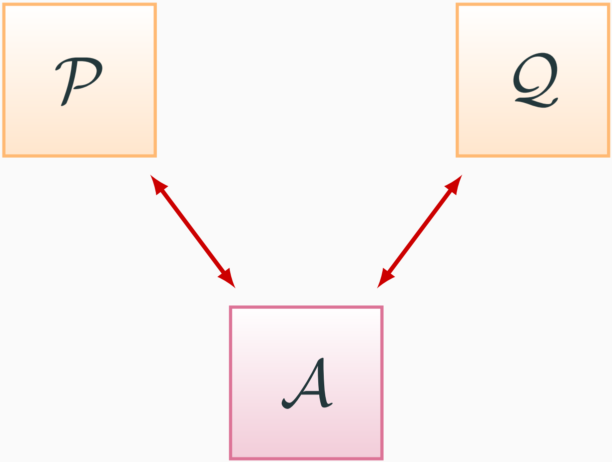
EUF-CMA $\models \phi_1 \sim$

$$\mathtt{sign}(r, sk_A), \begin{array}{l} \textbf{if} \ (\ \mathtt{checksign}(g_0(\phi_0), pk(sk_A))) \ \textbf{then} \\ \quad < \text{“}ok\text{”}, r > \end{array}$$

A compositional framework inside the computational model

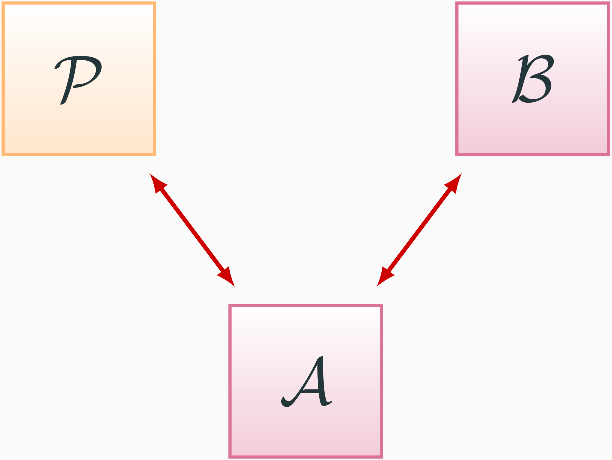
A classical proof technique

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while also having access to $\mathcal{Q}$.



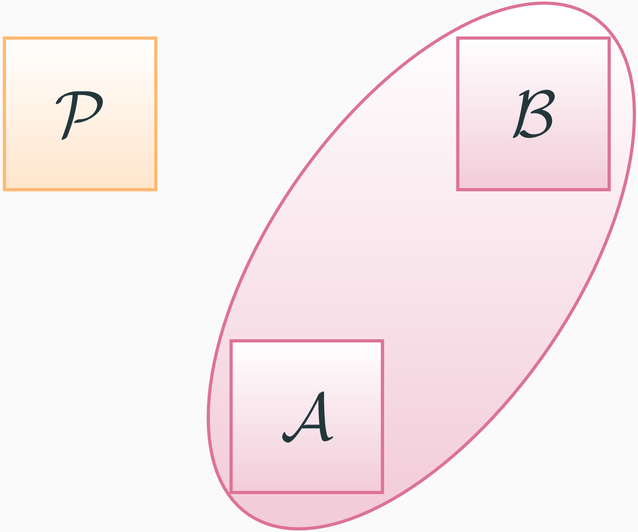
A classical proof technique

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while also simulating $\mathcal{Q}$.



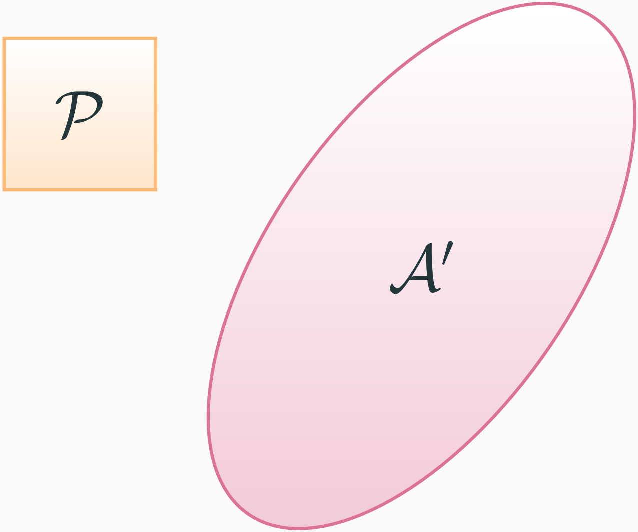
A classical proof technique

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while also simulating $\mathcal{Q}$.



