Zero-knowledge Proofs
and lookup tables

Arantxa Zapico
Ethereum Foundation

ASCRYPTO. Quito, October 2023
Table of Contents

1. Proof Systems
   1. Definition
   2. Security
   3. Zero-knowledge
   4. Examples!
2. Lookup Tables
   1. Definition
   2. Importance
   3. Examples
Proof Systems
Prover
Prover

Verifier
Something is true

Pedrinho

Valeria
Something is true

Pedrinho

Valeria
Something is true

Pedrinho

Something

Valeria
Something is true

Pedrinho

Valeria
Something is true

Pedrinho

Something

Valeria
Something is true
Examples of provers and verifiers
Examples of provers and verifiers

Me

Gmail
Examples of provers and verifiers

Google Cloud

Mobil Phone
Examples of provers and verifiers

You

Security at Club
Examples of provers and verifiers

Cryptobro

Block Builder
Something is true

Pedrinho

Something

Valeria

X

✓
Something is true

Pedrinho

Valeria

Completeness
Completeness

If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts
Something is true

Completeness
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

Soundness
Completeness  
If Something is indeed true and both Prover and Verifier follow the procedure, Verifier accepts

Soundness  
If something is false, then Verifier rejects with overwhelming probability


**Completeness**

If *Something* is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts.

**Soundness**

If *something* is false, then Verifier rejects with overwhelming probability.

**Zero-Knowledge**

*Something* is true.
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts.

If something is false, then Verifier rejects with overwhelming probability.

The Verifier does not learn anything but the truth of Something.
Something
Something
R is a PT decidable relation
$R = \{(x, w): \ldots\}$ is a PT decidable relation
$R = \{(x, w) : \ldots\}$ is a PT decidable relation

Something is true
\( R = \{(x, w) : \ldots\} \) is a PT decidable relation

\[ x \in \mathcal{L}_R \]
$R = \{(x, w) : \ldots \}$ is a PT decidable relation

$x \in \mathcal{L}_R$

$\mathcal{L}_R = \{x \ s.t. \ \exists w \ s.t. \ (x, w) \in R\}$
You

Security at Club
$R = \{(x, w) : x \text{ is a name and } w \text{ an age above } 18\}$
\[ R = \{(x, w) : x \text{ is a name and } w \text{ an age above 18}\} \]

"I am in \(\mathcal{L}_R\)": there exists a \(w\) (my age) such that \((\text{me}, w) \in R\)
Something is true

Completeness  If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

Soundness  If something is false, then Verifier rejects with overwhelming probability

Zero-Knowledge  The Verifier does not learn anything but the truth of Something
$R = \{(x, w) : \text{something}\}$

**Completeness**
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts.

**Soundness**
If something is false, then Verifier rejects with overwhelming probability.

**Zero-Knowledge**
The Verifier does not learn anything but the truth of Something.
Completeness
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

Soundness
If something is false, then Verifier rejects with overwhelming probability

Zero-Knowledge
The Verifier does not learn anything but the truth of Something

\[ R = \{(x, w) : \text{something}\} \]
\[ R = \{ (x, w) : something \} \]

\[ x, R \]

**Completeness**
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

**Soundness**
If *something* is false, then Verifier rejects with overwhelming probability

**Zero-Knowledge**
The Verifier does not learn anything but the truth of *Something*
Pedrinho\((x, w), R\)

Valeria \((x, R)\)
Pedrinho((x, w), R)

Valeria (x, R)

Probabilistic Polynomial Time Algorithms
Probabilistic Polynomial Time Algorithms

\textbf{P}rover\((srs, (x, w))\)

\textbf{V}erifier \((srs, x)\)
Prover($srs, (x, w))$

Verifier ($srs, x$)
\[(srs, \tau) \leftarrow \mathcal{K}(\lambda)\]

Prover\((srs, (x, w))\)

Verifier\((srs, x)\)
\[ R = \{(x, w) : \text{something}\} \]

Prover\((srs, (x, w))\)

Verifier \((srs, x)\)
Completeness  
If something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts.

Soundness  
If something is false, then Verifier rejects with overwhelming probability.

