Zero-knowledge Proofs

and lookup tables

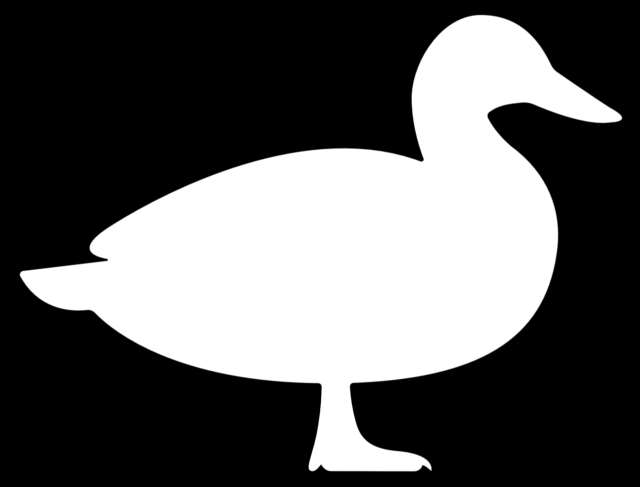
Arantxa Zapico
Ethereum Foundation

ASCRYPTO. Quito, October 2023

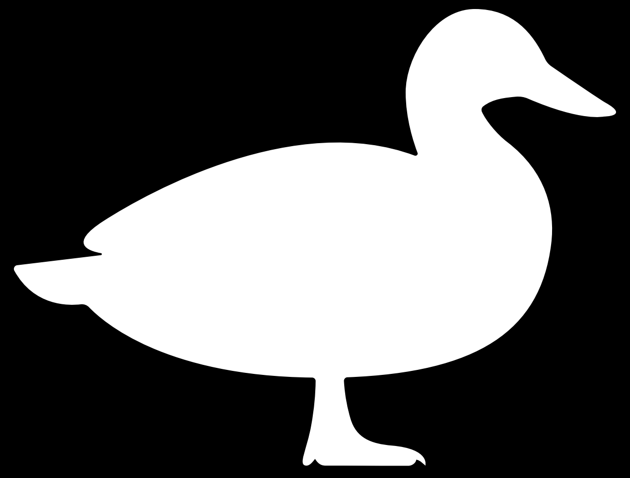
Table of Contents

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

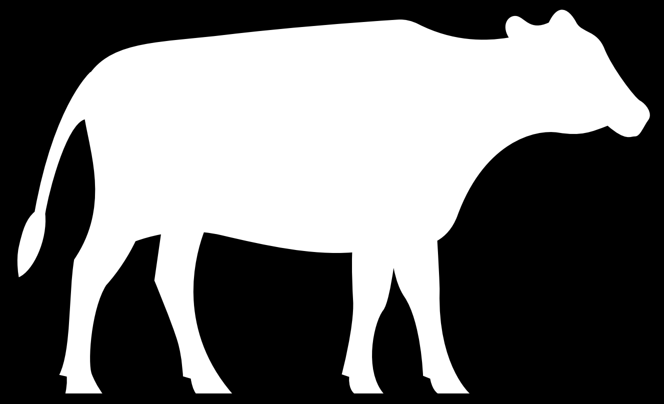
Proof Systems



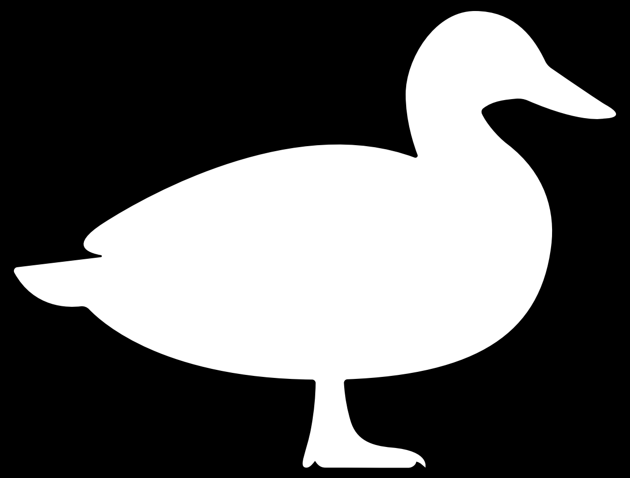
Prover



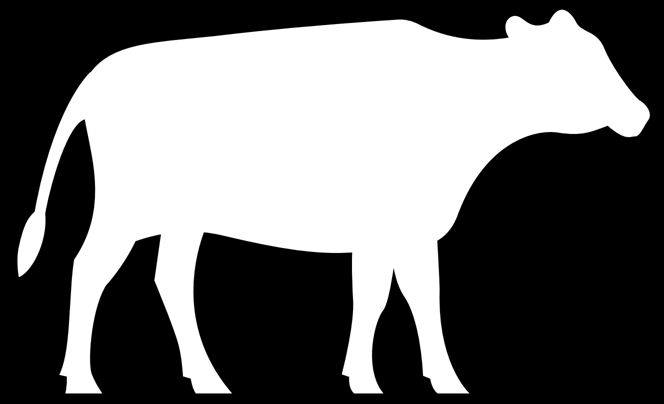
Prover



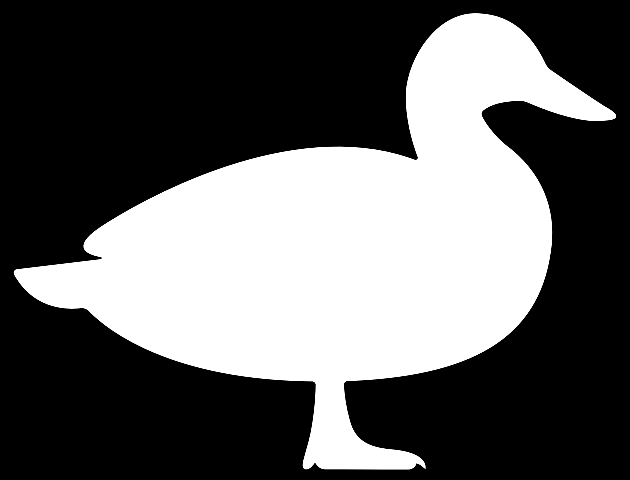
Verifier



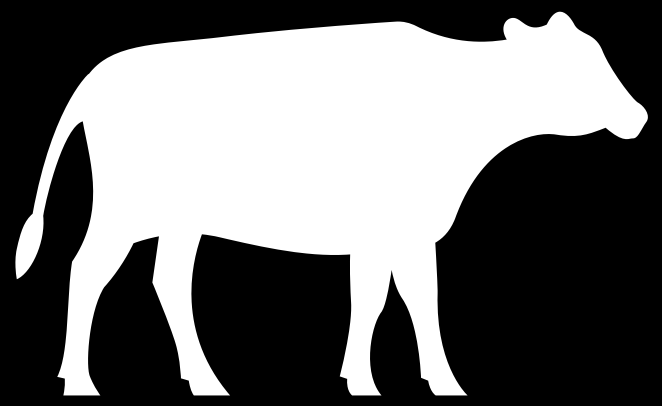
Peggy



Victor

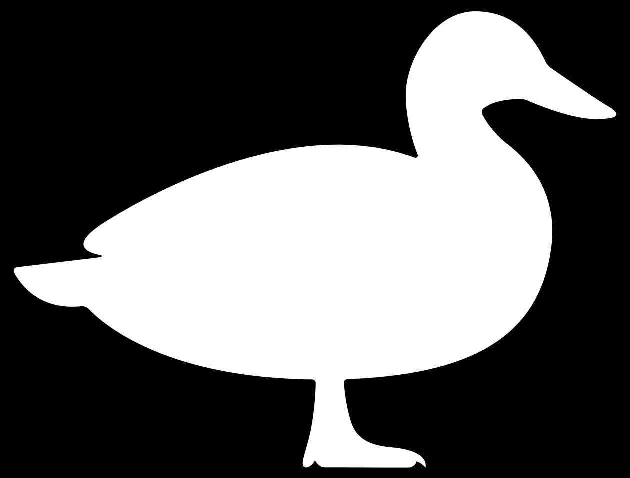


Pedrinho

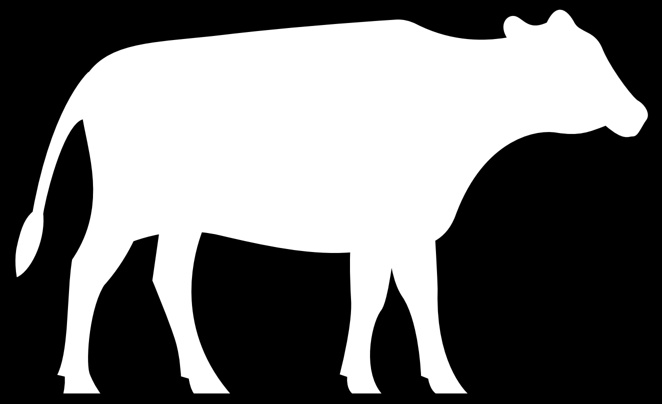


Valeria

Something is true



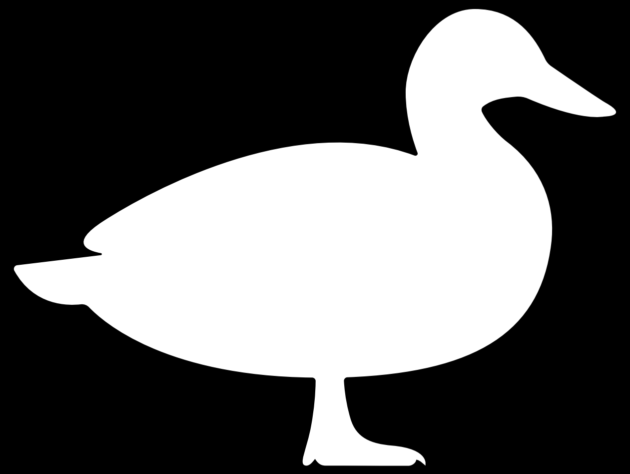
Pedrinho



