Unconditionally Secure    Universally Composable

Commitments

from

Physical Assumptions

Ivan Damgård and Alessandra Scafuro
Wait...Isn’t done yet?

Universally Composable
Wait...Isn’t done yet?

Universally Composable

Statefull tokens [K07..]
Stateless tokens [CGS08..]

Trusted PUFs [BSK11]
Malicious PUFs [OSVW13]

(trusted)
Signature card [HMQU11..]
Wait...Isn’t done yet?

Universally Composable AND Unconditionally Secure

Statefull tokens [K07..]
Stateless tokens [CGS08..]

Trusted PUFs [BFSK11]
Malicious PUFs [OSVW13]

(trusted)
Signature card [HMQU11..]
Wait...Isn’t done yet?

Universally Composable AND Unconditionally Secure

Stateful tokens [K07..]
Stateless tokens [CGS08..]

Trusted PUFs [BFSK11]
Malicious PUFs [OSW13]

(trusted)
Signature card [HMQU11..]

Stateful tokens [MS08..]
Wait...Isn’t done yet?

**Universally Composable** AND **Unconditionally Secure**

- Statefull tokens [K07..]
- Stateless tokens [CGS08..]
- Trusted PUFs [BFSK11]
- Malicious PUFs [OSVW13]
- (trusted)
- Signature card [HMQU11..]

- Statefull tokens [MS08..]
- Trusted PUFs
Wait...Isn’t done yet?

Universally Composable AND Unconditionally Secure

Statefull tokens [K07..]
Stateless tokens [CGS08..]
Trusted PUFs [BFSK11]
Malicious PUFs [OSVW13]
(trusted)
Signature card [HMQU11..]

Statefull tokens [MS08..]
this work

Trusted PUFs
this work
Unconditionally Secure   Universally Composable

Commitments
Unconditionally Secure    Universally Composable

Commitments

S         R
Unconditionally Secure Universally Composable

Commitments

S  Commitment phase  R

______________________________
Decommitment phase
Unconditionally Secure    Universally Composable

Commitments

S  Commitment phase  R

Encode of $m$

Decommitment phase
Unconditionally Secure  Universally Composable

Commitments

S
Commitment phase

Encode of $m$

hiding

Decommitment phase
Unconditionally Secure

Universally Composable

Commitments

Commitment phase

Encode of $m$

Decommitment phase

$m, \overline{m}$

hiding
Unconditionally Secure  Universally Composable

Commitments

S  
Commitment phase  R  
Encode of $m$  hiding

Decommitment phase  

$m,$  

Unconditionally Secure  Universally Composable

Commitments

Commitment phase

Encode of $m$

R

hiding

Decommitment phase

$m$, binding
Unconditionally Secure

Universally Composable

Commitments

Commitment phase

Encode of $m$

Decommitment phase

$R$

hiding

binding

$m, \underline{\text{binding}}$
Unconditionally Secure
Universally Composable

Commitments

Commitment phase: Encode of $m$
Decommitment phase: $m$, hiding, binding

R
Unconditionally Secure  Universally Composable

Commitments

Commitment phase

Encode of $m$

R

hiding

Decommitment phase

$m, R$

binding
Unconditionally Secure

Universally Composable

Commitments

Protocol A

$P_k$

Commitment phase

Encode of $m$

Decommitment phase

$m, R$

Protocol B

$hiding$

Protocol C

$binding$

[CKL03] Setup Assumptions
Unconditionally Secure

Universally Composable

Commitments

Protocol A

Encode of $m$

Commitment phase

Decommitment phase

Protocol B

Protocol C

[CKL03] Setup Assumptions
Unconditionally Secure

Universally Composable

Commitments

Encode of $m$

Commitment phase

Decommitment phase

hiding

binding

Protocol A

$P_k$

$P_1$

Protocol B

Protocol C

[CKL03] Setup Assumptions
Unconditionally Secure

Universally Composable

Commitments

Protocol A

$P_k$

Commitment phase

Encode of $m$

$hiding$

Decommitment phase

$m, R$

Protocol B

$P_1$

$hiding$

Protocol C

$P_1$

Physical Assumptions

[CKL03] Setup Assumptions
Physical Assumptions

Tamper-proof hardware token

Physically Uncloneable Functions (PUFs)
Physical Assumptions

Tamper-proof hardware token

Physically Uncloneable Functions PUFs
Physical Assumptions

Tamper-proof hardware token

Physically Uncloneable Functions PUFs

S f R
Physical Assumptions

Tamper-proof hardware token

Physically Uncloneable Functions (PUFs)