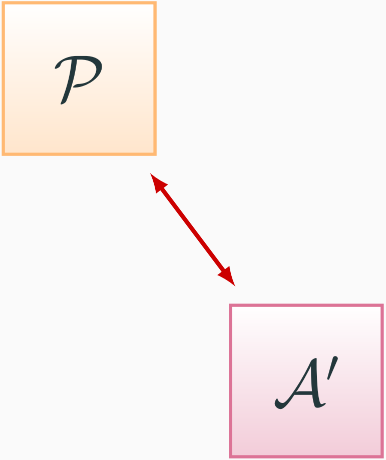
A classical proof technique

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while also simulating $\mathcal{Q}$.



A classical proof technique

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while also simulating $\mathcal{Q}$.



Simulation

The main idea

If $\mathcal{A}$ can simulate it, i.e produce exactly all the same messages:

we remove Q from the picture!

Simulation

The main idea
If $\mathcal{A}$ can simulate it, i.e produce exactly all the same messages:

we remove Q from the picture!

The difficulty
If P and Q share some secret key sk, $\mathcal{A}$ cannot simulate messages which require sk.

The main idea

Exemple for signatures

- $\mathcal{P}_{sk}$ may produce $\texttt{sign}(< m, \text{``tag}_1\text{''} >, sk)$
- $\mathcal{Q}_{sk}$ may produce $\texttt{sign}(< m', \text{``tag}_2\text{''} >, sk)$

The main idea

Exemple for signatures

- $\mathcal{P}_{sk}$ may produce $\texttt{sign}(< m, \text{"tag}_1\text{"} >, sk)$
- $\mathcal{Q}_{sk}$ may produce $\texttt{sign}(< m', \text{"tag}_2\text{"} >, sk)$

To prove $\mathcal{P}$ while abstracting $\mathcal{Q}$, the attacker must be able to produce $\texttt{sign}(< m', \text{"tag}_2\text{"} >, sk)$.

The main idea

Exemple for signatures

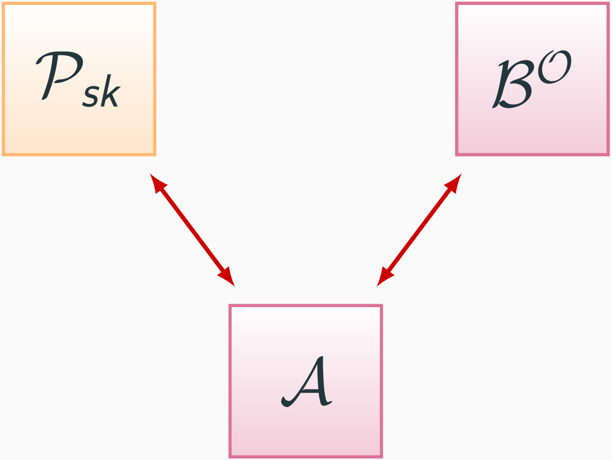
- $\mathcal{P}_{sk}$ may produce $\texttt{sign}(< m, \text{"tag}_1\text{"} >, sk)$
- $\mathcal{Q}_{sk}$ may produce $\texttt{sign}(< m', \text{"tag}_2\text{"} >, sk)$

To prove $\mathcal{P}$ while abstracting $\mathcal{Q}$, the attacker must be able to produce $\texttt{sign}(< m', \text{"tag}_2\text{"} >, sk)$.

$\hookrightarrow$ We may give an oracle to the attacker, allowing to obtain $\texttt{sign}(< m', \text{"tag}_2\text{"} >, sk)$ but not $\texttt{sign}(< m, \text{"tag}_1\text{"} >, sk)$