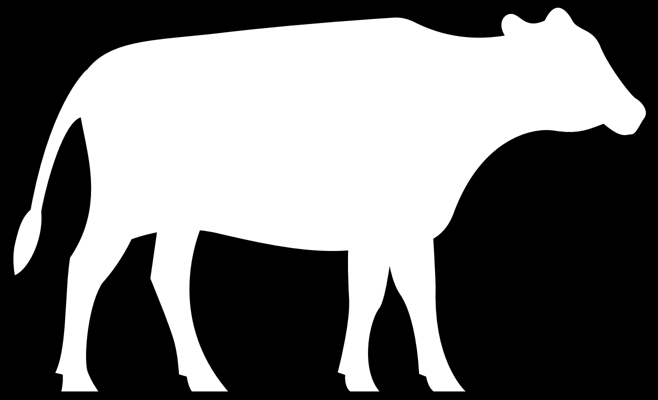
Valeria

Something is true

Something

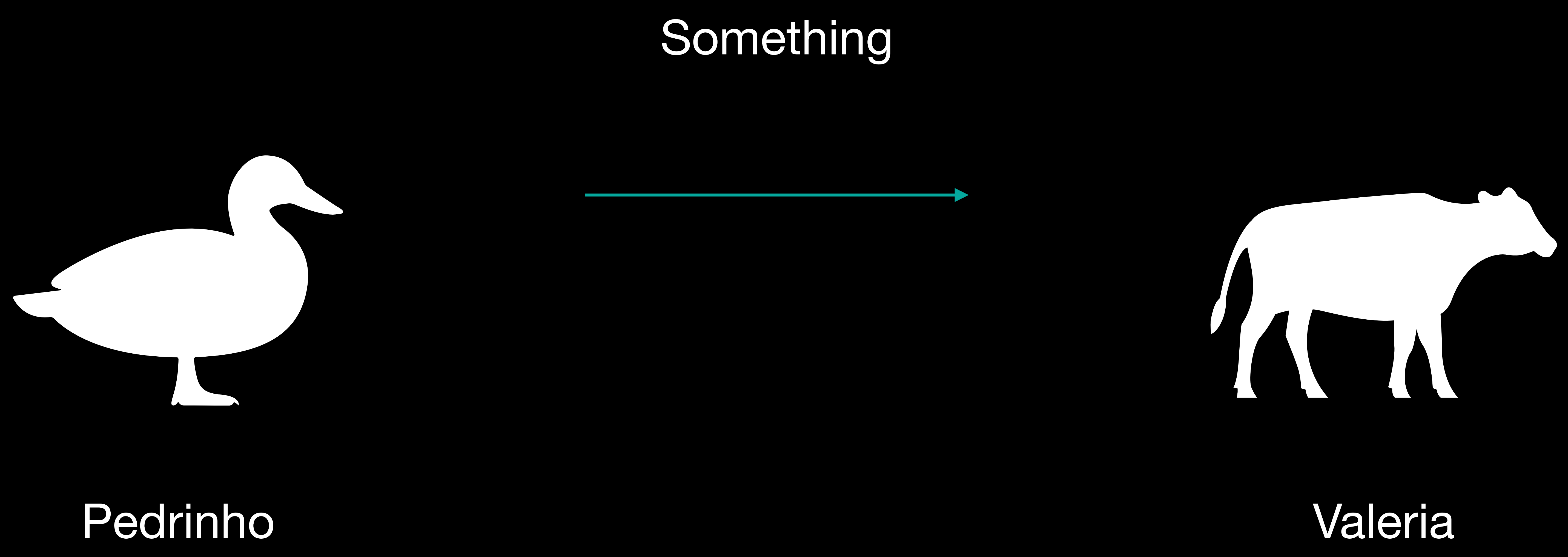


Pedrinho

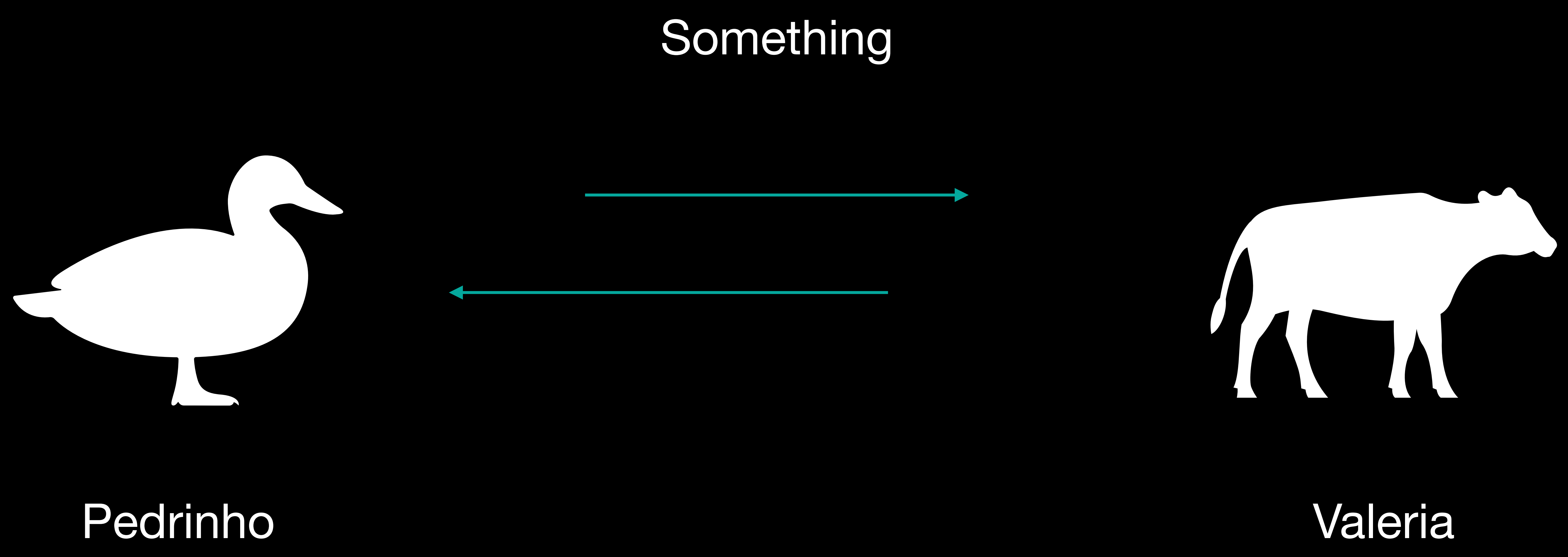


Valeria

Something is true

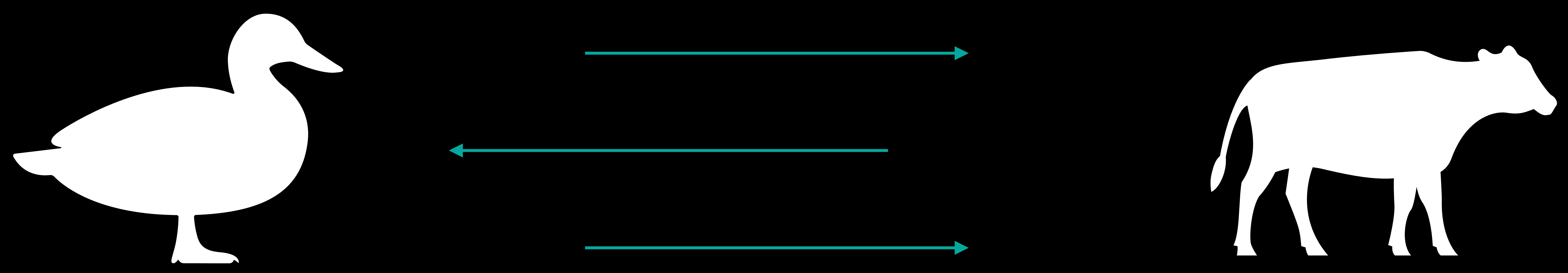


Something is true



Something is true

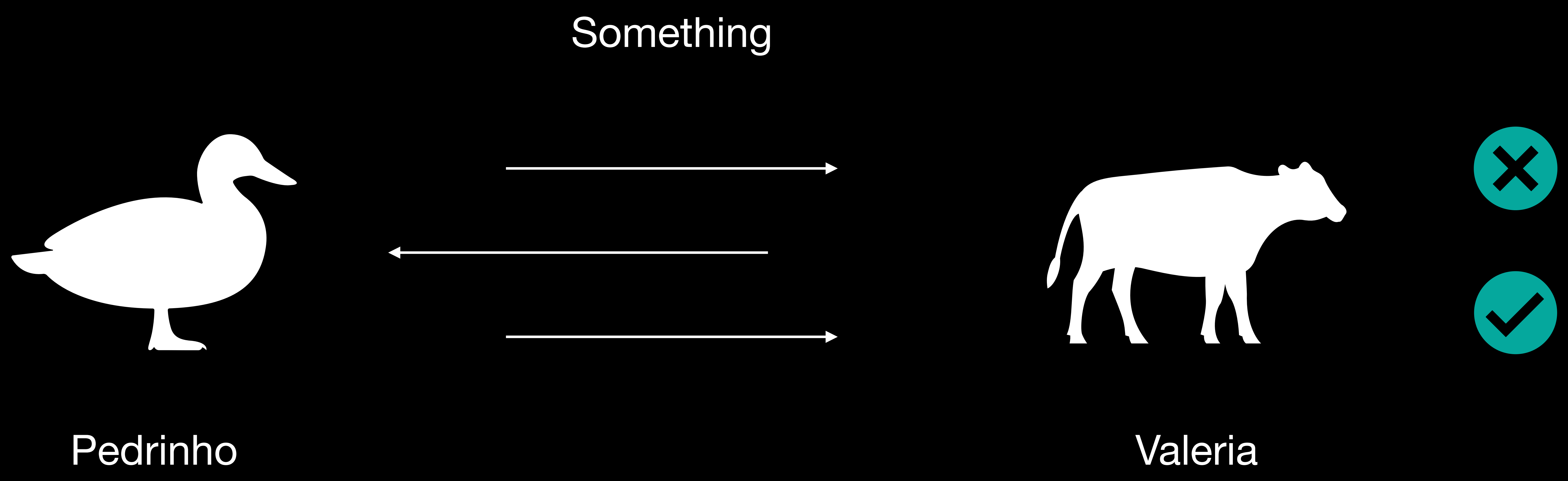
Something



Pedrinho

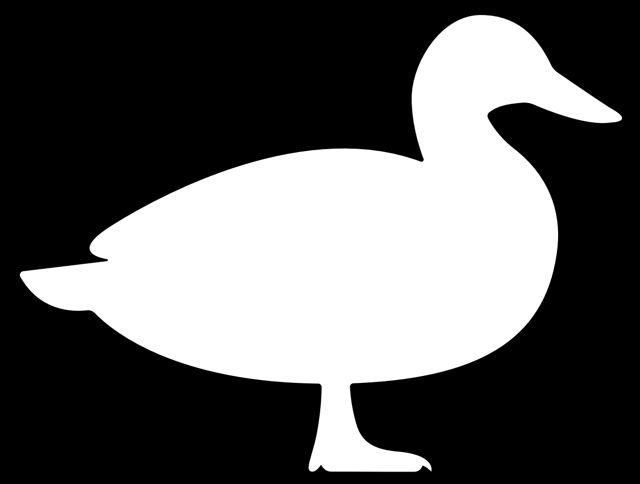
Valeria

Something is true

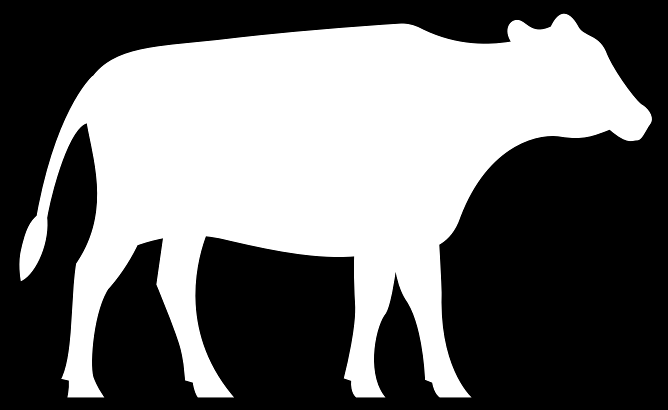


Examples of provers and verifiers

Examples of provers and verifiers

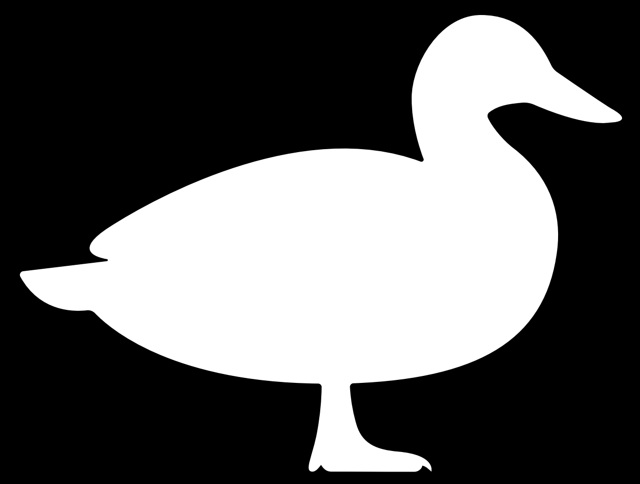


Me

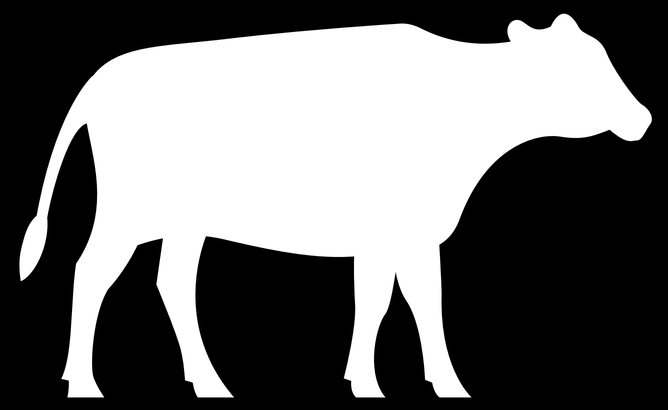


Gmail

Examples of provers and verifiers

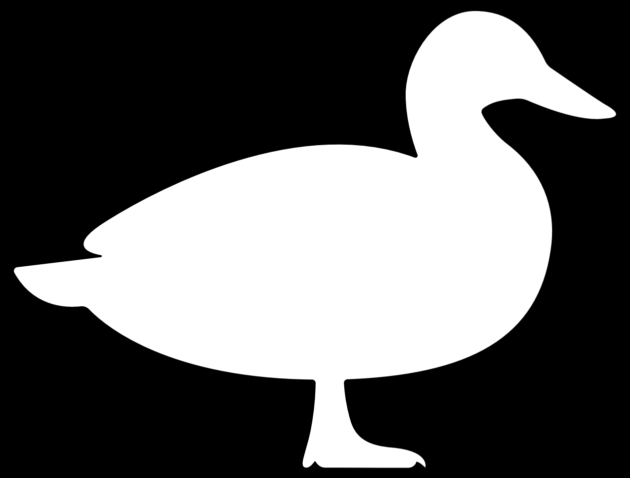


Google Cloud

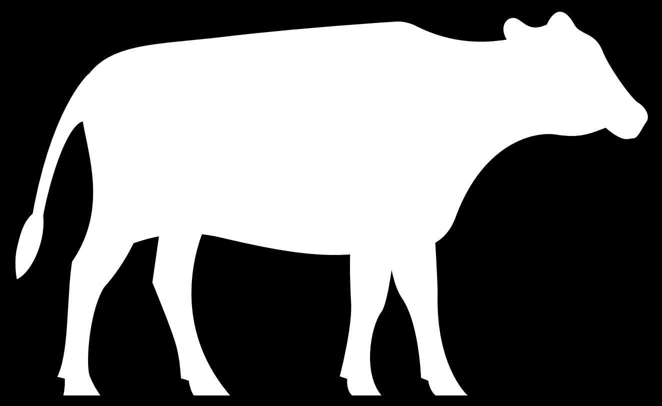