Zero-Knowledge  
The Verifier does not learn anything but the truth of Something.
Examples of provers and verifiers

Google Cloud

Mobil Phone
Examples of provers and verifiers

You

Security at Club
(Perfect) Completeness
(Perfect) Completeness

If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts
If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts...
(Perfect) Completeness

If \( x \in \mathcal{L}_R \) and both, Prover and Verifier, follow the procedure, Verifier accepts

\[
Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{K}(\lambda) \right] = 1
\]

\[
\pi \leftarrow \mathcal{P}(srs, (x, w))
\]

= 1
(Perfect) Completeness

If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$(s_{rs}, \tau) \leftarrow \mathcal{H}(\lambda)$$
If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$\pi \leftarrow \mathcal{P}(srs, (x, w))$$
(Perfect) Completeness

If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$(srs, \tau) \leftarrow \mathcal{K}(\lambda)$$

; $\pi \leftarrow \mathcal{P}(srs, (x, w))$$
(Perfect) **Completeness**

If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$Pr \left[ \mathcal{V}(srs, x, \pi) = 1 \right];$$
(Perfect) Completeness

If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad \pi \leftarrow \mathcal{P}(srs, (x, w)); \quad (srs, \tau) \leftarrow \mathcal{K}(\lambda) \right] = 1$$
(Perfect) **Completeness**

If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$\Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad \begin{array}{l} (srs, \tau) \leftarrow \mathcal{K}(\lambda) \\ \pi \leftarrow \mathcal{P}(srs, (x, w)) \end{array} \right] = 1$$
If $x \in \mathcal{L}_R$ and both, Prover and Verifier, follow the procedure, Verifier accepts

$$Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{K}(\lambda); \quad \pi \leftarrow \mathcal{P}(srs, (x, w)) \right] = 1$$
Completeness
If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

Soundness
If something is false, then Verifier rejects with overwhelming probability

Zero-Knowledge
The Verifier does not learn anything but the truth of Something

\[ R = \{(x, w) : something\} \]

\[ \text{Prover}(srs, (x, w)) \xrightarrow{\pi} \text{Verifier}(srs, x) \]
Complements: $R = \{(x, w) : \text{something}\}$

Completeness: $Pr\left(\forall (srs, x, \pi) = 1; \pi \leftarrow \mathcal{P}(srs, (x, w))\right) = 1$

Soundness: If something is false, then Verifier rejects with overwhelming probability

Zero-Knowledge: The Verifier does not learn anything but the truth of Something
(Computational) **Soundness**

If *something* is false, then Verifier rejects with overwhelming probability.
(Computational) Soundness

If $x \notin \mathcal{L}_R$, then Verifier rejects with overwhelming probability
(Computational) **Soundness**

If $x \notin \mathcal{L}_R$, then Verifier rejects with overwhelming probability.

If $\not\exists w \text{ s.t. } (x, w) \in R$, then Verifier rejects with overwhelming probability.
(Computational) **Soundness**

If $x \notin \mathcal{L}_R$, then Verifier rejects with overwhelming probability

If $\nexists \ w \ s.t. \ (x, w) \in R$, then Verifier rejects with overwhelming probability

$$
\Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{K} \quad (x, \pi) \leftarrow \mathcal{A}(srs) \right] \leq \text{negl}(\lambda)
$$
(Computational) **Soundness**

If \( x \notin \mathcal{L}_R \), then Verifier rejects with overwhelming probability.

If \( \not\exists \ w \text{ s.t. } (x, w) \in R \), then Verifier rejects with overwhelming probability.

\[
(srs, \tau) \leftarrow \mathcal{K}
\]
(Computational) **Soundness**

If \( x \notin \mathcal{L}_R \), then Verifier rejects with overwhelming probability.