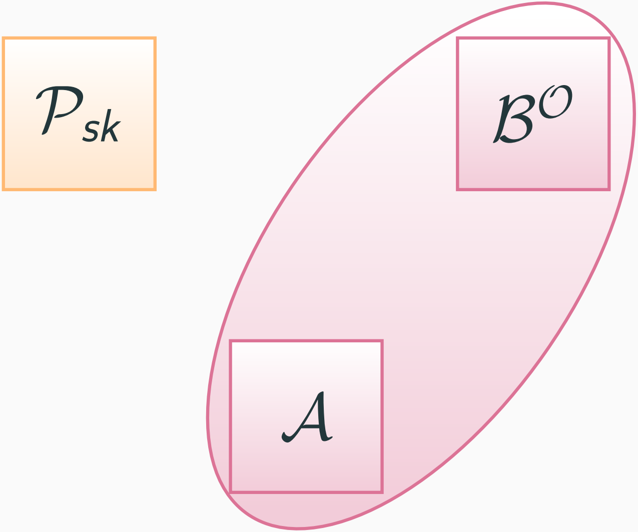
The main idea

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while simulating $\mathcal{Q}$ thanks to oracle $\mathcal{O}$.



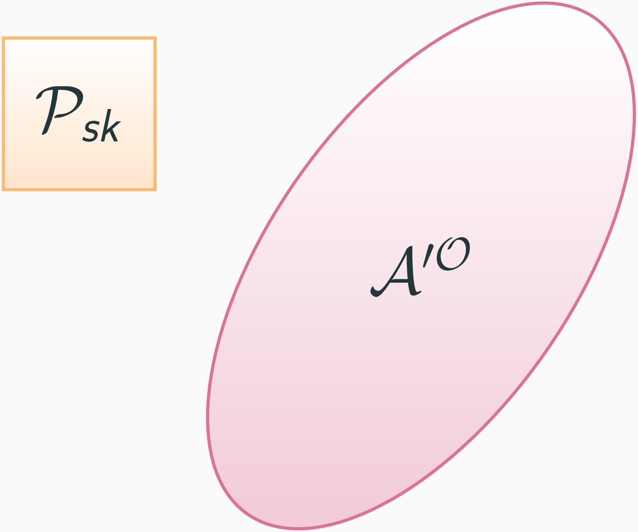
The main idea

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while simulating $\mathcal{Q}$ thanks to oracle $\mathcal{O}$.



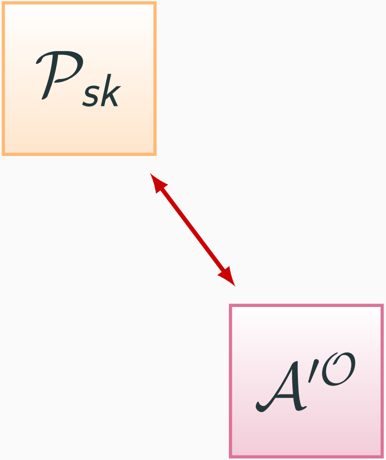
The main idea

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while simulating $\mathcal{Q}$ thanks to oracle $\mathcal{O}$.



The main idea

$\mathcal{A}$ is trying to break protocol $\mathcal{P}$, while simulating $\mathcal{Q}$ thanks to oracle $\mathcal{O}$.



In the BC model

Classical Setting

To prove the security of $\mathcal{P}$ against $\mathcal{A}$, we define axioms Ax that are sound for any PPT, and prove that $Ax \models \phi_P$.

In the BC model

Classical Setting
To prove the security of $\mathcal{P}$ against $\mathcal{A}$, we define axioms Ax that are sound for any PPT, and prove that $Ax \models \phi_P$.

New Axioms
To prove $\mathcal{P}$ against $\mathcal{A}^{\mathcal{O}}$, we just have find axioms $Ax_{\mathcal{O}}$ that are sounds for all PPTOM.

On an example

A small DDH example

Signed DDH

$$
\begin{array}{ccc}
A(a, skA) & & B(b, skB) \\
& \xrightarrow{\text{sign}(g^a, skA)} & \\
& & x_B = g^a \\
& \xleftarrow{\text{sign}(<g^a, g^b>, skB)} & \\
x_A = g^b & & \\
& \xrightarrow{\text{sign}(<g^a, g^b>, skA)} & \\
k_A = x_A^a & & k_B = x_B^b
\end{array}
$$