S \xrightarrow{f(x)} R \xleftarrow{f(x)}
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

R learns only $f(x)$

$S \xrightarrow{f} R \xleftarrow{f(x)} R$

Physically Uncloneable Functions
PUFs
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

\[ R \text{ learns only } f(x) \]

Stateful

- (tamper-proof) updatable memory
- reset attacks

Stateless

Physically Uncloneable Functions

PUFs
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

R learns only f(x)

S → R

x

f(x)

Stateful

• (tamper-proof) updatable memory
• reset attacks

Stateless

Physically Uncloneable Functions
PUFs
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

R learns only f(x)

Stateful

• (tamper-proof) updatable memory
• reset attacks

Physically Uncloneable Functions

PUFs

Stateless
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

\[ R \text{ learns only } f(x) \]

Stateful
- (tamper-proof) updatable memory
- reset attacks

Stateless

Physically Uncloneable Functions
PUFs

\[ S \xrightarrow{x} R \]

\[ S \xrightarrow{f(x)} R \]
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

\[ R \text{ learns only } f(x) \]

Stateful

• (tamper-proof) updatable memory
• reset attacks

Physically Uncloneable Functions

PUFs

Stateless
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

R learns only $f(x)$

Stateful

- (tamper-proof) updatable memory
- reset attacks

Stateless

Physically Uncloneable Functions

PUFs

Assumption: unpredictability

R cannot predict the answer on $y \neq x$ (with $y$ far from $x$)
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof

R learns only $f(x)$

$S \xrightarrow{f} R \xleftarrow{f(x)} S$

Stateful

• (tamper-proof) updatable memory
• reset attacks

Physically Uncloneable Functions

PUFs

Assumption: unpredictability

$R$ cannot predict the answer on $y \neq x$ (with $y$ far from $x$)

$S \xrightarrow{x} R \xleftarrow{u} S$

unpredictability holds for “far enough” challenges
Physical Assumptions

Tamper-proof hardware token

Assumption: tamper-proof
R learns only f(x)

Stateful
• (tamper-proof) updatable memory
• reset attacks

Physically Uncloneable Functions
PUFs

Assumption: unpredictability
R cannot predict the answer on y != x (with y far from x)

unpredictability holds for “far enough” challenges

Stateless

Assumption
physically uncloneable
UC-Modeling Physical Assumptions

Tamper-proof Model [Katz07]

(Malicious) PUF- Model [BFKSI11,OSVW13]
UC-Modeling Physical Assumptions

Tamper-proof Model
[Katz07]

(Malicious) PUF- Model
[BFKSI11,OSVW13]

$\text{F}_{\text{wrap}}$

S → f → x R
UC-Modeling Physical Assumptions

Tamper-proof Model [Katz07]

(Malicious) PUF- Model [BFKS11,OSVW13]

F_{\text{wrap}}

f \rightarrow f^* \rightarrow x \rightarrow R

arbitrary stateful function
UC-Modeling Physical Assumptions

Tamper-proof Model [Katz07]

(Malicious) PUF- Model [BFKS11,OSVW13]

programmable extraction of queries

arbitrary stateful function
UC-Modelling Physical Assumptions

Tamper-proof Model [Katz07]

- f
- f*
- arbitrary stateful function

- programmable extraction of queries

(Malicious) PUF- Model [BFKS11,OSVW13]

- F_{puf}
- create
- S

- R
UC-Modeling Physical Assumptions

Tamper-proof Model

[Katz07]