If \( \nexists w \text{ s.t. } (x, w) \in R \), then Verifier rejects with overwhelming probability.

\[
(x, \pi) \leftarrow \mathcal{A}(srs)
\]
(Computational) **Soundness**

If $x \notin \mathcal{L}_R$, then Verifier rejects with overwhelming probability

If $\nexists \ w \ s.t. \ (x, w) \in R$, then Verifier rejects with overwhelming probability

$$
Pr \left[ \mathcal{V}(srs, x, \pi) = 1 \right];
$$
(Computational) **Soundness**

If $x \not\in \mathcal{L}_R$, then Verifier rejects with overwhelming probability

If $\nexists \; w \cdot (x, w) \in R$, then Verifier rejects with overwhelming probability

$$Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{K} \quad (x, \pi) \leftarrow \mathcal{A}(srs) \right] \leq \text{negl}(\lambda)$$
(Computational) Soundness

If \( x \notin \mathcal{L}_R \), then Verifier rejects with overwhelming probability

If \( \nexists \ w \text{ s.t. } (x, w) \in R \), then Verifier rejects with overwhelming probability

\[
Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{K} \quad (x, \pi) \leftarrow \mathcal{A}(srs) \right] \leq \text{negl}(\lambda)
\]
(Computational) **Soundness**

If $x \notin \mathcal{L}_R$, then Verifier rejects with overwhelming probability

If $\not\exists \ w \ s.t. \ (x, w) \in R$, then Verifier rejects with overwhelming probability

$$\Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \ (srs, \tau) \leftarrow \mathcal{K} \ (x, \pi) \leftarrow \mathcal{A}(srs) \right] \leq \text{negl}(\lambda)$$

We are actually talking about arguments
Completeness

\[ Pr \left[ \forall (srs, x, \pi) = 1; \quad ^{(srs, \tau)} \leftarrow \mathcal{K}(\lambda); \quad \pi \leftarrow \mathcal{P}(srs, (x, w)) \right] = 1 \]

Soundness

If \textit{something} is false, then Verifier rejects with overwhelming probability

Zero-Knowledge

The Verifier does not learn anything but the truth of \textit{Something}
\[ R = \{ (x, w) : \text{something} \} \]

**Completeness**
\[
Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{H}(\lambda) \right] = 1
\]

**Soundness**
\[
Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \quad (srs, \tau) \leftarrow \mathcal{H}, (x, \pi) \leftarrow \mathcal{A}(srs) \right] \leq \text{negl}(\lambda)
\]

**Zero-Knowledge**
The Verifier does not learn anything but the truth of *Something*
Examples of provers and verifiers

Me

Gmail
Examples of provers and verifiers

There exists a password for this email address
Examples of provers and verifiers

There exists a password for this email address

Not enough!!!
I should know it
Knowledge-soundness

There exists a PT algorithm $\mathcal{E}$, the extractor, such that for every malicious prover $\mathcal{P}^*$:
Knowledge-soundness

There exists a PT algorithm $\mathcal{E}$, the extractor, such that for every malicious prover $\mathcal{P}^*$:

\[
Pr \left[ (x, w) \not\in R \land \forall (srs, x, \pi) = 1; (srs, \tau) \leftarrow \mathcal{K} ; (x, \pi) \leftarrow \mathcal{P}^*(srs) \right] \leq \text{negl}(\lambda)
\]

$w \leftarrow \mathcal{E}(srs, x, \pi)$
Knowledge-soundness

There exists a PT algorithm $\mathcal{E}$, the extractor, such that for every malicious prover $\mathcal{P}^*$:

$$\Pr \left[ (x, w) \notin R \land \forall (srs, x, \pi) = 1 \right] \leq \text{negl}(\lambda)$$
Knowledge-soundness

There exists a PT algorithm \( \mathcal{E} \), the extractor, such that for every malicious prover \( \mathcal{P}^* \):

\[
\Pr \left[ (x, w) \notin R \land \forall (srs, x, \pi) = 1 \mid (srs, \tau) \leftarrow \mathcal{K} ; (x, \pi) \leftarrow \mathcal{P}^*(srs) \land w \leftarrow \mathcal{E}(srs, x, \pi) \right] \leq \text{negl}(\lambda)
\]

An argument that satisfies knowledge-soundness is an argument of knowledge.
Completeness

\[
Pr \left[ \mathcal{V}(srs, x, \pi) = 1; \left( srs, \tau \right) \leftarrow \mathcal{K}(\lambda) \right] = 1
\]

\[
Pr \left[ (x, w) \notin R \land \mathcal{V}(srs, x, \pi) = 1; \right. \\
\left. \left( srs, \tau \right) \leftarrow \mathcal{K}; (x, \pi) \leftarrow \mathcal{P}^*(srs); w \leftarrow \mathcal{E}(srs, x, \pi) \right] \leq \text{negl}(\lambda)
\]