Mobil Phone

Examples of provers and verifiers

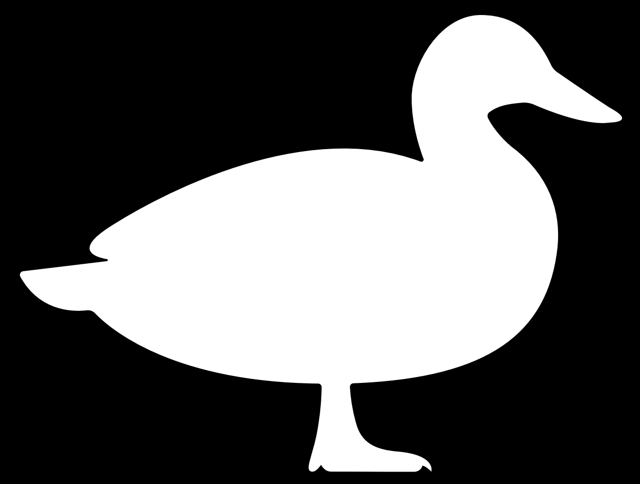


You

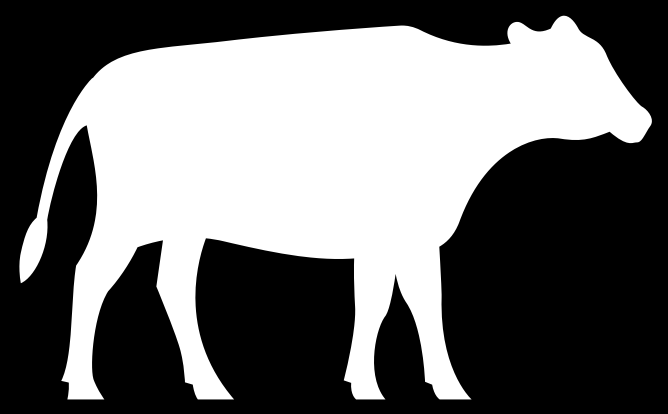


Security at Club

Examples of provers and verifiers

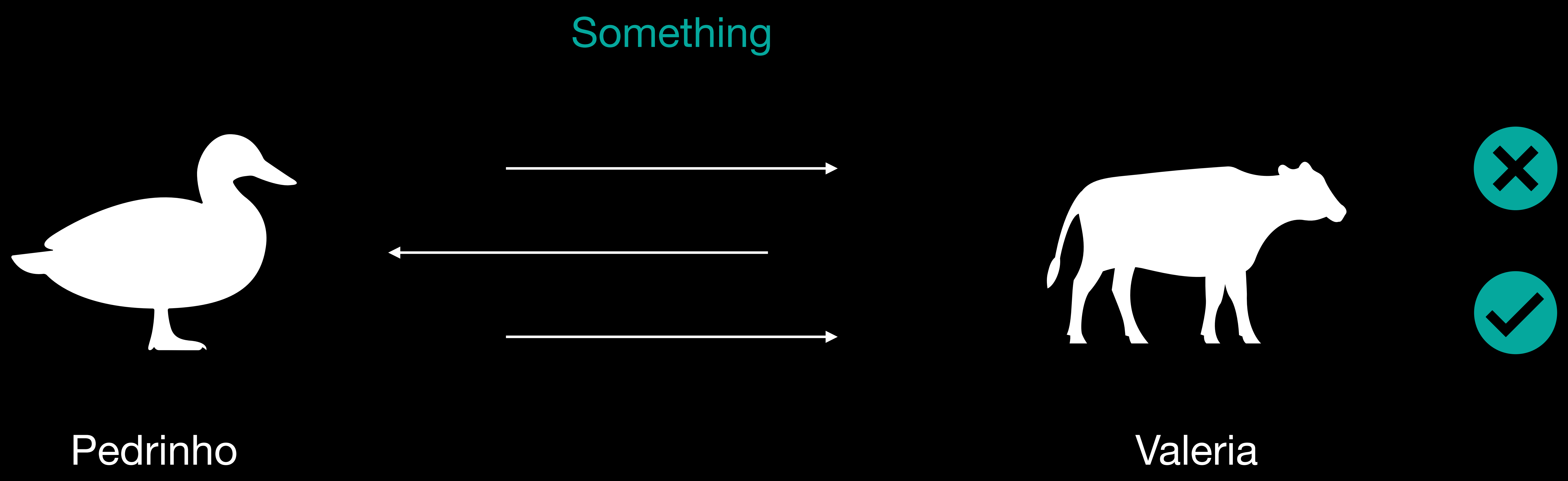


Cryptobro



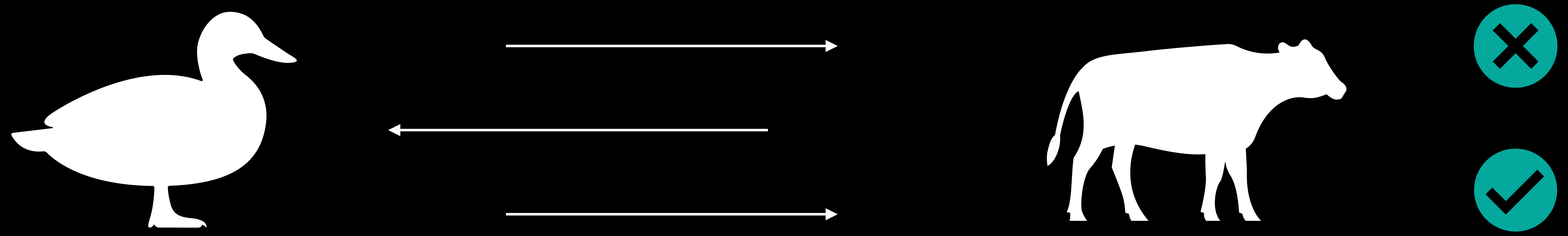
Block Builder

Something is true



Something is true

Something

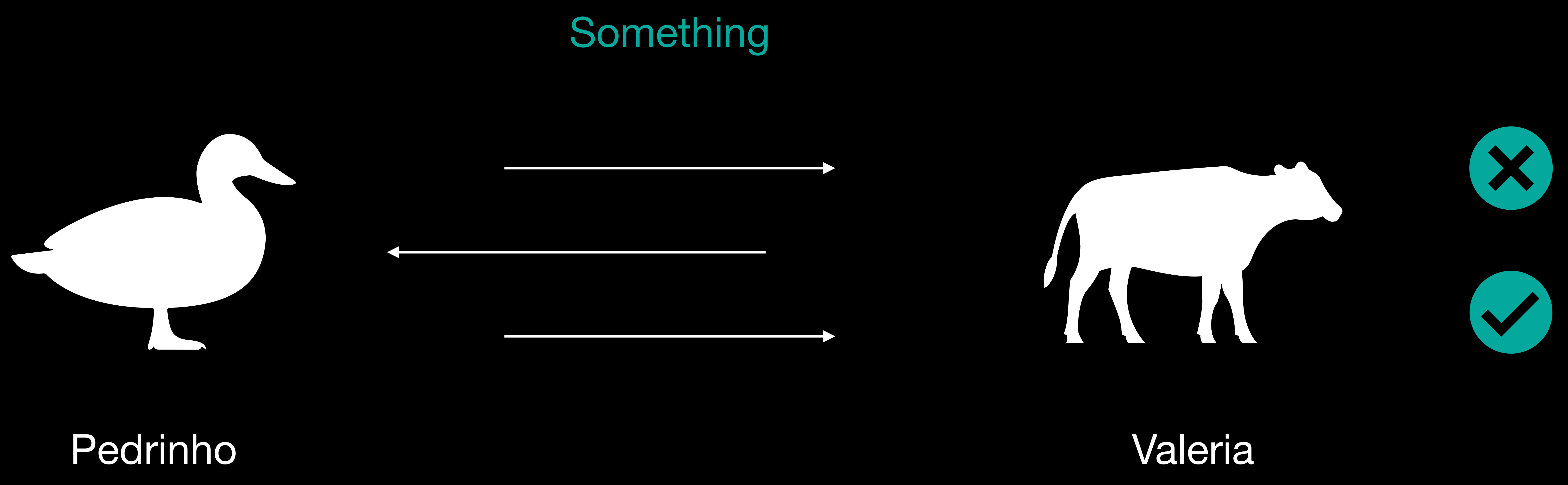


Pedrinho

Valeria

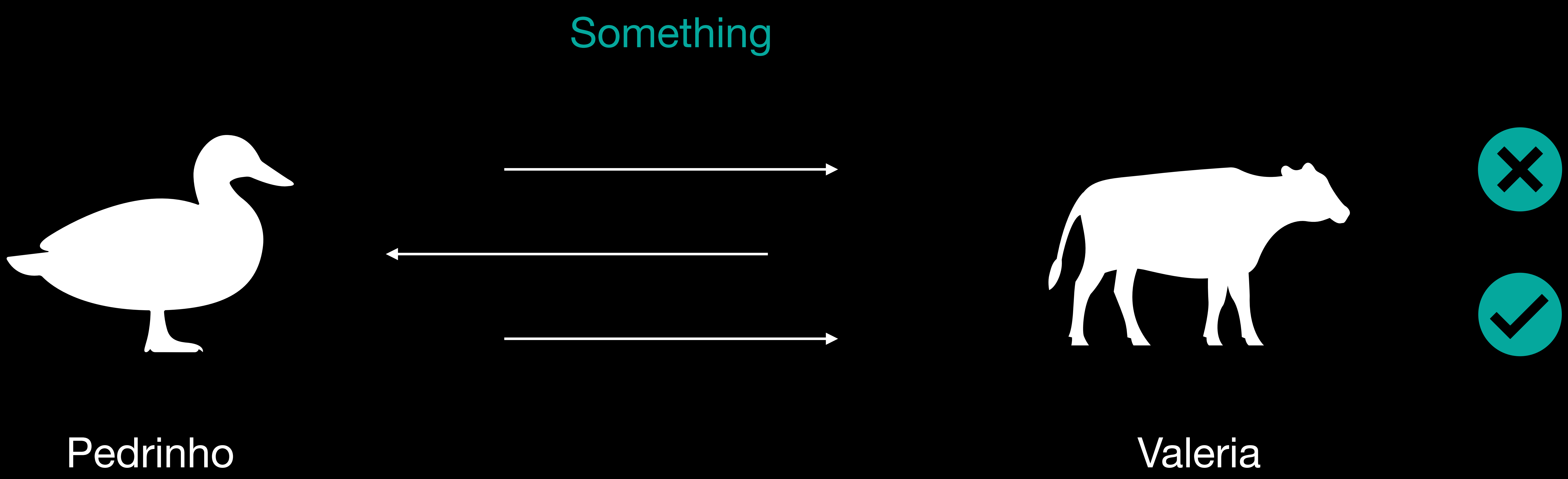
Completeness

Something is true



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

Something is true

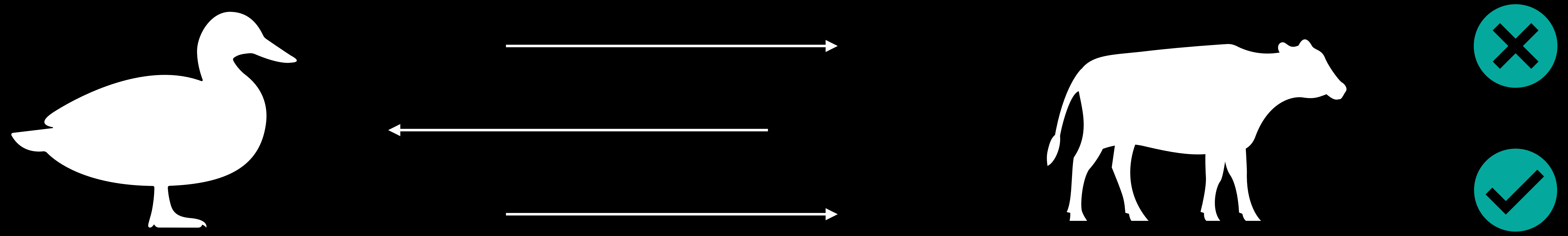


Completeness If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

Soundness

Something is true

Something