(Malicious) PUF- Model

[BFKS11,OSVW13]

programmable extraction of queries

arbitrary stateful function

F_{puf}
UC-Modeling Physical Assumptions

Tamper-proof Model
[Katz07]

(Malicious) PUF- Model
[BFKS11,OSVW13]

programmable extraction of queries

arbitrary stateful function

arbitrary stateful function

\(f \rightarrow f^*\)

\(x \rightarrow R\)

\(F^{p,p^*}_{\text{puf}}\)

\(\text{create} \rightarrow \text{create}^*\)

\(x \rightarrow R\)
UC-Modeling Physical Assumptions

Tamper-proof Model
[\textit{Katz07}]

(Malicious) PUF- Model
[\textit{BFKS11,OSVW13}]

\[ f^* \xrightarrow{\text{create}} \mathbf{F}^p p^* \]
\[ \mathbf{R} \times \]

programmable extraction of queries

arbitrary \textit{stateful} function

arbitrary \textit{stateful} function

Difference with trusted PUFs model
UC-Modeling Physical Assumptions

Tamper-proof Model
[Katz07]

(Malicious) PUF- Model
[BFKS11,OSVW13]

programmable extraction of queries

arbitrary stateful function

\( f \rightarrow f^* \)

create

create*

\( F_{puf}^{p,p^*} \)

arbitrary stateful function

Difference with trusted PUFs model
UC-Modeling Physical Assumptions

Tamper-proof Model
[Katz07]

(Malicious) PUF- Model
[BFKS11,OSVW13]

programmable extraction of queries

non-programmable extraction of queries

arbitrary stateful function

Difference with trusted PUFs model
UC-Modeling Physical Assumptions

Tamper-proof Model
[Katz07]

(Malicious) PUF- Model
[BFKS11,OSVW13]

Crucial: the security of each party depends only on the “goodness” of its own hardware.
### State of the art

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateless Tokens</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unconditional UC Oblivious Transfer</td>
<td>Unconditional UC Commitment</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Stateless Tokens</td>
<td>Impossible [GIMS10]</td>
<td></td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unconditional UC Oblivious Transfer</td>
<td>Unconditional UC Commitment</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### State of the art

<table>
<thead>
<tr>
<th>Stateless Tokens</th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
</table>

(Malicious) PUFs
## State of the art

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stateless Tokens</strong></td>
<td>Impossible [GIMS10]</td>
<td>stand-alone Com [GIMS10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restricted adversary</td>
</tr>
<tr>
<td><strong>(Malicious) PUFs</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

-不可能 [GIMS10]
## State of the art

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stateless Tokens</strong></td>
<td>Impossible [GIJM10]</td>
<td>stand-alone Com [GIJM10] restricted adversary</td>
</tr>
<tr>
<td><strong>(Malicious) PUFs</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## State of the art

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stateless Tokens</strong></td>
<td>Impossible [GIMS10]</td>
<td>stand-alone Com [GIMS10] restricted adversary</td>
</tr>
<tr>
<td><strong>(Malicious) PUFs</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

stand-alone Com [OSVW13]
### State of the art

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateless Tokens</td>
<td>Impossible [GIMS10]</td>
<td>stand-alone Com [GIMS10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restricted adversary</td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stand-alone Com [OSVW13]</td>
</tr>
</tbody>
</table>

**Caveat:** Adv allowed to only poly queries
In this paper

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateless Tokens</td>
<td>Impossible [GIMS10]</td>
<td>Stand-alone Com [GIMS10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restricted adversary</td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stand-alone Com [OSVW13]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>?</td>
</tr>
</tbody>
</table>
In this paper

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateless Tokens</td>
<td>Impossible [GIMS10]</td>
<td>Yes</td>
</tr>
<tr>
<td>(Malicious) PUFs</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

stand-alone Com [OSVW13]
In this paper

<table>
<thead>
<tr>
<th></th>
<th>Unconditional UC Oblivious Transfer</th>
<th>Unconditional UC Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stateless Tokens</strong></td>
<td>Impossible [GIMS10]</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>(Malicious) PUFs</strong></td>
<td>?</td>
<td>Yes</td>
</tr>
</tbody>
</table>
UC-Commitments with Physical Assumptions

