Universal Composability is Secure Compilation

Marco Patrignani$^{1,2}$  Riad S. Wahby$^1$  Robert Künneeman$^2$

25$^{th}$ January 2020

1 Stanford University  2 CISPA
Unveil a similarity between two fields
Goal

Unveil a similarity between two fields

Explore how each field can benefit from the other
Fields: UC
Universal Composability: UC

- **gold standard** for proving security of crypto protocols under concurrent composition
Universal Composability: UC

- **gold standard** for proving security of crypto protocols under concurrent composition
- overcome main drawback in protocol vulnerabilities: composition
Universal Composability: UC

- gold standard for proving security of crypto protocols under concurrent composition
- overcome main drawback in protocol vulnerabilities: composition
- many flavours: UC\(^1\), SaUCy\(^2\), iUC\(^3\)

---

\(^1\)Canetti. 2001. “Universally composable security”


\(^3\)Camenisch et al. 2019 “iUC: Flexible Universal Composability Made Simple”
Universal Composability: UC

• gold standard for proving security of crypto protocols under concurrent composition
• overcome main drawback in protocol vulnerabilities: composition
• many flavours: UC¹, SaUCy², iUC³

This talk: generic presentation, geared towards the newer theories SaUCy and iUC

¹Canetti. 2001. “Universally composable security”
³Camenisch et al. 2019 “iUC: Flexible Universal Composability Made Simple”
UC Base Notions: ITMs

- protocols $\Pi$ (using concrete crypto)

```
commitment for $b \in \{0, 1\}$ with SID sid:
compute $G_{pk_b}(r)$ for random $r \in \{0, 1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\text{Com}, sid, y)$ to the receiver

Upon receiving $(\text{Com}, sid, y)$ from $P_i$, $P_j$ outputs $(\text{Receipt}, sid, cid, P_i, P_j)$
```

UC Base Notions: ITMs

• protocols $\Pi$ (using concrete crypto)

\[
\text{commitment for } b \in \{0, 1\} \text{ with } SID \text{ sid:}
\]
\[
\begin{align*}
&\text{compute } G_{pk_b}(r) \text{ for random } r \in \{0, 1\}^n \\
&\text{set } y = G_{pk_b}(r) \text{ for } b = 0, \text{ or } y = G_{pk_b}(r) \oplus \sigma \text{ for } b = 1 \\
&\text{send } (\text{Com}, \text{sid}, y) \text{ to the receiver}
\end{align*}
\]

Upon receiving $(\text{Com}, \text{sid}, y)$ from $P_i, P_j$ outputs $(\text{Receipt}, \text{sid}, cid, P_i, P_j)$

• functionalities $F$ (using abstract notions)

1. Upon receiving a value $(\text{Commit}, \text{sid}, P_i, P_j, b)$ from $P_i$, where $b \in \{0, 1\}$, record the value $b$ and send the message $(\text{Receipt}, \text{sid}, P_i, P_j)$ to $P_j$ and $S$. Ignore any subsequent Commit messages.

---

UC Base Notions: ITMs

• protocols Π (using concrete crypto)

commitment for $b \in \{0,1\}$ with SID sid:

compute $G_{pk_b}(r)$ for random $r \in \{0,1\}^n$
set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
send $(\text{Com}, sid, y)$ to the receiver

Upon receiving $(\text{Com}, sid, y)$ from $P_i$, $P_j$ outputs $(\text{Receipt}, sid, cid, P_i, P_j)$

• functionalities F (using abstract notions)

1. Upon receiving a value $(\text{Commit}, sid, P_i, P_j, b)$ from $P_i$, where $b \in \{0,1\}$, record the value $b$ and send the message $(\text{Receipt}, sid, P_i, P_j)$ to $P_j$ and $S$. Ignore any subsequent Commit messages.

• attackers A & S (corrupting parties etc.)