Pedrinho

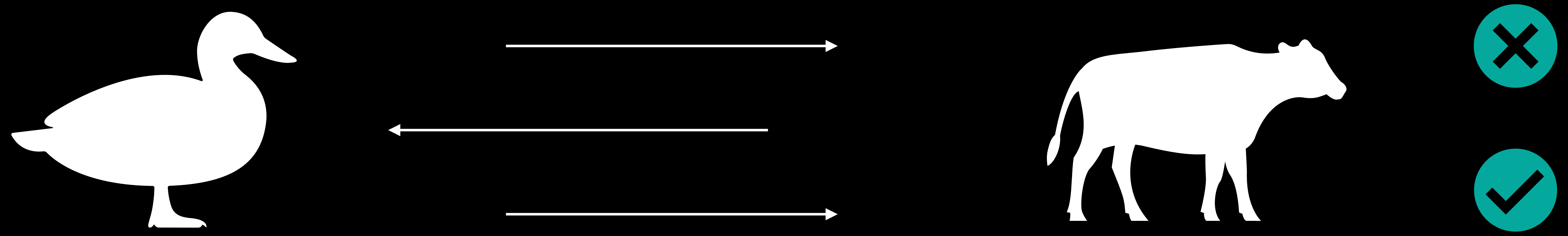
Valeria

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

Something is true

Something



Pedrinho

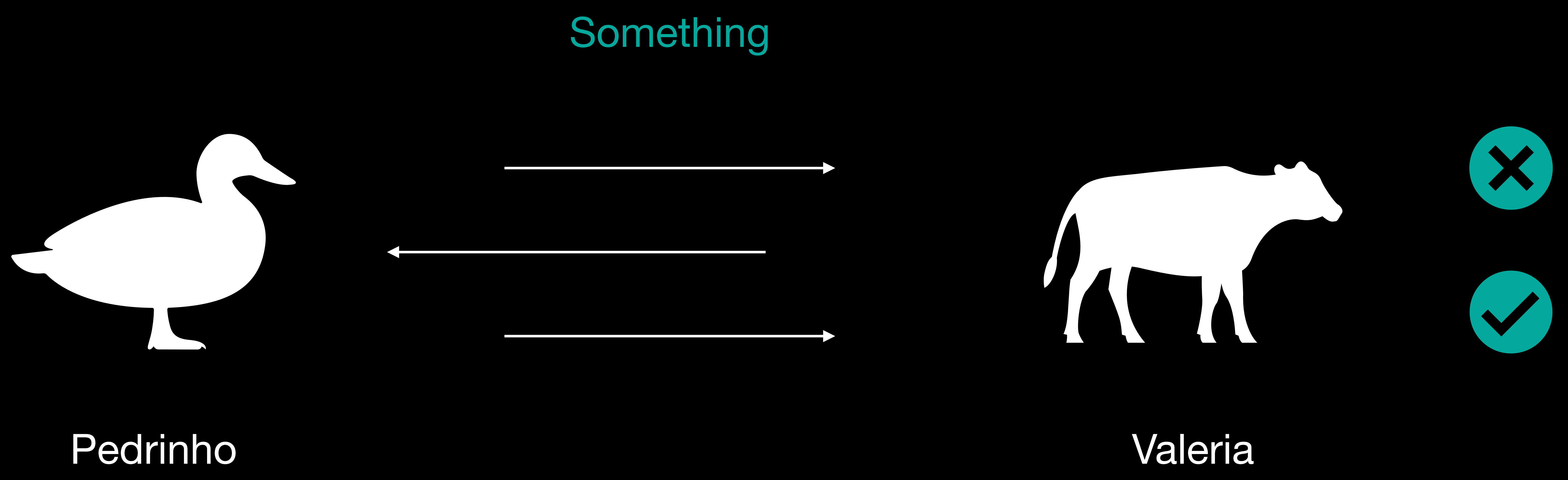
Valeria

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



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*

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

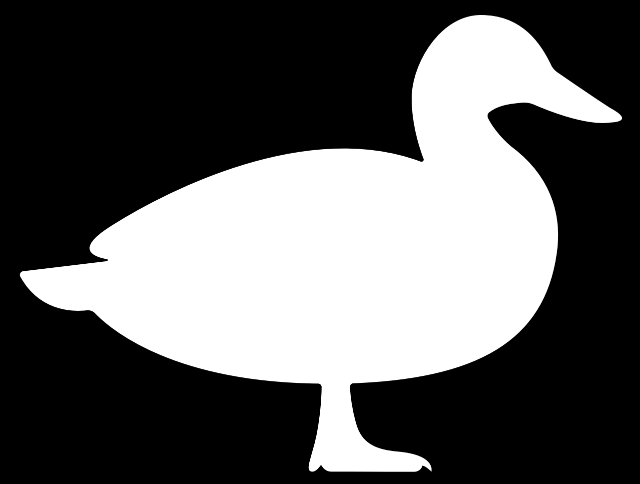
$R = \{(x, w) : \ldots\}$ is a PT decidable relation

$$x \in \mathscr{L}_R$$

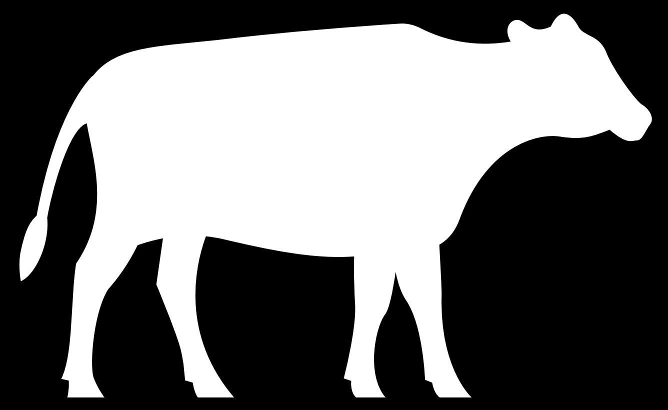
$R = \{(x, w) : \ldots\}$ is a PT decidable relation

$$x \in \mathscr{L}_R$$

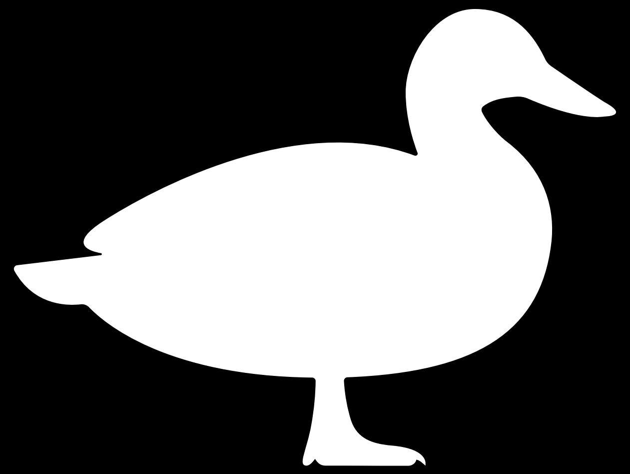
$$\mathscr{L}_R = \{x \ s.t. \ \exists w \ s.t. \ (x, w) \in R\}$$