Knowledge-Soundness

Zero-Knowledge

The Verifier does not learn anything but the truth of Something

\[
R = \{(x, w) : \text{something}\}
\]
Zero-knowledge

The Verifier does not learn anything but the truth of *Something*
Zero-knowledge

The prover output is *almost random*, therefore, could be anything.
Zero-knowledge

There exists a PT simulator $S$, with access to the private information, such that for all $V^*$
There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

\[
Pr\left[\mathcal{V}^*(srs, \pi) = 1; \quad x \leftarrow \mathcal{V}^*(srs) \quad \pi \leftarrow \mathcal{P}(srs, (x, w))\right] \approx Pr\left[\mathcal{V}^*(srs, \pi_{\text{sim}}) = 1; \quad x \leftarrow \mathcal{V}^*(srs) \quad \pi_{\text{sim}} \leftarrow \mathcal{S}(srs, \tau, x)\right]
\]
There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

\[
\Pr \left[ \mathcal{V}^*(srs, \pi) = 1; \quad x \leftarrow \mathcal{V}^*(srs); \quad \pi \leftarrow \mathcal{P}(srs, (x, w)) \right] \approx 1
\]
Zero-knowledge

There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

$$\Pr \left[ \mathcal{V}^*(srs, \pi) = 1; \begin{cases} (srs, \tau) \leftarrow \mathcal{K} \\ x \leftarrow \mathcal{V}^*(srs) \\ \pi \leftarrow \mathcal{P}(srs, (x, w)) \end{cases} \right] \approx$$
Zero-knowledge

There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

$$
Pr \left[ \mathcal{V}^*(srs, \pi) = 1; \ x \leftarrow \mathcal{V}^*(srs)
\right. \left. \pi \leftarrow \mathcal{P}(srs, (x, w)) \right] \approx
Pr \left[ \mathcal{V}^*(srs, \pi_{\text{sim}}) = 1; \ x \leftarrow \mathcal{V}^*(srs)
\right. \left. \pi_{\text{sim}} \leftarrow \mathcal{S}(srs, \tau, x) \right]
$$
There exists a PT simulator $S$, with access to the private information, such that for all $V^*$

\[
Pr \left[ V^*(srs, \pi_{sim}) = 1; \quad (srs, \tau) \leftarrow K \right.
\]

\[
\left. x \leftarrow V^*(srs) \quad \pi_{sim} \leftarrow S(srs, \tau, x) \right]
\]
Zero-knowledge

There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

$$Pr \left[ \mathcal{V}^*(srs, \pi_{sim}) = 1; \quad (srs, \tau) \leftarrow \mathcal{K} \quad x \leftarrow \mathcal{V}^*(srs) \quad \pi_{sim} \leftarrow \mathcal{S}(srs, \tau, x) \right]$$
Zero-knowledge

There exists a PT simulator $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

$$
\Pr \left[ \begin{array}{c}
\mathcal{V}^*(srs, \pi) = 1; \\
(srs, \tau) \leftarrow \mathcal{H} \\
x \leftarrow \mathcal{V}^*(srs) \\
\pi \leftarrow \mathcal{P}(srs, (x, w))
\end{array} \right] \approx 
\Pr \left[ \begin{array}{c}
\mathcal{V}^*(srs, \pi_{\text{sim}}) = 1; \\
(srs, \tau) \leftarrow \mathcal{H} \\
x \leftarrow \mathcal{V}^*(srs) \\
\pi_{\text{sim}} \leftarrow \mathcal{S}(srs, \tau, x)
\end{array} \right]
$$
Completeness

\[ Pr \left[ \forall (srs, x, \pi) = 1; \pi \leftarrow P(srs, (x, w)) \right] = 1 \]

Knowledge-Soundness

\[ Pr \left[ (x, w) \notin R \land \forall (srs, x, \pi) = 1; \pi \leftarrow P(srs, (x, w)), w \leftarrow E(srs, x, \pi) \right] \leq negl(\lambda) \]

Zero-Knowledge

The Verifier does not learn anything but the truth of *Something*
Completeness

\[ Pr \left[ V(srs, x, \pi) = 1; \pi \leftarrow P(srs, (x, w)) \right] = 1 \]