A small DDH example

The security property:

$$\|^{i\leq N}(A(a_i, skA); \mathbf{out}(k_A) \| B(b_i, skB); \mathbf{out}(k_B))$$

$$\sim$$

$$\begin{aligned}
&\|^{i\leq N-1}(A(a_i, skA); \mathbf{out}(k_A) \| B(b_i, skB); \mathbf{out}(k_B)) \\
&\| \ A(a_N, skA); \textbf{ if } x_A = g^{b_N} \textbf{ then } \mathbf{out}(k_{N,N}) \\
&\qquad\qquad\qquad \textbf{else if } x_A \notin \{g^{b_i}\}_{1\leq i\leq N} \textbf{ then } \perp \\
&\| \ B(b_N, skB); \textbf{ if } x_B = g^{a_N} \textbf{ then } \mathbf{out}(k_{N,N}) \\
&\qquad\qquad\qquad \textbf{else if } x_B \notin \{g^{a_i}\}_{1\leq i\leq N} \textbf{ then } \perp
\end{aligned}$$

A small DDH example

The final security property:
Let's assume the attacker can simulate

$$\|^{i \le N-1}(A(a_i, skA); \mathbf{out}(k_A) \| B(b_i, skB); \mathbf{out}(k_B))$$

.

A small DDH example

The final security property:
Let's assume the attacker can simulate

$$\|^{i \leq N-1}(A(a_i, skA); \mathbf{out}(k_A) \| B(b_i, skB); \mathbf{out}(k_B))$$

. We can simply prove:

$$\begin{array}{c} A(a_N, skA); \mathbf{out}(k_A) \| B(b_N, skB); \mathbf{out}(k_B) \\ \sim \\ A(a_N, skA);\ \mathbf{if}\ x_A = g^{b_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ \qquad\qquad \mathbf{else\ if}\ x_A \notin \{g^{b_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \bot \\ \|\ B(b_N, skB);\ \mathbf{if}\ x_B = g^{a_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ \qquad\qquad \mathbf{else\ if}\ x_B \notin \{g^{a_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \bot \end{array}$$

A small DDH example

The final security property:
Let's assume the attacker can simulate

$$\|^{i \leq N-1}(A(a_i, skA); \mathbf{out}(k_A) \| B(b_i, skB); \mathbf{out}(k_B))$$

. We can simply prove:

$$\begin{array}{c} A(a_N, skA); \mathbf{out}(k_A) \| B(b_N, skB); \mathbf{out}(k_B) \\ \sim \\ A(a_N, skA);\ \mathbf{if}\ x_A = g^{b_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ \qquad\qquad\qquad \mathbf{else\ if}\ x_A \notin \{g^{b_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \perp \\ \|\ B(b_N, skB);\ \mathbf{if}\ x_B = g^{a_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ \qquad\qquad\qquad \mathbf{else\ if}\ x_B \notin \{g^{a_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \perp \end{array}$$

$\hookrightarrow$ How to simulate the $N-1$ sessions ?

Simulating the sessions

What must the attacker be able to produce ?
He must be able to start some A:

$$\forall 1 \leq i \leq N-1.\ \mathsf{sign}(g^{a_i}, skA)$$

Simulating the sessions

What must the attacker be able to produce ?
He must be able to start some A:

$$\forall 1 \leq i \leq N-1.\ \mathrm{sign}(g^{a_i}, skA)$$

And for any DDH share r he receives, he should be able to produce:

- $\forall 1 \leq i \leq N-1.\ \mathrm{sign}(< g^{a_i}, r >, skA)$

Simulating the sessions

What must the attacker be able to produce ?
He must be able to start some A:

$$\forall 1 \leq i \leq N-1.\ \mathrm{sign}(g^{a_i}, skA)$$

And for any DDH share r he receives, he should be able to produce:

- $\forall 1 \leq i \leq N-1.\ \mathrm{sign}(< g^{a_i}, r >, skA)$
- $\forall 1 \leq i \leq N-1.\ \mathrm{sign}(< r, g^{b_i} >, skB)$

Generic signing oracles

T signing oracle

$$\begin{aligned}\mathcal{O}^{\mathsf{sign}}_{T,sk} : \ & \mathsf{input}(m) \\ & \textbf{if } T(m) \textbf{ then} \\ & \quad \mathsf{output}(\mathsf{sign}(m, sk)))\end{aligned}$$

Generic signing oracles

T signing oracle

$$
\begin{aligned}
\mathcal{O}^{\mathrm{sign}}_{T,sk} : &\ \mathrm{input}(m) \\
&\ \textbf{if } T(m) \textbf{ then} \\
&\quad \mathrm{output}(\mathrm{sign}(m, sk)))
\end{aligned}
$$