Commitment phase

Encode of \( m \)

Straight-line extractable

Commitment phase

\textit{junk}

Straight-line equivocal

Decommitment phase

\( m \)
Our Technique
Our Technique

• Black-box Unconditional compiler
Our Technique

• Black-box Unconditional compiler

Extractable Com $\Rightarrow$ Equivocal + Extractable Com
Our Technique

- Black-box Unconditional compiler
  
  \( \text{Extractable Com} \Rightarrow \text{Equivocal} + \text{Extractable Com} \)

- Extractable Com
  
  (Malicious) PUF
  
  Stateless Token
Black-Box Compiler
Black-box Proof of Equality of Commitments

S

R
Black-box Proof of Equality of Commitments

S \quad R

\[ \text{[a, b]} \]
Black-box Proof of Equality of Commitments

S \rightarrow R

\begin{array}{cc}
a & b \\
\end{array}

\text{zero-knowledge} \quad \begin{array}{c}
? \\
a = b \\
\end{array}
Black-box Proof of Equality of Commitments

\[ a = b \]

Kilian 92

S \[ \rightarrow \]

R

zero-knowledge

\[ a = b \]
Black-box Proof of Equality of Commitments

Kilian 92

S \quad R

\[ a \quad b \]

\[ a = b \]

zero-knowledge
Black-box Proof of Equality of Commitments

Kilian 92

\( S \quad R \)

\[
\begin{array}{c}
\text{zero-knowledge} \\
\text{a} = b
\end{array}
\]

\( S \quad a \quad R \)

\[
\begin{array}{c}
a_0 \\
a_1
\end{array}
\]
Black-box Proof of Equality of Commitments

S \rightarrow R

a \rightarrow b

zero-knowledge \quad a = b

Kilian 92

S \quad a \quad b \quad R

\quad \quad a_0 \quad b_0

\quad \quad a_1 \quad b_1
Black-box Proof of Equality of Commitments

Kilian 92

S \rightarrow R

a \quad b

\text{zero-knowledge}

a = b

S

\begin{array}{c}
    a_0 \\
    a_1
\end{array}

\rightarrow

R

\begin{array}{c}
    b_0 \\
    b_1
\end{array}
Black-box Proof of Equality of Commitments

\[ a = b \]

Kilian 92

Zero-knowledge
Black-box Proof of Equality of Commitments

Kilian 92

\[ a = b \]

\[ y = a_0 + b_0 \]
Black-box Proof of Equality of Commitments

S

? 

a = b

R

Kilian 92

S

a

b

R

a₀  b₀

a₁  b₁

y = a₀ + b₀

zero-knowledge
Black-box Proof of Equality of Commitments

\[ a = b \]

Kilian 92

\[ y = a_0 + b_0 \]

zero-knowledge
Black-box Proof of Equality of Commitments

Kilian 92

\[ S \rightarrow R \]

\[ a \]  \[ b \]

\[ a = b \]

\[ y = a_0 + b_0 \]

\[ y = a_e + b_e \]

\[ S \]

\[ a \]

\[ a_0 \]

\[ a_1 \]

\[ b \]

\[ b_0 \]

\[ b_1 \]

zero-knowledge
Black-box Proof of Equality of Commitments

Kilian 92

S → R

? \(a = b\)

zero-knowledge

\[y = a_0 + b_0\]

zero-knowledge

\[y = a_e + b_e\]
Black-box Proof of Equality of Commitments

Zero-knowledge

Kilian 92

Soundness 1/2

\[ y = a_0 + b_0 \]

\[ y = a_e + b_e \]
Straight-line ZK BB proof of equality
Straight-line ZK BB proof of equality