Knowledge-Soundness

\[ Pr \left[ (x, w) \notin R \land V(srs, x, \pi) = 1; (srs, \tau) \leftarrow \mathcal{K}(\lambda); (x, \pi) \leftarrow A(srs); w \leftarrow \mathcal{E}(srs, x, \pi) \right] \leq \text{negl}(\lambda) \]

Zero-Knowledge

\[ Pr \left[ V'(srs, \pi_{sim}) = 1; x \leftarrow V'(srs); \pi \leftarrow P(srs, (x, w)) \right] \approx Pr \left[ V'(srs, \pi_{sim}) = 1; x \leftarrow V'(srs); \pi_{sim} \leftarrow S(srs, \tau, x) \right] \]

\[ R = \{(x, w) : \text{something}\} \]
Everything that can be proven (NP) can be proven in Zero-Knowledge
Knowledge of discrete log
Knowledge of discrete log

Discrete logs are hard to compute (in some groups)
Knowledge of discrete log

Discrete logs are hard to compute (in some groups)

Let $G$ be a cyclic group of order $q$ (prime) and $g$ be a generator.
Knowledge of discrete log

Let \( G \) be a cyclic group of order \( q \) (prime) and \( g \) be a generator.

\[
R = \{ (x, h) : x \in \mathbb{Z}_q \land h \in G \land h = g^x \}
\]
Knowledge of discrete log

Let $\mathbb{G}$ be a cyclic group of order $q$ (prime) and $g$ be a generator.

\[
R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\}
\]

Discrete logs are hard to compute (in some groups)

Famous secret-key, public-key couple:

\[
sk \leftarrow \mathbb{Z}_q, \quad pk = g^{sk}
\]
Knowledge of discrete log - Schnorr
Knowledge of discrete log - Schnorr
Knowledge of discrete log - Schnorr

\[ R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]
Knowledge of discrete log - Schnorr

\[ R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]

\[ P = ((h, \mathbb{G}), (x, h)) \]

\[ V = \]
Knowledge of discrete log - Schnorr

\[ R = \{ (x, h) : x \in \mathbb{Z}_q \land h \in G \land h = g^x \} \]

\[ P ((h, G), (x, h)) \]

\[ V ((g, G), h) \]
Knowledge of discrete log - Schnorr

\[ R = \{(x,h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]

\[ P \quad ((h, \mathbb{G}), (x, h)) \]

\[ V \quad ((g, \mathbb{G}), h) \]

\[ r \leftarrow \mathbb{Z}_q \]

\[ u = g^r \]
Knowledge of discrete log - Schnorr

\[
R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\}
\]

\[\mathcal{P}\]

\[((h, \mathbb{G}), (x, h))\]

\[r \leftarrow \mathbb{Z}_q\]

\[u = g^r\]

\[\mathcal{V}\]

\[((g, \mathbb{G}), h)\]

\[u\]
Knowledge of discrete log - Schnorr

\[ R = \{ (x, h) : x \in \mathbb{Z}_q \land h \in G \land h = g^x \} \]

\[ \mathcal{P} \]

\[ ((h, G), (x, h)) \]

\[ r \leftarrow \mathbb{Z}_q \]

\[ u = g^r \]

\[ \mathcal{V} \]

\[ ((g, G), h) \]

\[ u \]

\[ c \leftarrow \mathbb{Z}_q \]
Knowledge of discrete log - Schnorr

\[ R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]

\[ \mathcal{P} \]

- \(((h, \mathbb{G}), (x, h))\)
- \(r \leftarrow \mathbb{Z}_q\)
- \(u = g^r\)

\[ \mathcal{V} \]

- \(((g, \mathbb{G}), h)\)
- \(c \leftarrow \mathbb{Z}_q\)
- \(c\)
Knowledge of discrete log - Schnorr

\[ R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]

**P**

\[ ((h, \mathbb{G}), (x, h)) \]

- \( r \leftarrow \mathbb{Z}_q \)
- \( u = g^r \)

**V**

\[ ((g, \mathbb{G}), h) \]