---

UC Base Notions: ITMs

- protocols $\Pi$ (using concrete crypto)
  
  commitment for $b \in \{0, 1\}$ with SID $sid$:
  
  compute $G_{pk_b}(r)$ for random $r \in \{0, 1\}^n$
  set $y = G_{pk_b}(r)$ for $b = 0$, or $y = G_{pk_b}(r) \oplus \sigma$ for $b = 1$
  send $(\text{Com}, \text{sid}, y)$ to the receiver

  Upon receiving $(\text{Com}, \text{sid}, y)$ from $P_i, P_j$ outputs $(\text{Receipt}, \text{sid}, \text{cid}, P_i, P_j)$

- functionalities $F$ (using abstract notions)

  1. Upon receiving a value $(\text{Commit}, \text{sid}, P_i, P_j, b)$ from $P_i$, where $b \in \{0, 1\}$, record the value $b$ and send the message $(\text{Receipt}, \text{sid}, P_i, P_j)$ to $P_j$ and $S$. Ignore any subsequent Commit messages.

- attackers $A$ & $S$ (corrupting parties etc.)

- environments $Z$ (objective witness)

---

UC (Semi-formally)

\[ \Pi \leftarrow A \leftrightarrow \text{represent communication channels} \]

\[ \text{∀ poly} A, \exists S, \forall Z. \text{Exec}[Z, A, \Pi] \approx \text{Exec}[Z, S, F] \]
$\Pi \models_{\text{UC}} F \overset{\text{def}}{=} \forall \text{poly } A, \exists S, \forall Z.$

$\text{Exec}[Z, A, \Pi] \approx \text{Exec}[Z, S, F]$
UC, Pros and Cons

- modularise protocols
- small building blocks
- reusable results
UC, Pros and Cons

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

Existing work: points one.osf and two.osf.
Our work: points three.osf and four.osf.
UC, **Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

Existing work (SaUCy and iUC): points 1 and 2
**UC, Pros and Cons**

- modularise protocols
- small building blocks
- reusable results

1. informal formalism
2. pseudocode protocols
3. (PL-wise) informal proofs
4. no (ish) mechanisation

Existing work (SaUCy and iUC): points 1 and 2

**Our work:** points 3 and 4
if $\Pi_1 \vdash_{UC} F_1$

• and $\Pi_{big} \overset{\text{def}}{=} \Pi_{part} [\Pi_1]$

• and $F_{big} \overset{\text{def}}{=} \Pi_{part} [F_1]$
UC Benefits: Compositionality

- if $\Pi_1 \vdash_{UC} F_1$
- and $\Pi_{\text{big}} \overset{\text{def}}{=} \Pi_{\text{part}} [\Pi_1]$
- and $F_{\text{big}} \overset{\text{def}}{=} \Pi_{\text{part}} [F_1]$

recall they are all ITMs
• if $\Pi_1 \vdash_{UC} F_1$
• and $\Pi_{big} \overset{\text{def}}{=} \Pi_{part} [\Pi_1]$
• and $F_{big} \overset{\text{def}}{=} \Pi_{part} [F_1]$
• then $\Pi_{big} \vdash_{UC} F_{big}$
• if $\Pi_1 \vdash_{UC} F_1$
• and $\Pi_{big} \overset{\text{def}}{=} \Pi_{part} [\Pi_1]$
• and $F_{big} \overset{\text{def}}{=} \Pi_{part} [F_1]$
• then $\Pi_{big} \vdash_{UC} F_{big} = \Pi_{part} [\Pi_1] \vdash_{UC} F_{big}$
• if $\Pi_1 \vdash_{UC} F_1$
• and $\Pi_{\text{big}} \overset{\text{def}}{=} \Pi_{\text{part}}[\Pi_1]$
• and $F_{\text{big}} \overset{\text{def}}{=} \Pi_{\text{part}}[F_1]$
• then $\Pi_{\text{big}} \vdash_{UC} F_{\text{big}} =$

$\Pi_{\text{part}}[\Pi_1] \vdash_{UC} F_{\text{big}} =$

$\Pi_{\text{part}}[\Pi_1] \vdash_{UC} \Pi_{\text{part}}[F_1]$
Fields

UC SC
Fields: $SC$
Secure Compilation: $SC$
Secure Compilation: \( SC \)

- many criteria: \( FAC^5 \), \( TPC^6 \), \( RSCC^7 \), ...