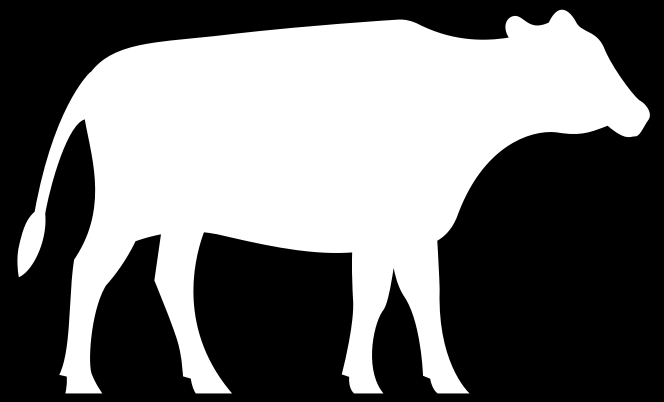
You



Security at Club

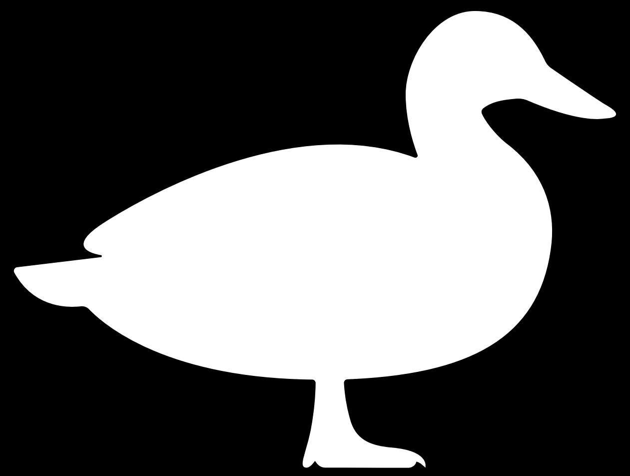


You

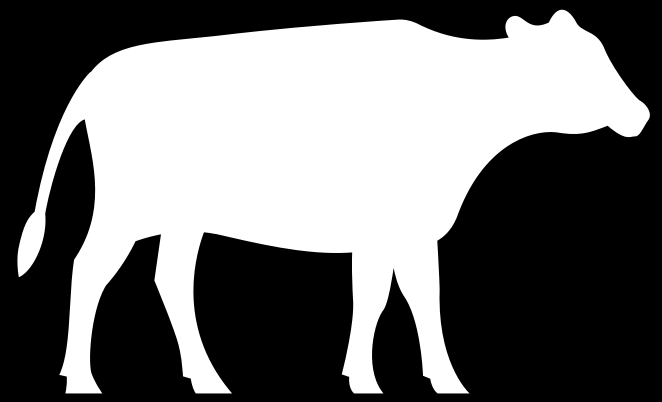


Security at Club

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



You

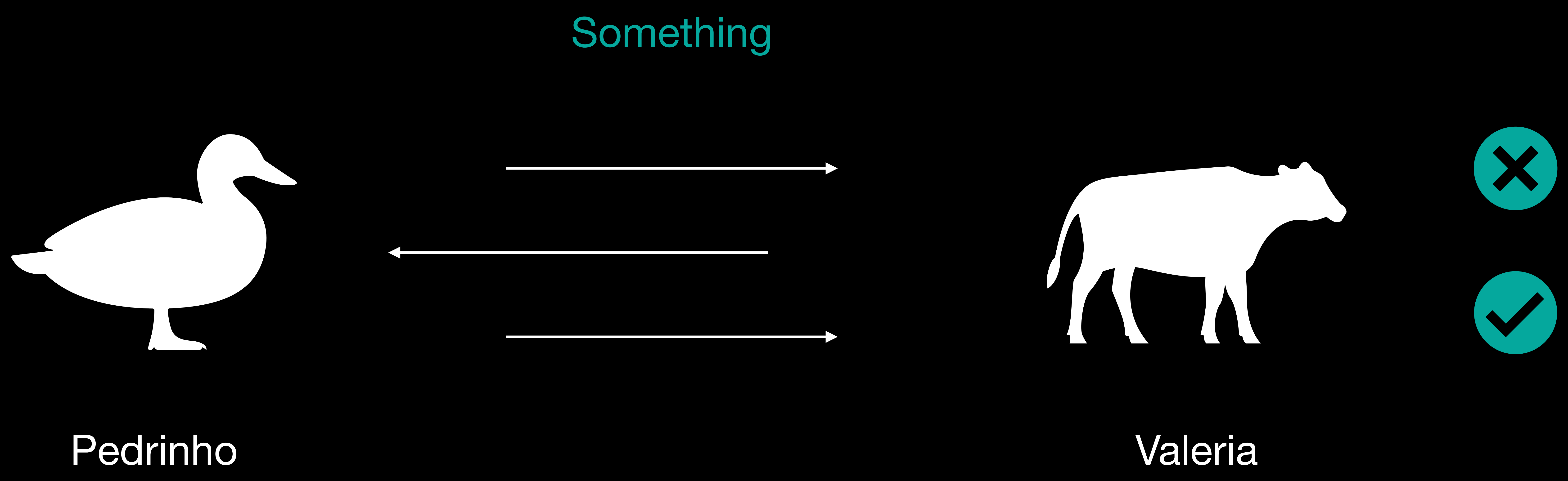


Security at Club

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

"I am in $\mathscr{L}_R$": there exists a w (my age) such that (me, w) $\in R$

Something is true

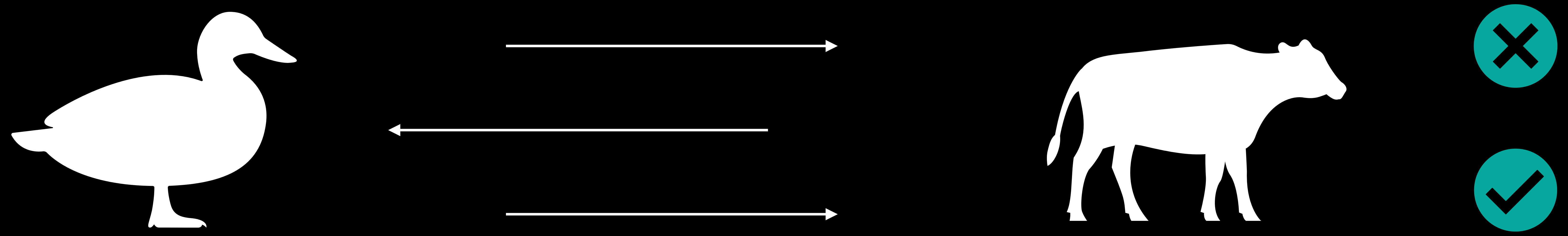


Completeness If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

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

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

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



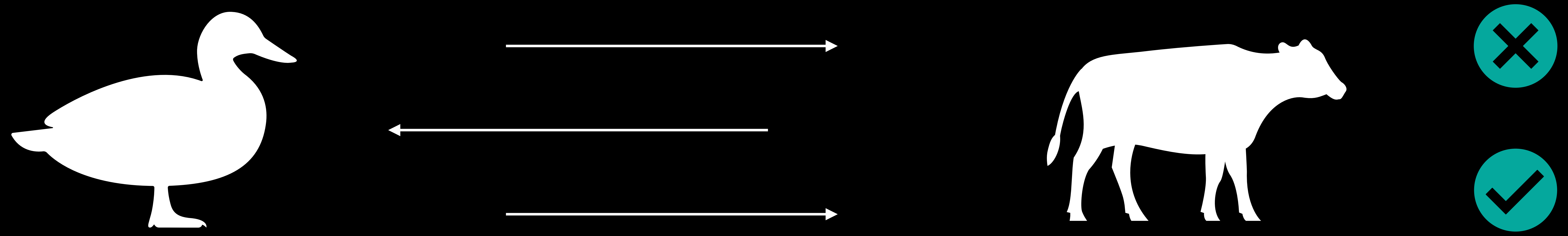
Completeness If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

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

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

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

x, R



Pedrinho

Valeria

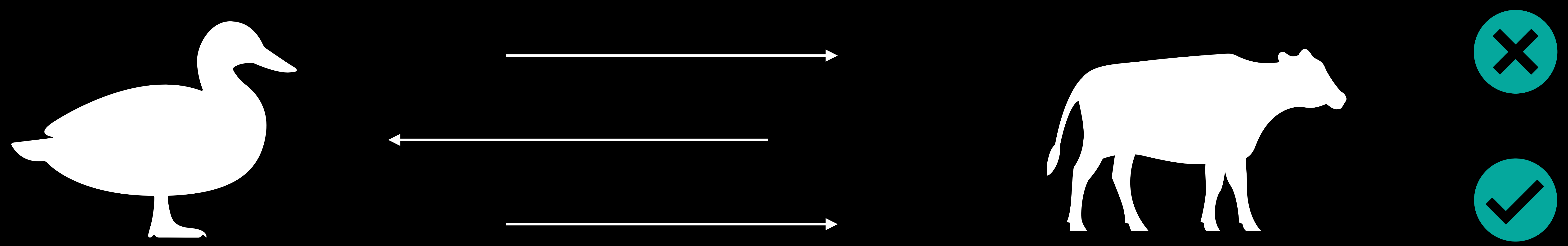
Completeness If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

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

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

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

x, R

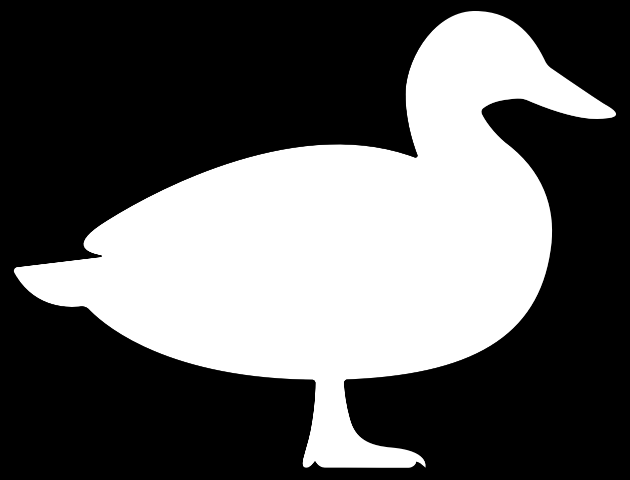


Pedrinho$((x, w), R)$ Valeria (x, R)