- \( c \leftarrow \mathbb{Z}_q \)

\[ z = r + cx \]
Knowledge of discrete log - Schnorr

\[ R = \{ (x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x \} \]

\[ ((h, \mathbb{G}), (x, h)) \quad \quad \quad \quad \quad (g, \mathbb{G}), h) \]

\[ r \leftarrow \mathbb{Z}_q \]
\[ u = g^r \]
\[ z = r + cx \]

\[ u \]
\[ c \]
\[ z \]
Knowledge of discrete log - Schnorr

\[ R = \{(x, h) : x \in \mathbb{Z}_q \land h \in \mathbb{G} \land h = g^x\} \]

\[ \mathcal{P} \]

\[ ((h, \mathbb{G}), (x, h)) \]

\[ r \leftarrow \mathbb{Z}_q \]

\[ u = g^r \]

\[ z = r + cx \]

\[ \mathcal{V} \]

\[ ((g, \mathbb{G}), h) \]

\[ c \leftarrow \mathbb{Z}_q \]

\[ g^z = uh^c \]
Completeness
Completeness

Theorem: The scheme satisfies completeness
Completeness

Theorem: The scheme satisfies completeness

\[ g^z = uh^c \]
Completeness

Theorem: The scheme satisfies completeness

$$g^z = uh^c$$
Completeness

Theorem: The scheme satisfies completeness

\[ g^z = uh^c \]

\[ g^{r+cx} = uh^c \]
Completeness

Theorem: The scheme satisfies completeness

\[ g^z = uh^c \]

\[ g^{r+cx} = uh^c \]
Completeness

Theorem: The scheme satisfies completeness

\[
g^z = uh^c
\]

\[
g^{r+cx} = uh^c
\]

\[
g^{r+cx} = g^{r}h^c
\]
Completeness

Theorem: The scheme satisfies completeness

\[ g^z = uh^c \]

\[ g^{r+cx} = uh^c \]

\[ g^{r+cx} = g^r h^c \]
Completeness

Theorem: The scheme satisfies completeness

\[ g^z = uh^c \]

\[ g^{r+cx} = uh^c \]

\[ g^{r+cx} = g^r h^c \]

\[ g^{r+cx} = g^r (g^x)^c \]
Knowledge-soundness
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

$$u_1 \leftarrow \mathbb{Z}_q$$

$$u_2 \leftarrow \mathbb{Z}_q$$

$$x = z_1 - z_2$$

$$c_1 - c_1 \in \mathbb{Z}_q$$
Knowledge-soundness

Let $P^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $E$ as follows:

1. $E$ runs prover $P^*$ to obtain initial message $u$

   $c_1 \leftarrow \mathbb{Z}_q$

   $z_1 \leftarrow \mathcal{P}^*$

   $c_2 \leftarrow \mathbb{Z}_q$

   $z_2 \leftarrow \mathcal{P}^*$

   $x = z_1 - z_2$

   $c_1 - c_1 \in \mathbb{Z}_q$
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

- $\mathcal{E}$ runs prover $\mathcal{P}^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $\mathcal{P}^*$ and obtains response $z_1$
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

- $\mathcal{E}$ runs prover $\mathcal{P}^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $\mathcal{P}^*$ and obtains response $z_1$
- Rewind $\mathcal{P}^*$ to its state after $u$
Knowledge-soundness

Let $P^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $E$ as follows:

- $E$ runs prover $P^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $P^*$ and obtains response $z_1$
- Rewind $P^*$ to its state after $u$
- Send $c_2 \leftarrow \mathbb{Z}_q$ and get response $z_2$
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

- $\mathcal{E}$ runs prover $\mathcal{P}^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $\mathcal{P}^*$ and obtains response $z_1$
- Rewind $\mathcal{P}^*$ to its state after $u$
- Send $c_2 \leftarrow \mathbb{Z}_q$ and get response $z_2$
- Output $x = \frac{z_1 - z_2}{c_1 - c_1} \in \mathbb{Z}_q$
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

- $\mathcal{E}$ runs prover $\mathcal{P}^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $\mathcal{P}^*$ and obtains response $z_1$
- Rewind $\mathcal{P}^*$ to its state after $u$
- Send $c_2 \leftarrow \mathbb{Z}_q$ and get response $z_2$
- Output $x = \frac{z_1 - z_2}{c_1 - c_1} \in \mathbb{Z}_q$