---

7 Abate et al. 2018. “When Good Components Go Bad …”
Robust Criteria for $SC$

Abate et al. 2019. “Journey Beyond Full Abstraction …”
Robust Criteria for $SC$

Abate et al. 2019. “Journey Beyond Full Abstraction ...”
Robust Hyperproperty Preservation: \( RHC \)
Robust Hyperproperty Preservation: $RHC$

\[
\text{[P]} \bowtie A \iff P \bowtie A
\]

\[
\llbracket \cdot \rrbracket \vdash RHC \overset{\text{def}}{=} \forall P, A. \exists A. \forall t.
\]

\[
A \bowtie \llbracket P \rrbracket \sim t \iff A \bowtie P \sim t
\]
∀poly A, ∃S, ∀Z.

∀P, A. ∃A. ∀t.

Π ✢ A

0/1

↑

Z

≈

F ✢ S

Π ✢ A

0/1

↑

Z

[P] ✢ A ✢ P ✢ A

12/19
### Analogy

<table>
<thead>
<tr>
<th>UC</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>compiled program</td>
</tr>
<tr>
<td>concrete attacker</td>
<td>target context</td>
</tr>
<tr>
<td>ideal functionality</td>
<td>source program</td>
</tr>
<tr>
<td>simulator</td>
<td>source context</td>
</tr>
<tr>
<td>environment, output</td>
<td>trace, semantics</td>
</tr>
<tr>
<td>communication</td>
<td>linking</td>
</tr>
<tr>
<td>probabilistic equiv.</td>
<td>trace equality</td>
</tr>
</tbody>
</table>

- Human translation: $\Pi \rightarrow F \cdot P : P \rightarrow P$

- General composition result...

- Communication $\leftrightarrow \bowtie$

- Probabilistic equiv. $\approx \iff$
## Analogy

<table>
<thead>
<tr>
<th>UC</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>compiled program</td>
</tr>
<tr>
<td>concrete attacker</td>
<td>target context</td>
</tr>
<tr>
<td>ideal functionality</td>
<td>source program</td>
</tr>
<tr>
<td>simulator</td>
<td>source context</td>
</tr>
<tr>
<td>environment, output</td>
<td>trace, semantics</td>
</tr>
<tr>
<td>communication</td>
<td>linking</td>
</tr>
<tr>
<td>probabilistic equiv.</td>
<td>trace equality</td>
</tr>
<tr>
<td>human translation  ( \Pi \rightarrow \mathcal{F} )</td>
<td>( \cdot \cdot \cdot ): ( P \rightarrow \mathcal{P} ) compiler</td>
</tr>
<tr>
<td>general composition result</td>
<td>...</td>
</tr>
</tbody>
</table>
Our Claim

UC and $RHC$ are similar enough so that we can reuse metatheoretical results of one system for the other.
Benefits

Cryptographers:

• must specify hidden UC assumptions\(^8\)
• more formal UC proofs
• mechanisation of UC results

\(^8\)As advocated by: Barbosa et al. 2019. “SoK: Computer-aided Cryptography”
Benefits

Cryptographers:

- must specify hidden UC assumptions
- more formal UC proofs
- mechanisation of UC results

Secure-compilationers:

- understand composition of $SC$ results

---

Cryptographers:

- must specify hidden UC assumptions\(^8\)
- more formal UC proofs
- mechanisation of UC results

Secure-compilationers:

- understand composition of $SC$ results

\(^8\) As advocated by: Barbosa et al. 2019. “SoK: Computer-aided Cryptography”
1. • formalise simple functionalities and protocols in ILC
• prove their compiler is \textit{RHC}
UC Roadmap

1. • formalise simple functionalities and protocols in ILC
   • prove their compiler is \( RHC \)

2. • formally prove (a version of) UC (iUC) and \( RHC \) are equivalent
• *RHC* defined for [$\cdot$] but paper mentions
  \[ \text{chains} = \text{compiler, linker(s), ...} = (\underbrace{\text{\textbf{[\cdot]}}, \text{\textbf{\[\cdot\]}}, \text{\textbf{\[\cdot\]}}, \text{\textbf{\[\cdot\]}}, \text{\textbf{\[\cdot\]}}}_{\text{chains}}) \]
**SC Roadmap**