Give the attacher access to $\mathcal{O}^{sign}_{T,skA}$ and $\mathcal{O}^{sign}_{T,skB}$ with:

$$
T(m) = \mathrm{true} \Leftrightarrow \exists 1 \leq i \leq N-1, r. \begin{cases} m = g^{a_i} \\ m = < g^{a_i}, r > \\ m = < r, g^{b_i} > \end{cases}
$$

Generic signing oracles

T signing oracle

$$\mathcal{O}^{\mathsf{sign}}_{T,sk} : \mathsf{input}(m)$$
$$\quad \textbf{if } T(m) \textbf{ then}$$
$$\qquad \mathsf{output}(\mathsf{sign}(m, sk)))$$

Give the attacher access to $\mathcal{O}^{sign}_{T,skA}$ and $\mathcal{O}^{sign}_{T,skB}$ with:

$$T(m) = \mathsf{true} \Leftrightarrow \exists 1 \leq i \leq N-1, r. \begin{cases} m = g^{a_i} \\ m =< g^{a_i}, r > \\ m =< r, g^{b_i} > \end{cases}$$

$\hookrightarrow$ How to make the proof for such attackers ?

T-EUFCMA

For any computable function T, for all terms t such that sk only appears in key position:

$$
\begin{aligned}
&\mathtt{checksign}(t, pk(sk))) \Rightarrow \\
&\quad T(\ \mathtt{getmess}(t)) \\
&\quad \bigvee_{\mathsf{sign}(x,sk) \in\ \mathtt{St}(t)} (t \doteq \mathsf{sign}(x, sk))) \\
&\sim \mathit{true}
\end{aligned}
$$

The final proof

Assumption

$$\begin{array}{l} \texttt{checksign}(t, pk(sk))) \Rightarrow \\ \quad \exists 1 \leq i \leq N-1, r.\ \ \texttt{getmess}(t) \in \{g^{a_i}, < g^{a_i}, r >, < r, g^{b_i} >\} \\ \quad \bigvee_{\mathsf{sign}(x,sk) \in \texttt{St}(t)} (t \doteq \mathsf{sign}(x, sk))) \\ \sim \mathit{true} \end{array}$$

The final proof

Assumption

$$
\begin{array}{l}
\quad \mathtt{checksign}(t, pk(sk)) \Rightarrow \\
\qquad \exists 1 \leq i \leq N-1, r.\ \ \mathtt{getmess}(t) \in \{g^{a_i}, < g^{a_i}, r >, < r, g^{b_i} >\} \\
\qquad \bigvee_{\mathsf{sign}(x,sk) \in \mathtt{St}(t)} (t \doteq \mathsf{sign}(x, sk))) \\
\quad \sim \mathit{true} \\
\wedge \quad DDH : g^{a_N}, g^{b_N}, g^{a_N b_N} \sim g^{a_N}, g^{b_N}, k_{N,N}
\end{array}
$$

The final proof

Goal

$$A(a_N, skA); \mathbf{out}(k_A) \| B(b_N, skB); \mathbf{out}(k_B)$$

$$\sim$$

$$\begin{aligned} & A(a_N, skA);\ \mathbf{if}\ x_A = g^{b_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ & \qquad \mathbf{else\ if}\ x_A \notin \{g^{b_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \perp \\ \| \ & B(b_N, skB);\ \mathbf{if}\ x_B = g^{a_N}\ \mathbf{then}\ \mathbf{out}(k_{N,N}) \\ & \qquad \mathbf{else\ if}\ x_B \notin \{g^{a_i}\}_{1 \leq i \leq N}\ \mathbf{then}\ \perp \end{aligned}$$

The final proof

Synchronization

$$
\begin{aligned}
& A(a_N, skA); \textbf{ if } x_A = g^{b_N} \textbf{ then out}(g^{a_N b_N}) \\
& \qquad \textbf{else if } x_A \notin \{g^{b_i}\}_{1\leq i\leq N} \textbf{ then out}(x_A^{a_N}) \\
\parallel\ & B(b_N, skB); \textbf{ if } x_B = g^{a_N} \textbf{ then out}(g^{a_N b_N}) \\
& \qquad \textbf{else if } x_B \notin \{g^{a_i}\}_{1\leq i\leq N} \textbf{ then out}(x_B^{b_N})
\end{aligned}
$$

$$\sim$$

$$
\begin{aligned}
& A(a_N, skA); \textbf{ if } x_A = g^{b_N} \textbf{ then out}(k_{N,N}) \\
& \qquad \textbf{else if } x_A \notin \{g^{b_i}\}_{1\leq i\leq N} \textbf{ then } \perp \\
\parallel\ & B(b_N, skB); \textbf{ if } x_B = g^{a_N} \textbf{ then out}(k_{N,N}) \\
& \qquad \textbf{else if } x_B \notin \{g^{a_i}\}_{1\leq i\leq N} \textbf{ then } \perp
\end{aligned}
$$