Assume boxes are extractable
Straight-line ZK BB proof of equality

Assume boxes are extractable

\[
\begin{array}{c}
S \quad a \quad b \\
\hline
a_0 \quad b_0 \\
a_1 \quad b_1
\end{array}
\]
Straight-line ZK BB proof of equality

Assume boxes are extractable

zero-knowledge
Straight-line ZK BB proof of equality

Assume boxes are extractable

\[
y = a_e + b_e
\]
Straight-line ZK BB proof of equality

Assume boxes are extractable

$y = a_e + b_e$
Compiler

Extractable Commitment => Equivocal + Extractable
Compiler: Equivocal commitments from Extractable Commitments

b

S

R

Decommitment
Compiler: Equivocal commitments from Extractable Commitments

\[ b \quad \xrightarrow{S} \quad b \quad \begin{array}{c} b_0 \\ b_1 \end{array} \quad \xrightarrow{R} \]

Decommitment
Compiler: Equivocal commitments from Extractable Commitments

\[ \text{Decommitment} \]
Compiler: Equivocal commitments from Extractable Commitments

\[ b \]

\[ S \]

\[ b \]
\[ b_0 \]
\[ b_1 \]

\[ R \]
\[ b \]
\[ b' \]
\[ b'_0 \]
\[ b'_1 \]

\[ ZK \]
\[ b = b' \]

Decommitment
Compiler: Equivocal commitments from Extractable Commitments

S

\[ b \]

\[ b_0\]
\[ b_1\]

ZK

\[ b = b \]

R

\[ b \]
\[ b'_0 \]
\[ b'_1 \]

Decommitment

open either first or second com
Compiler: Equivocal commitments from Extractable Commitments

idea used in Hof11

Decommitment

open either first or second com
Equivocality

Straight-line equivocality

Decommitment
Equivocality

Straight-line equivocality

\[
\begin{array}{c}
0 \\
b_0 \\
b_1
\end{array}
\]

Decommitment
Equivocality

Straight-line equivocality

\[
\begin{array}{c|c}
0 & 1 \\
\hline
b_0 & b'_0 \\
b_1 & b'_1 \\
\end{array}
\]

Decommitment
Equivocality

Straight-line equivocality

\[
\begin{array}{c}
0 \\
\hline
b_0 \quad b_1 \\
\hline
1 \\
\hline
b'_0 \quad b'_1 \\
\end{array}
\]

ZK
0 = 1

Decommitment
Equivocality

Straight-line equivocality

Decommitment

\[
\begin{array}{cccc}
0 & 1 \\
\hline
b_0 & b'_0 \\
b_1 & b'_1 \\
\end{array}
\]

ZK

0 = 1
Equivocality

Straight-line equivocality

\[ \begin{array}{cc}
0 & 1 \\
\hline
b_0 & b'_0 \\
b_1 & b'_1 \\
\end{array} \]

ZK
\[ 0 = 1 \]

Decommitment
Equivocality

Straight-line equivocality

0 \quad 1

\begin{array}{c}
\begin{array}{c}
0
\end{array} \\
\begin{array}{c}
1
\end{array}
\end{array}

\begin{array}{c}
\begin{array}{c}
0'
\end{array} \\
\begin{array}{c}
1'
\end{array}
\end{array}

ZK

0=1

Decommitment

open \textbf{b}
Extractability

Decommitment

open either
a or b

Straight-line extractability

a = b

b_0  b'_0

b_1  b'_1
Extractability

Check that all open to the same bit

Decommitment

Open either $a$ or $b$
Our Technique

• Black-box compiler

Extractable Com \Rightarrow\text{Equivocal} + \text{Extractable Com}

• Extractable Com

(Malicious) PUF

Stateless Token
Extractable Commitment from (malicious) PUFs
Extractable Commitment from (malicious) PUFs
Extractable Commitment from (malicious) PUFs