With probability $\epsilon^2$, $g^{z_1} = uh^{c_1} \land g^{z_2} = uh^{c_2}$. Then,
Knowledge-soundness

Let $\mathcal{P}^*$ be a malicious prover that convinces the verifier with probability $\epsilon$. We construct the extractor $\mathcal{E}$ as follows:

- $\mathcal{E}$ runs prover $\mathcal{P}^*$ to obtain initial message $u$
- Send $c_1 \leftarrow \mathbb{Z}_q$ to $\mathcal{P}^*$ and obtains response $z_1$
- Rewind $\mathcal{P}^*$ to its state after $u$
- Send $c_2 \leftarrow \mathbb{Z}_q$ and get response $z_2$
- Output $x = \frac{z_1 - z_2}{c_1 - c_1} \in \mathbb{Z}_q$

With probability $\epsilon^2$, $g^{z_1} = uh^{c_1} \land g^{z_2} = uh^{c_2}$. Then,

$$
\frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \rightarrow \frac{g^{z_1}}{h^{c_2}} = \frac{h^{c_1}}{h^{c_2}} \rightarrow g^{z_1 - z_2} = h^{c_1 - c_2} \rightarrow g^{z_1 - z_2} = (g^x)^{(c_1 - c_2)} \rightarrow \frac{z_1 - z_2}{c_1 - c_2} = (g^x)
$$
Honest-Verifier Zero-knowledge
Honest-Verifier Zero-knowledge

We need to construct a simulator $S(h)$ that outputs an accepting proof with the same distribution than an honestly generated one (random)
Honest-Verifier Zero-knowledge

We need to construct a simulator $S(h)$ that outputs an accepting proof with the same distribution than an honestly generated one (random)

- $z \leftarrow \mathbb{Z}_q$
Honest-Verifier Zero-knowledge

We need to construct a simulator \( S(h) \) that outputs an accepting proof with the same distribution than an honestly generated one (random)

\[
- z \leftarrow \mathbb{Z}_q \\
- c \leftarrow \mathbb{Z}_q
\]
Honest-Verifier Zero-knowledge

We need to construct a simulator \( S(h) \) that outputs an accepting proof with the same distribution than an honestly generated one (random)

\[
\begin{align*}
- & \quad z \leftarrow \mathbb{Z}_q \\
- & \quad c \leftarrow \mathbb{Z}_q \\
- & \quad u = \frac{g^z}{h^c} \\
- & \quad g^z = uh^c
\end{align*}
\]
Honest-Verifier Zero-knowledge

We need to construct a simulator $\mathcal{S}(h)$ that outputs an accepting proof with the same distribution than an honestly generated one (random)

- $z \leftarrow \mathbb{Z}_q$
- $c \leftarrow \mathbb{Z}_q$
- $u = \frac{g^z}{h^c}$
- $g^z = uh^c$
- Output $(u, c, z)$
Lookup Tables
$\vec{T} = (v_1, v_2, v_3, \ldots, v_m)$

$C$ is a commitment to elements $s_i \in \vec{T}$
Importance

- Building blocks to many systems
- Efficiency: mostly do not depend on the size of the table
- Flexibility: zero-knowledge/succinctness/pre-computable
Some examples
Some examples

\[ \mathbf{T} = (18, 19, \ldots, 120) \]
Some examples

\[ \mathbf{T} = (18, 19, \ldots, 120) \]

C is your age
Some examples

C is your age

\[ \overrightarrow{T} = (18, 19, \ldots, 120) \]

\[ \overrightarrow{T} = \begin{array}{cc}
  x_1 & f(x_1) \\
  x_2 & f(x_2) \\
  \vdots & \vdots \\
  x_m & f(x_m)
\end{array} \]
Some examples

C is your age

C is \((x_i, y_i)\)

\[
\vec{T} = (18, 19, \ldots, 120)
\]

\[
\begin{array}{cc}
\hline
x_1 & f(x_1) \\
\vdots & \vdots \\
x_m & f(x_m) \\
\hline
\end{array}
\]
Some examples

$\vec{T} = (18, 19, \ldots, 120)$

$\vec{T} = \begin{bmatrix} x_1 & f(x_1) \\ x_2 & f(x_2) \\ \vdots & \vdots \\ x_m & f(x_m) \end{bmatrix}$