Synchronization

$$\begin{aligned}
&A(a_N, skA);\ \textbf{if}\ x_A = g^{b_N}\ \textbf{then out}(g^{a_N b_N})\\
&\qquad\qquad \textbf{else if}\ x_A \notin \{g^{b_i}\}_{1\le i\le N}\ \textbf{then out}(x_A^{a_N})\\
\parallel\ &B(b_N, skB);\ \textbf{if}\ x_B = g^{a_N}\ \textbf{then out}(g^{a_N b_N})\\
&\qquad\qquad \textbf{else if}\ x_B \notin \{g^{a_i}\}_{1\le i\le N}\ \textbf{then out}(x_B^{b_N})
\end{aligned}$$

$$\sim$$

$$\begin{aligned}
&A(a_N, skA);\ \textbf{if}\ x_A = g^{b_N}\ \textbf{then out}(k_{N,N})\\
&\qquad\qquad \textbf{else if}\ x_A \notin \{g^{b_i}\}_{1\le i\le N}\ \textbf{then}\ \bot\\
\parallel\ &B(b_N, skB);\ \textbf{if}\ x_B = g^{a_N}\ \textbf{then out}(k_{N,N})\\
&\qquad\qquad \textbf{else if}\ x_B \notin \{g^{a_i}\}_{1\le i\le N}\ \textbf{then}\ \bot
\end{aligned}$$

The final proof

Synchronization

$$
\begin{aligned}
& A(a_N, skA);\ \text{if } x_A = g^{b_N} \text{ then out}(g^{a_N b_N}) \\
& \qquad \text{else if } x_A \notin \{g^{b_i}\}_{1\le i\le N} \text{ then out}(x_A^{a_N}) \\
\parallel\ & B(b_N, skB);\ \text{if } x_B = g^{a_N} \text{ then out}(g^{a_N b_N}) \\
& \qquad \text{else if } x_B \notin \{g^{a_i}\}_{1\le i\le N} \text{ then out}(x_B^{b_N})
\end{aligned}
$$

$$\sim$$

$$
\begin{aligned}
& A(a_N, skA);\ \text{if } x_A = g^{b_N} \text{ then out}(k_{N,N}) \\
& \qquad \text{else if } x_A \notin \{g^{b_i}\}_{1\le i\le N} \text{ then } \perp \\
\parallel\ & B(b_N, skB);\ \text{if } x_B = g^{a_N} \text{ then out}(k_{N,N}) \\
& \qquad \text{else if } x_B \notin \{g^{a_i}\}_{1\le i\le N} \text{ then } \perp
\end{aligned}
$$

The final proof

Synchronization
Proof steps Split the conditonals into four cases and,

The final proof

Synchronization
Proof steps Split the conditonals into four cases and,

1. use DDH to show indistinguishability,

The final proof

Synchronization
Proof steps Split the conditonals into four cases and,

1. use DDH to show indistinguishability,
2. use T-EUF-CMA, to show that $x_A \notin \{g^{b_i}\}_{1\leq i\leq N}$ is never true (e.g, $\perp$ unreachable),

The final proof

Synchronization
Proof steps Split the conditonals into four cases and,

1. use DDH to show indistinguishability,
2. use T-EUF-CMA, to show that $x_A \notin \{g^{b_i}\}_{1 \leq i \leq N}$ is never true (e.g, $\perp$ unreachable),
3. similar to (2);
4. similar to (2);

Conclusion

Done and to do?

The composition framework

- Composition results for parallel and sequential composition (in the BC model),

Done and to do?

The composition framework

- Composition results for parallel and sequential composition (in the BC model),
- allows for long-term shared secrets and state-passing,

The composition framework

- Composition results for parallel and sequential composition (in the BC model),
- allows for long-term shared secrets and state-passing,
- allows for reduction from unbounded number of sessions to a single one,

Done and to do?

The composition framework

- Composition results for parallel and sequential composition (in the BC model),
- allows for long-term shared secrets and state-passing,
- allows for reduction from unbounded number of sessions to a single one,
- applied to key exchange (with key confirmations).