- *RHC* defined for $[\cdot]$ but paper mentions
  \[\text{chains} = \text{compiler, linker(s), \ldots} = (\cdot, \star, \star)\]

Assuming these are *RHC*:

- \((\cdot)^S_T, \star, \star\) \((\cdot)^O_T, \star, \star\) \((\cdot)^T_B, \star, \star\)

What can we say about:

- \((\cdot)^S_B = (\cdot)^S_T \circ (\cdot)^T_B, \star, \star)\)?
• *RHC* defined for \([\cdot]\) but paper mentions chains = compiler, linker(s), \ldots = (\([\cdot]\), \(\bowtie\), \(\bowtie\))

Assuming these are *RHC*:

• \((\([\cdot]\)^S_T, \bowtie, \bowtie) (\([\cdot]\)^O_T, \bowtie, \bowtie) (\([\cdot]\)^T_B, \bowtie, \bowtie)\)

What can we say about:

• \((\([\cdot]\)^S_B = \([\cdot]\)^S_T \circ \([\cdot]\)^T_B, \bowtie, \bowtie)\)?
• \((\([\cdot]\)^S_T \cup \([\cdot]\)^O_T = \([\cdot]\)^S_T \cup \([\cdot]\)^O_T, \bowtie \cup \bowtie, \bowtie)\)?
**Roadmap**

- **RHC** defined for $[\cdot]$ but paper mentions
  
  = compiler, linker(s), ... = ($[\cdot]$, $\bowtie$, $\bowtie$)

Assuming these are **RHC**:

- ($[\cdot]^S_T$, $\bowtie$, $\bowtie$) ($[\cdot]^O_T$, $\bowtie$, $\bowtie$) ($[\cdot]^T_B$, $\bowtie$, $\bowtie$)

What can we say about:

- $([\cdot]^S_B = [\cdot]^S_T \circ [\cdot]^T_B$, $\bowtie$, $\bowtie$)?
- $([\cdot]^S_T \cup [\cdot]^O_T = [\cdot]^S_T \cup [\cdot]^O_T$, $\bowtie$ $\cup$ $\bowtie$, $\bowtie$)?
- $P = \left[ P \right]^S_T \bowtie \left[ P \right]^O_T$
But Fully Abstract Compilation …

\[ \Pi \sim \frac{0/1}{\uparrow} Z \xrightarrow{\approx} \frac{0/1}{\uparrow} Z \]

\[ \frac{[P_1]}{[P_2]} \quad \overset{\bowtie}{\Rightarrow} \quad \frac{A \Downarrow}{A \Downarrow} \quad \overset{\equiv}{\Rightarrow} \quad \frac{P_1 \bowtie P_2}{A \Downarrow} \quad \overset{\equiv}{\Rightarrow} \quad \frac{A \Downarrow}{A \Downarrow} \]

\textit{FAC} is relational, \textit{RHC} is propositional, like UC
But Fully Abstract Compilation …
Questions?
But What is the $\forall P$?

- each pair $P - [P]$ is a pair of UC $\mathbb{F} - \Pi$
- $[P]_{S}^{T} = \begin{cases} P & \text{if } P \vdash_{UC} P \\ P & \text{otherwise} \end{cases}$

In this interpretation, $S$ and $T$ are ITMs.
But Attackers and Environments …

- UC works employ a dummy attacker
- the \( \forall Z \) accounts for attacker behaviour
- \( Z \) has some “objective” behaviour
But Attackers and Environments …

- UC works employ a dummy attacker
- the $\forall Z$ accounts for attacker behaviour
- $Z$ has some “objective” behaviour
- we leave the attacker business in $A$
- and the semantics ($\rightarrow$) to the objectivity

this is similar to the EasyUC work
But UC was Mechanised in EasyCrypt[^9]

But UC was Mechanised in EasyCrypt\textsuperscript{9}

- with a titanic effort

But UC was Mechanised in EasyCrypt

- with a titanic effort
- our analogy is tool-indipendent

---

But UC was Mechanised in EasyCrypt\(^9\)

- with a titanic effort
- our analogy is tool-independent
- some similarities between the approaches (see next)

---