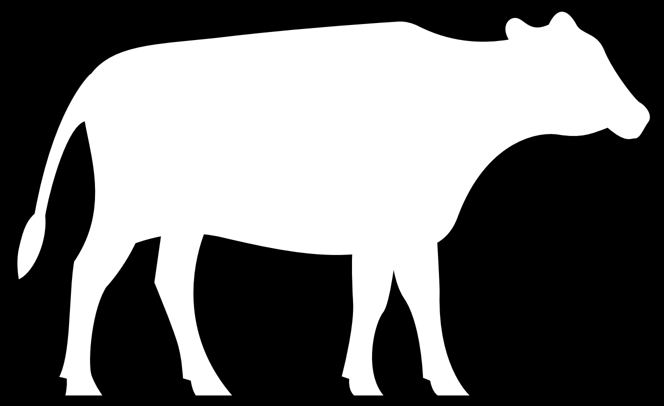
Completeness If Something is indeed true and both, Prover and Verifier, follow the procedure, Verifier accepts

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

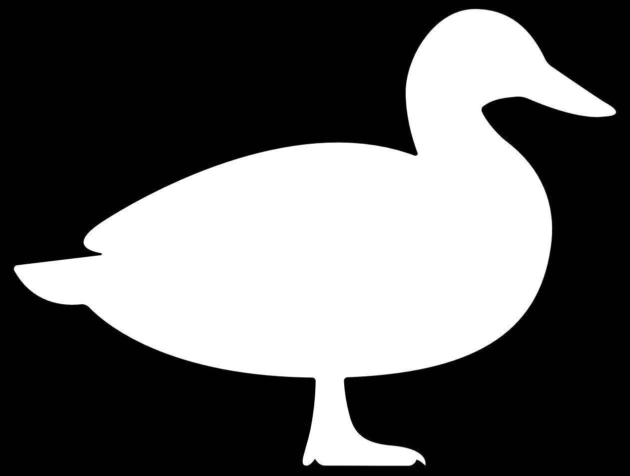
Zero-Knowledge The Verifier does not learn anything but the truth of *Something*



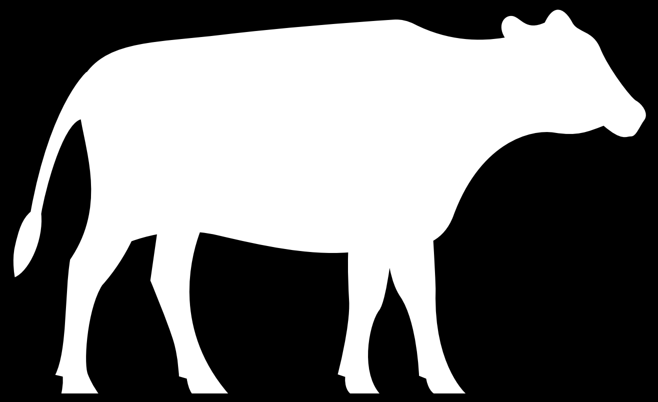
Pedrinho$((x, w), R)$



Valeria (x, R)



Pedrinho$((x, w), R)$



Valeria (x, R)

Probabilistic Polynomial Time Algorithms

$$\mathcal{P}$$

Prover$(srs, (x, w))$

$$\mathcal{V}$$

Verifier (srs, x)

Probabilistic Polynomial Time Algorithms

$$\mathcal{P}$$

Prover$(srs, (x, w))$

$$\mathcal{V}$$

0

1

Verifier (srs, x)

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

$\mathcal{P}$

Prover$(srs, (x, w))$

$\mathcal{V}$

0

1

Verifier (srs, x)

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

srs

$\mathscr{P}$ $\longrightarrow$ $\longleftarrow$ $\xrightarrow[\pi]{}$ $\mathscr{V}$

0

1

Prover$(srs, (x, w))$

Verifier (srs, x)

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

srs

$\mathcal{P}$ $\xrightarrow{\hspace{2cm}}$ $\xleftarrow{\hspace{2cm}}$ $\xrightarrow[\pi]{\hspace{2cm}}$ $\mathcal{V}$

0

1

Prover$(srs, (x, w))$

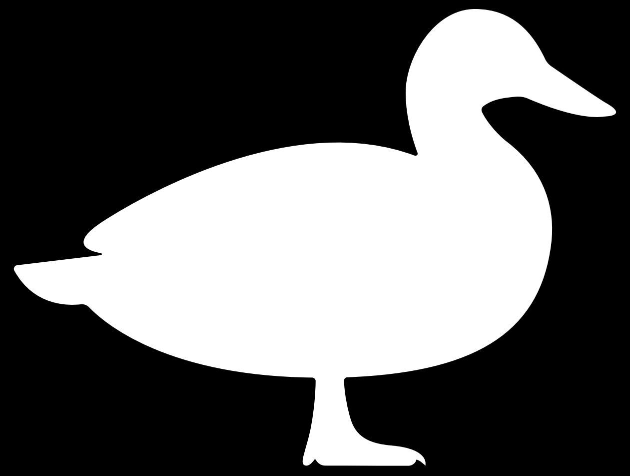
Verifier (srs, x)

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

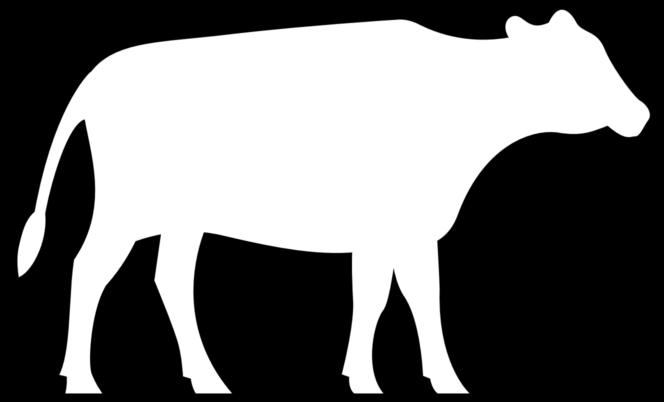
Soundness If *something* is false, then Verifier rejects with overwhelming probability

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

Examples of provers and verifiers

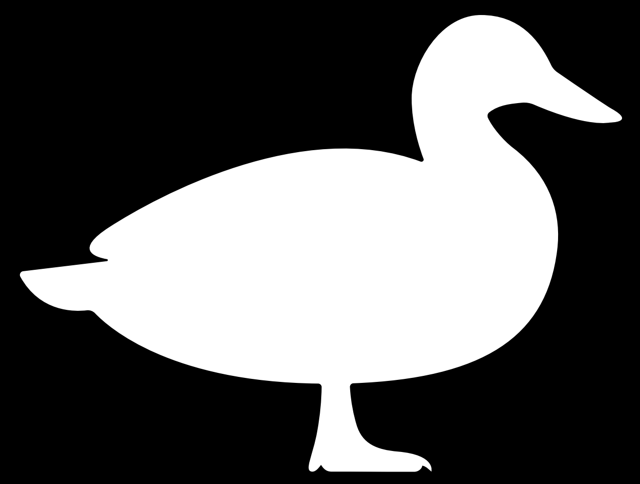


Google Cloud

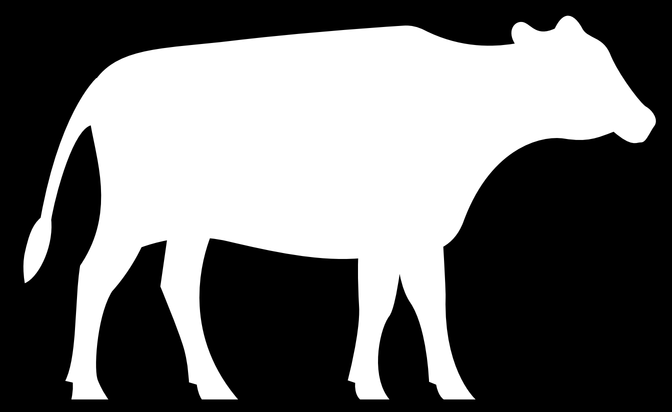


Mobil Phone

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

(Perfect) Completeness

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

(Perfect) Completeness

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

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{c}(srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w))\end{array}\right] = 1$$

(Perfect) Completeness

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

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

(Perfect) Completeness

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

$$\pi \leftarrow \mathscr{P}(srs, (x, w))$$

(Perfect) Completeness

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

$$; \begin{matrix} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{matrix} \Bigg]$$

(Perfect) Completeness

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

$$Pr \left[\mathscr{V}(srs, x, \pi) = 1; \right.$$

(Perfect) Completeness

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

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \qquad\qquad \right] = 1$$

(Perfect) Completeness

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

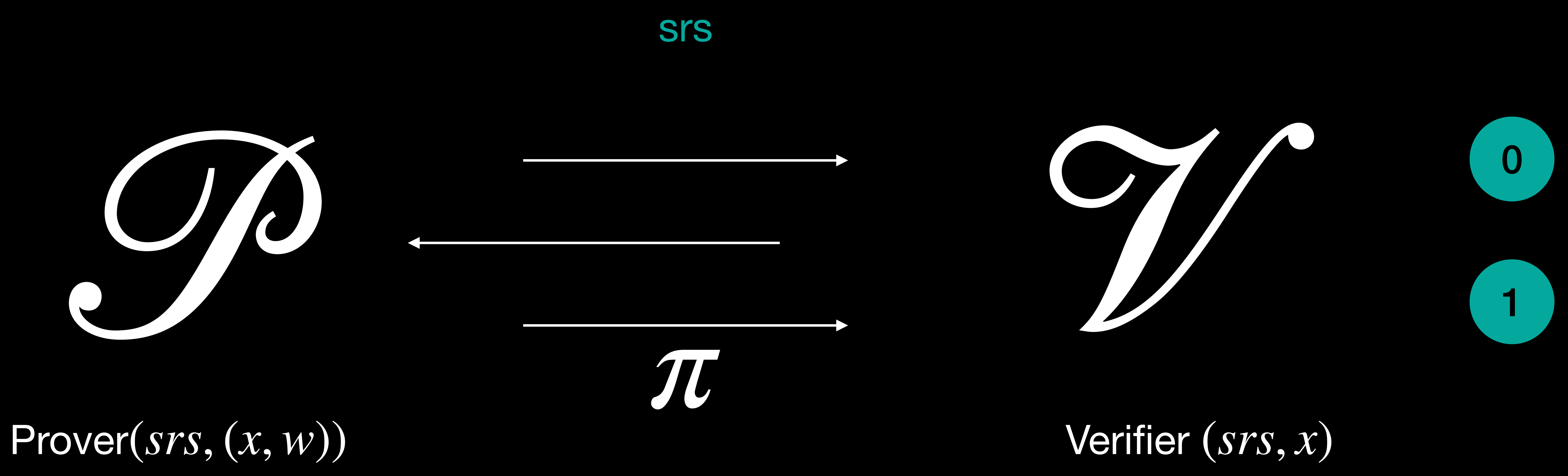
$$Pr\left[\mathscr{V}(srs, x, \pi) = 1;\ \begin{array}{r}(srs, \tau) \leftarrow \mathscr{K}(\lambda)\\ \pi \leftarrow \mathscr{P}(srs, (x, w))\end{array}\right] = 1$$