**for free:** stand-alone unconditional commitment from [OSVW13]

PUF-Com
Extractable Commitment from (malicious) PUFs

S          R

for free: stand-alone unconditional commitment from [OSVW13]
PUF-Com
Extractable Commitment from (malicious) PUFs

for free: stand-alone unconditional commitment from [OSVW13]
PUF-Com
Extractable Commitment from (malicious) PUFs

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com
Extractable Commitment from (malicious) PUFs

\[ c = \text{PUF-Com}(a_b) \]

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com
Extractable Commitment from (malicious) PUFs

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

\[ c = \text{PUF-Com}(a_b) \]
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

\[ a_b \rightarrow b \]

\[ c = \text{PUF-Com}(a_b) \]

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

open \( a_b \)
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea \[\text{for free: stand-alone unconditional commitment from [OSVW13]}\]

\[c = \text{PUF-Com}(b)\]

open \(a_b\)
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07,CGS08]

$ab \xrightarrow{opening\ of\ c} \xleftarrow{c=PUF-Com(b)}$

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

open $ab$
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07, CGS08]

ab opening of c

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

c=PUF-Com(b)

PUF-Com(ab)

open ab
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07, CGS08]

ab opening of c

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

S

R

c=PUF-Com(b)

PUF-Com(ab)

open ab open c
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea

Problem 2: Adv queries with strings that are “close” to the actual opening

unpredictability does not hold for close queries

for free: stand-alone unconditional commitment from [OSVW13]
PUF-Com

S

R

a

b

opening of c

c=PUF-Com(b)

open a

open c

PUF-Com(a

b)

open c
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

\[ \text{idea \cite{MQU07,CGS08}} \]

\[ a_b \xrightarrow{\text{ECC(opening)}} \]

Problem 2: Adv queries with strings that are “close” to the actual opening

unpredictability does not hold for close queries

for free: stand-alone unconditional commitment from \cite{OSVW13}

PUF-Com

\[ \text{Problem 2: Adv queries with strings that are “close” to the actual opening} \]

\[ c=\text{PUF-Com}(b) \]

\[ \text{PUF-Com}(a_b) \]

\[ \text{open } a_b \]

\[ \text{open } c \]
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07,CGS08]

\[ a_b \rightarrow \text{ECC(opening)} \]

Problem 2: Adv queries with strings that are "close" to the actual opening

unpredictability does not hold for close queries

for free: stand-alone unconditional commitment from [OSVW13]

PUF-Com

token

we construct: stand-alone unconditional commitment from malicious tokens

\[ S \rightarrow c=\text{PUF-Com}(b) \]

\[ \text{PUF-Com}(a_b) \rightarrow R \]

\[ \text{open } a_b \rightarrow \text{open } c \]
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07, CGS08]

\(ab\) ECC(opening)

For free: stand-alone unconditional commitment from [OSVW13] PUF-Com

token

Problem 2: Adv queries with strings that are "close" to the actual opening

we construct: stand-alone unconditional commitment from malicious tokens

### Diagram

- **Sender (S)**:
  - Input: \(ab\)
  - Output: \(PUF-Com(ab)\)

- **Receiver (R)**:
  - Input: \(c = PUF-Com(b)\)
  - Output: Opened values: \(ab\) and \(c\)
Extractable Commitment from (malicious) PUFs

Problem 1: Adv can query with 0/1

idea [MQU07, CGS08]

ab ECC(opening)

token

no unconditional unpredictability for free: prevent uncontrolled access to the (stateless) token

for free: stand-alone unconditional commitment from [OSVW13] PUF-Com

Problem 2: Adv queries with strings that are "close" to the actual opening we construct: stand-alone unconditional commitment from malicious tokens

S

R

c=PUF-Com(b)

open ab

open c

open ECC(opening)
Conclusion

• black-box compiler any extractable commitments => UC-commitments

• Extractable commitments from Malicious PUFs => the first unconditional UC-security with PUFs

• Extractable commitments from Stateless token admitting arbitrary malicious adversary => the first unc. UC-secure protocol with stateless tokens. Complete the picture of unconditional UC security with stateless tokens.

• Unconditional OT with malicious PUFs??
Thanks