$\vec{T} = (user_1, \ldots, user_m)$

C is your age

C is $(x_i, y_i)$
Some examples

$\overrightarrow{T}$ is your age

$\overrightarrow{T} = (18, 19, \ldots, 120)$

$\overrightarrow{T} = \begin{pmatrix} x_1 & f(x_1) \\ x_2 & f(x_2) \\ \vdots & \vdots \\ x_m & f(x_m) \end{pmatrix}$

$\overrightarrow{T} = (user_1, \ldots, user_m)$

C is your age

C is $\left(x_i, y_i\right)$

C is my user name
Some examples

$\vec{T} = (18, 19, \ldots, 120)$

\[
\begin{bmatrix}
    x_1 & f(x_1) \\
x_2 & f(x_2) \\
  \vdots & \vdots \\
x_m & f(x_m)
\end{bmatrix}
\]

$\vec{T} = (\text{user}_1, \ldots, \text{user}_m)$
Membership proofs from Lookup tables

\[ \vec{T} = (user_1, \ldots, user_m) \]

C is my user name
Membership proofs from Lookup tables

I am an authorized member/
my name is on the list

C is my user name

\[ \vec{T} = (user_1, \ldots, user_m) \]
Membership proofs from Lookup tables

\[ \vec{T} = (user_1, \ldots, user_m) \]

C is my user name
Membership proofs from Lookup tables

\[ \vec{T} = (user_1, \ldots, user_m) \]

\[ sk \leftarrow \mathbb{Z}_q \]

C is my user name
Membership proofs from Lookup tables

C is my user name

\[ \vec{T} = (user_1, \ldots, user_m) \]

\[ sk \leftarrow \mathbb{Z}_q \]
\[ pk = g^{sk} \]
Membership proofs from Lookup tables

C is my user name

\[ sk \leftarrow \mathbb{Z}_q \]
\[ pk = g^{sk} \]
\[ \vec{T} = (pk_1, \ldots, pk_m) \]
Membership proofs from Lookup tables

C is my user name

\[ \begin{align*}
    sk & \leftarrow \mathbb{Z}_q \\
    pk &= g^{sk} \\
    \overrightarrow{T} &= (pk_1, \ldots, pk_m) \\
    C &= \text{Com}(pk) = g^{x+r.sk}
\end{align*} \]
Membership proofs from Lookup tables

C is my user name

\[ sk \leftarrow \mathbb{Z}_q \]
\[ pk = g^{sk} \]
\[ C = Com(pk) = g^{x+r.sk} \]

“\textit{I am authorized}”: 

\[ \vec{T} = (pk_1, \ldots, pk_m) \]
Membership proofs from Lookup tables

C is my user name

\[ sk \leftarrow \mathbb{Z}_q \]
\[ pk = g^{sk} \]
\[ C = Com(pk) = g^{x+r.sk} \]

\[ \overrightarrow{T} = (pk_1, \ldots, pk_m) \]

“I am authorized”:

1. Use a lookup table to prove in zero-knowledge C is a commitment to something in \( \overrightarrow{T} \)
Membership proofs from Lookup tables

C is my user name

\[ sk \leftarrow \mathbb{Z}_q \]
\[ pk = g^{sk} \]
\[ C = \text{Com}(pk) = g^{x+r.sk} \]
\[ \vec{T} = (pk_1, \ldots, pk_m) \]

“I am authorized”:

1. Use a lookup table to prove in zero-knowledge \( C \) is a commitment to something in \( \vec{T} \)
2. Use Schnorr to prove knowledge of the corresponding \( sk \)
Membership proofs from Lookup tables

- $C$ is my user name

$$\begin{align*}
sk & \leftarrow \mathbb{Z}_q \\
pk &= g^{sk} \\
\vec{T} &= (pk_1, \ldots, pk_m) \\
C &= \text{Com}(pk) = g^{x+r.sk}
\end{align*}$$

“I am authorized”:
1. Use a lookup table to prove in zero-knowledge $C$ is a commitment to something in $\vec{T}$
2. Use Schnorr to prove knowledge of the corresponding $sk$
3. It is me!
¡¡¡Gracias!!!

Obrigado!!

arantxa@ethereum.org

www.criptolatino.org
@criptolatinoOrg