(Perfect) Completeness

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

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

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



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*

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

srs

$$\mathscr{P} \quad \begin{array}{c} \longrightarrow \\ \longleftarrow \\ \xrightarrow[\pi]{} \end{array} \quad \mathscr{V}$$

0

1

Prover$(srs, (x, w))$

Verifier (srs, x)

Completeness

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{c} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

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

(Computational) Soundness

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

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

(Computational) Soundness

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

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

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

(Computational) Soundness

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

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

$$(srs, \tau) \leftarrow \mathscr{K}$$

(Computational) Soundness

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

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

$$(x, \pi) \leftarrow \mathscr{A}(srs)$$

(Computational) Soundness

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

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

$$Pr\Big[\mathscr{V}(srs, x, \pi) = 1;$$

(Computational) Soundness

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

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

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1;\ \begin{array}{c}(srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{A}(srs)\end{array}\right] \leq negl(\lambda)$$

(Computational) Soundness

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

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

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1;\ \begin{array}{c}(srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{A}(srs)\end{array}\right] \leq negl(\lambda)$$

(Computational) Soundness

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

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

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

We are actually talking about arguments

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

srs

$$\mathscr{P} \quad \longrightarrow \quad \longleftarrow \quad \xrightarrow[\pi]{} \quad \mathscr{V}$$

0

1

Prover$(srs, (x, w))$

Verifier (srs, x)

Completeness

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

Soundness

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

Zero-Knowledge

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

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

srs

$$\mathscr{P} \quad \begin{matrix} \longrightarrow \\ \longleftarrow \\ \longrightarrow \\ \pi \end{matrix} \quad \mathscr{V}$$

0

1

Prover$(srs, (x, w))$

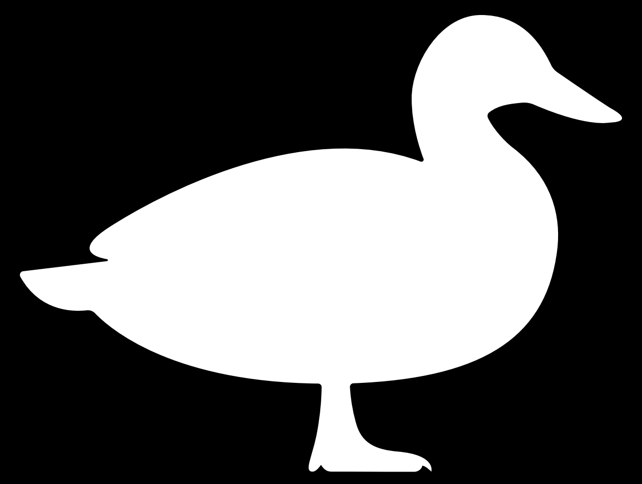
Verifier (srs, x)

Completeness
$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{matrix} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{matrix}\right] = 1$$

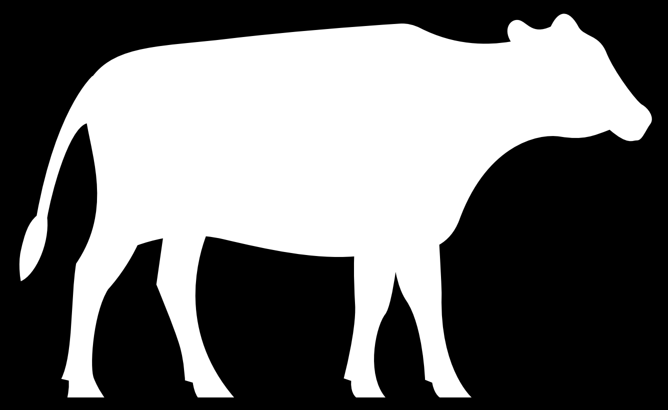
Soundness
$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{matrix} (srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{A}(srs) \end{matrix}\right] \leq negl(\lambda)$$

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

Examples of provers and verifiers

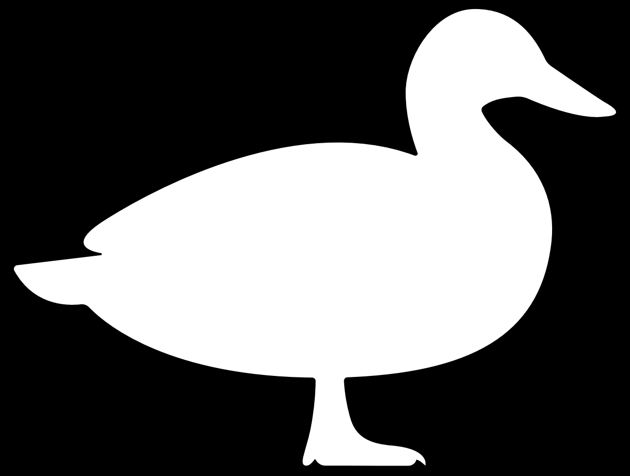


Me

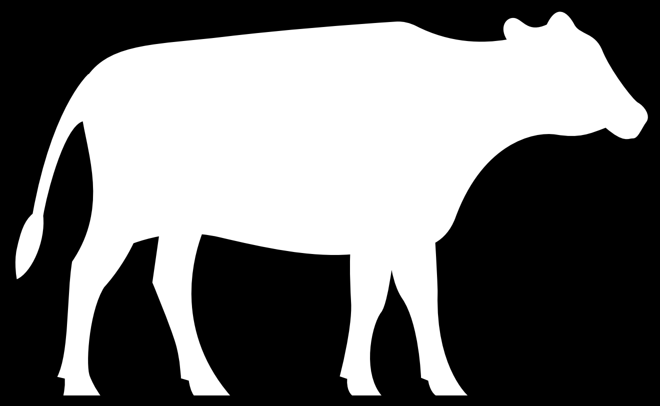


Gmail

Examples of provers and verifiers



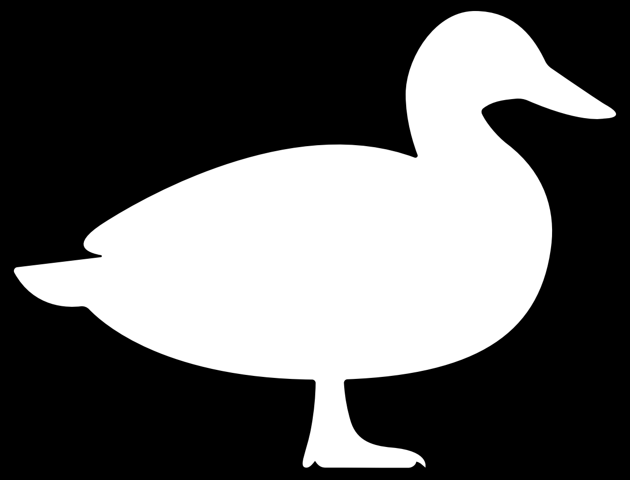
Me



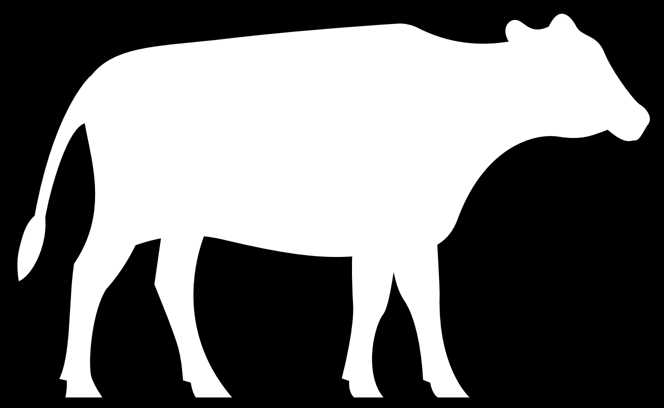
Gmail

There exists a password for this email address

Examples of provers and verifiers



Me



Gmail

There exists a password for this email address

Not enough!!!
I should *know* it

Knowledge-soundness

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

Knowledge-soundness

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

$$Pr\left[\begin{array}{c}(x,w) \notin R \wedge \\ \mathscr{V}(srs,x,\pi) = 1\end{array}; \begin{array}{c}(srs,\tau) \leftarrow \mathscr{K} \\ (x,\pi) \leftarrow \mathscr{P}^*(srs) \\ w \leftarrow \mathscr{E}(srs,x,\pi)\end{array}\right] \leq negl(\lambda)$$

Knowledge-soundness

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

$$Pr\left[\begin{matrix}\boxed{(x,w) \notin R \wedge} \\ \mathscr{V}(srs,x,\pi)=1\end{matrix} ; \begin{matrix}(srs,\tau) \leftarrow \mathscr{K} \\ (x,\pi) \leftarrow \mathscr{P}*(srs) \\ \boxed{w \leftarrow \mathscr{E}(srs,x,\pi)}\end{matrix}\right] \leq negl(\lambda)$$

Knowledge-soundness

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

$$Pr\left[\begin{array}{c} \boxed{(x,w) \notin R \wedge} \\ \mathscr{V}(srs,x,\pi) = 1 \end{array} ; \begin{array}{c} (srs,\tau) \leftarrow \mathscr{K} \\ (x,\pi) \leftarrow \mathscr{P}^*(srs) \\ \boxed{w \leftarrow \mathscr{E}(srs,x,\pi)} \end{array}\right] \leq negl(\lambda)$$

An argument that satisfies knowledge-soundness is an **argument of knowledge**

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

srs

$$\mathscr{P} \quad \begin{array}{c} \longrightarrow \\ \longleftarrow \\ \xrightarrow[\pi]{} \end{array} \quad \mathscr{V}$$

0

1

Prover$(srs, (x, w))$

Verifier (srs, x)

Completeness

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

Knowledge-Soundness

$$Pr\left[\begin{array}{c} (x, w) \notin R \wedge \\ \mathscr{V}(srs, x, \pi) = 1 \end{array}; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{P}^*(srs) \\ w \leftarrow \mathscr{E}(srs, x, \pi) \end{array}\right] \leq negl(\lambda)$$

Zero-Knowledge The Verifier does not learn anything but the truth of *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 $\mathcal{S}$, with access to the private information, such that for all $\mathcal{V}^*$

Zero-knowledge

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

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

$$Pr\left[\mathcal{V}*(srs, \pi_{sim}) = 1; \begin{array}{r} (srs, \tau) \leftarrow \mathcal{K} \\ x \leftarrow \mathcal{V}*(srs) \\ \pi_{sim} \leftarrow \mathcal{S}(srs, \tau, x)) \end{array}\right]$$

Zero-knowledge

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

$$Pr\left[\mathscr{V}*(srs,\pi)=1;\quad \begin{array}{r}(srs,\tau)\leftarrow\mathscr{K}\\ x\leftarrow\mathscr{V}*(srs)\\ \pi\leftarrow\mathscr{P}(srs,(x,w))\end{array}\right]\approx$$