Done and to do?

A tool
We are working on an interactive prover:

1. First allow to performe (un)-reachability proofs, (WIP)

Done and to do?

A tool

We are working on an interactive prover:

1. First allow to performe (un)-reachability proofs, (WIP)
2. then integrate with indistinguishability proofs,

Done and to do?

A tool

We are working on an interactive prover:

1. First allow to performe (un)-reachability proofs, (WIP)
2. then integrate with indistinguishability proofs,
3. and use the composition framework along with the tool to perform case studies.

Extra slides with too many details

A core theorem

Composition without replication

Let $C[\_1, \dots, \_n]$ be a context such that the variable k_i is bound in each hole $\_i$ and $P_1(x), \dots, P_n(x)$ be parametrized protocols, such that all channels are disjoint. Given an oracle $\mathcal{O}$, with $\overline{n} \supset \mathcal{N}(C) \cap \mathcal{N}(P_1, \dots, P_n)$, if, with $k'_1, \dots, k'_n$ fresh names,

1. $C[\mathbf{out}(1, k_1), \dots, \mathbf{out}(n, k_n)] \cong_{\mathcal{O}} C[\mathbf{out}(1, k'_1), \dots, \mathbf{out}(n, k'_n)]$
2. $\nu \overline{n}.\mathbf{in}(x).P_1(x) \| \dots \| \mathbf{in}(x).P_n(x)$ is $\mathcal{O}$-simulatable

Then $C[P_1(k_1), \dots, P_n(k_n)] \cong_{\mathcal{O}} C[P_1(k'_1), \dots, P_n(k'_n)]$

A core theorem

Unbounded parallel Composition
Let $\mathcal{O}_r$ be an oracle and Ax a set of axioms both parametrized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of shared secrets, $P(\overline{x})$, $R(\overline{x}, \overline{y}, \overline{z})$ and $Q(\overline{x}, \overline{y})$ be parametrized protocols. If we have, for a sequence of names $\overline{lsid}$ and any integers n, if with $\overline{s} = \overline{lsid}_1, \dots, \overline{lsid}_n$ n copies of $\overline{lsid}$:

1. $\forall\ 1 \leq i \leq n, \nu \overline{p}.t_{R(\overline{p}, \overline{lsid}_i, \overline{s})}$ is $\mathcal{O}_r$ simulatable.
2. Ax is $\mathcal{O}_r$ sound.
3. $Ax \models t_{P(\overline{p})} \sim t_{Q(\overline{p}, \overline{s})}$

Then, for any integer n:

$$P(\overline{p}) \| \, !_n R(\overline{p}, \overline{lsid}, \overline{s}) \\ \cong Q(\overline{p}, \overline{s}) \| \, !_n R(\overline{p}, \overline{lsid}, \overline{s})$$

A core theorem

Unbounded parallel Composition
Let $\mathcal{O}_r$ be an oracle and Ax a set of axioms both parametrized by a sequence of names $\overline{s}$. Let $\overline{p}$ be a sequence of shared secrets, $P(\overline{x}, \overline{y})$ and $Q(\overline{x}, \overline{y}, \overline{z})$ be parametrized protocols. If we have, for sequences of names $\overline{lsid}_p, \overline{lsid}_q$ and any integers n, if with $\overline{s} = \overline{lsid}_{p,1}, \dots, \overline{lsid}_{p,n}, \dots, \overline{lsid}_{q,n}$ sequences of copies of $\overline{lsid}_p, \overline{lsid}_q$

1. $\forall\ 1 \leq i \leq n, \nu\overline{p}.t_{P(\overline{p}, \overline{lsid}_{p,i})}$ is $\mathcal{O}_r$ simulatable.
2. $\forall\ 1 \leq i \leq n, \nu\overline{p}.t_{Q(\overline{p}, \overline{lsid}_{q,i}, \overline{s})}$ is $\mathcal{O}_r$ simulatable.
3. Ax is $\mathcal{O}_r$ sound.
4. $Ax \models t_{P(\overline{p}, \overline{lsid}_p)} \sim t_{Q(\overline{p}, \overline{lsid}_q, \overline{s})}$

Then, for any integers n:

$$!_n P(\overline{p}, \overline{lsid}_p) \cong_{\mathcal{O}} !_n Q(\overline{p}, \overline{s}, \overline{lsid}_q)$$