Zero-knowledge

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

$$Pr\left[\boxed{\mathscr{V}*(srs,\pi) =} 1;\ \begin{array}{c} (srs,\tau) \leftarrow \mathscr{K} \\ \boxed{x \leftarrow \mathscr{V}*(srs)} \\ \pi \leftarrow \mathscr{P}(srs,(x,w)) \end{array}\right] \approx$$

Zero-knowledge

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

$$Pr\left[\mathcal{V}*(srs,\pi)=1;\quad \begin{array}{r}(srs,\tau)\leftarrow\mathcal{K}\\ x\leftarrow\mathcal{V}*(srs)\\ \pi\leftarrow\mathcal{P}(srs,(x,w))\end{array}\right]\approx$$

$$Pr\left[\mathcal{V}*(srs,\pi_{sim})=1;\quad \begin{array}{r}(srs,\tau)\leftarrow\mathcal{K}\\ x\leftarrow\mathcal{V}*(srs)\\ \pi_{sim}\leftarrow\mathcal{S}(srs,\tau,x))\end{array}\right]$$

Zero-knowledge

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

$$Pr\left[\mathcal{V}*(srs, \pi_{sim}) = 1; \quad \begin{array}{r} (srs, \tau) \leftarrow \mathcal{K} \\ x \leftarrow \mathcal{V}*(srs) \\ \pi_{sim} \leftarrow \mathcal{S}(srs, \tau, x)) \end{array}\right]$$

Zero-knowledge

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

$$Pr\left[\mathscr{V}*(srs, \pi_{sim}) = 1; \quad \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K} \\ x \leftarrow \mathscr{V}*(srs) \\ \boxed{\pi_{sim} \leftarrow \mathscr{S}(srs, \tau, x))} \end{array}\right]$$

Zero-knowledge

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

$$Pr\left[\mathcal{V}*(srs,\pi)=1;\quad \begin{array}{r}(srs,\tau)\leftarrow\mathcal{K}\\ x\leftarrow\mathcal{V}*(srs)\\ \pi\leftarrow\mathcal{P}(srs,(x,w))\end{array}\right]\approx$$

$$Pr\left[\mathcal{V}*(srs,\pi_{sim})=1;\quad \begin{array}{r}(srs,\tau)\leftarrow\mathcal{K}\\ x\leftarrow\mathcal{V}*(srs)\\ \pi_{sim}\leftarrow\mathcal{S}(srs,\tau,x)\end{array}\right]$$

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

srs

$$\mathscr{P} \qquad \xrightarrow{\hspace{2em}} \qquad \mathscr{V}$$

$$\xleftarrow{\hspace{2em}}$$

$$\xrightarrow[\pi]{\hspace{2em}}$$

0

1

Prover$(srs, (x, w))$ — Verifier (srs, x)

Completeness $$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{c} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

Knowledge-Soundness $$Pr\left[\begin{array}{c} (x, w) \notin R \wedge \\ \mathscr{V}(srs, x, \pi) = 1 \end{array}; \begin{array}{c} (srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{A}(srs) \\ w \leftarrow \mathscr{E}(srs, x, \pi) \end{array}\right] \leq negl(\lambda)$$

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

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

srs

$$\mathscr{P} \quad \rightleftarrows \quad \mathscr{V}$$

$$\pi$$

0

1

Prover$(srs, (x, w))$

Verifier (srs, x)

Completeness

$$Pr\left[\mathscr{V}(srs, x, \pi) = 1; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K}(\lambda) \\ \pi \leftarrow \mathscr{P}(srs, (x, w)) \end{array}\right] = 1$$

Knowledge-Soundness

$$Pr\left[\begin{array}{c} (x, w) \notin R \wedge \\ \mathscr{V}(srs, x, \pi) = 1 \end{array}; \begin{array}{r} (srs, \tau) \leftarrow \mathscr{K} \\ (x, \pi) \leftarrow \mathscr{A}(srs) \\ w \leftarrow \mathscr{E}(srs, x, \pi) \end{array}\right] \leq negl(\lambda)$$

Zero-Knowledge

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

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

$\boxed{\text{Discrete logs are hard to compute (In some groups)}}$

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

Knowledge of discrete log

Discrete logs are hard to compute
(In some groups)

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

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ \underline{h = g^x}\}$$

Knowledge of discrete log

Discrete logs are hard to compute
(In some groups)

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

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ \underline{h = g^x}\}$$

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

$\mathcal{P}$

$\mathcal{V}$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$ $\mathscr{V}$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$ $\mathscr{V}$

$((h, \mathbb{G}), (x, h))$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathcal{P}$

$((h, \mathbb{G}), (x, h))$

$\mathcal{V}$

$((g, \mathbb{G}), h)$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$

$((h, \mathbb{G}), (x, h))$

$r \leftarrow \mathbb{Z}_q$

$u = g^r$

$\mathscr{V}$

$((g, \mathbb{G}), h)$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$		$\mathscr{V}$
$((h, \mathbb{G}), (x, h))$		$((g, \mathbb{G}), h)$
$r \leftarrow \mathbb{Z}_q$ $u = g^r$	$\xrightarrow{\quad u \quad}$	

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$		$\mathscr{V}$
$((h, \mathbb{G}), (x, h))$		$((g, \mathbb{G}), h)$
$r \leftarrow \mathbb{Z}_q$ $u = g^r$	$\xrightarrow{\quad u \quad}$	
		$c \leftarrow \mathbb{Z}_q$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$		$\mathscr{V}$
$((h, \mathbb{G}), (x, h))$		$((g, \mathbb{G}), h)$
$r \leftarrow \mathbb{Z}_q$ $u = g^r$	$\xrightarrow{\quad u \quad}$	
		$c \leftarrow \mathbb{Z}_q$
	$\xleftarrow{\quad c \quad}$	

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$$\mathcal{P} \qquad\qquad\qquad\qquad \mathcal{V}$$

$$((h, \mathbb{G}), (x, h)) \qquad\qquad ((g, \mathbb{G}), h)$$

$$\begin{array}{ccc} r \leftarrow \mathbb{Z}_q & & \\ u = g^r & \xrightarrow{\quad u \quad} & \\ & & c \leftarrow \mathbb{Z}_q \\ & \xleftarrow{\quad c \quad} & \\ z = r + cx & & \end{array}$$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$$\begin{array}{ccc}
\mathcal{P} & & \mathcal{V} \\
((h, \mathbb{G}), (x, h)) & & ((g, \mathbb{G}), h) \\
r \leftarrow \mathbb{Z}_q & & \\
u = g^r & \xrightarrow{\quad u \quad} & \\
 & & c \leftarrow \mathbb{Z}_q \\
 & \xleftarrow{\quad c \quad} & \\
z = r + cx & & \\
 & \xrightarrow{\quad z \quad} &
\end{array}$$

Knowledge of discrete log - Schnorr

$$R = \{(x, h) : x \in \mathbb{Z}_q \ \wedge \ h \in \mathbb{G} \ \wedge \ h = g^x\}$$

$\mathscr{P}$ $((h, \mathbb{G}), (x, h))$

$\mathscr{V}$ $((g, \mathbb{G}), h)$

$r \leftarrow \mathbb{Z}_q$

$u = g^r$

$\xrightarrow{\quad u \quad}$

$c \leftarrow \mathbb{Z}_q$

$\xleftarrow{\quad c \quad}$

$z = r + cx$

$\xrightarrow{\quad z \quad}$

$g^z = uh^c$

Completeness

Completeness

Theorem: The scheme satisfies completeness

Completeness

Theorem: The scheme satisfies completeness

$$g^z = uh^c$$

Completeness

Theorem: The scheme satisfies completeness

$$\underline{g^z} = uh^c$$

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

$$g^z = uh^c$$

$$g^{r+cx} = \underline{uh^c}$$

$$g^{r+cx} = \underline{g^r h^c}$$

Completeness

Theorem: The scheme satisfies completeness

$$g^z = uh^c$$

$$g^{r+cx} = uh^c$$

$$g^{r+cx} = g^r \underline{h^c}$$

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

Knowledge-soundness

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

- $\mathscr{E}$ runs prover $\mathscr{P}^*$ to obtain initial message u

Knowledge-soundness

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

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

Knowledge-soundness

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

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

Knowledge-soundness

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

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

Knowledge-soundness

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

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

Knowledge-soundness

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

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

With probability ϵ^2, $g^{z_1} = uh^{c_1} \;\wedge\; g^{z_2} = uh^{c_2}$. Then,

Knowledge-soundness

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

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

With probability ϵ^2, $\quad g^{z_1} = uh^{c_1} \ \wedge \ g^{z_2} = uh^{c_2}$. Then,

$$\boxed{\frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \rightarrow \frac{g^{z_1}}{g^{z_2}} = \frac{h^{c_1}}{h^{c_2}} \rightarrow g^{z_1 - z_2} = h^{c_1 - c_2} \rightarrow g^{z_1 - z_2} = (g^x)^{(c_1 - c_2)} \rightarrow g^{\frac{z_1 - z_2}{c_1 - c_2}} = (g^x)}$$

Honest-Verifier Zero-knowledge

Honest-Verifier Zero-knowledge

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

- $z \leftarrow \mathbb{Z}_q$
- $c \leftarrow \mathbb{Z}_q$

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)

- $z \leftarrow \mathbb{Z}_q$
- $c \leftarrow \mathbb{Z}_q$
- $u = \dfrac{g^z}{h^c}$

$$g^z = uh^c$$

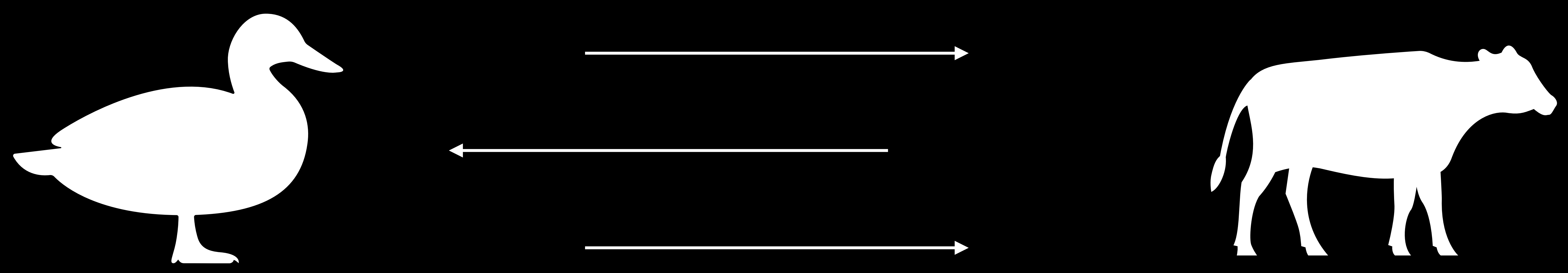
Honest-Verifier Zero-knowledge

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

- $z \leftarrow \mathbb{Z}_q$
- $c \leftarrow \mathbb{Z}_q$
- $u = \dfrac{g^z}{h^c}$
- Output (u, c, z)

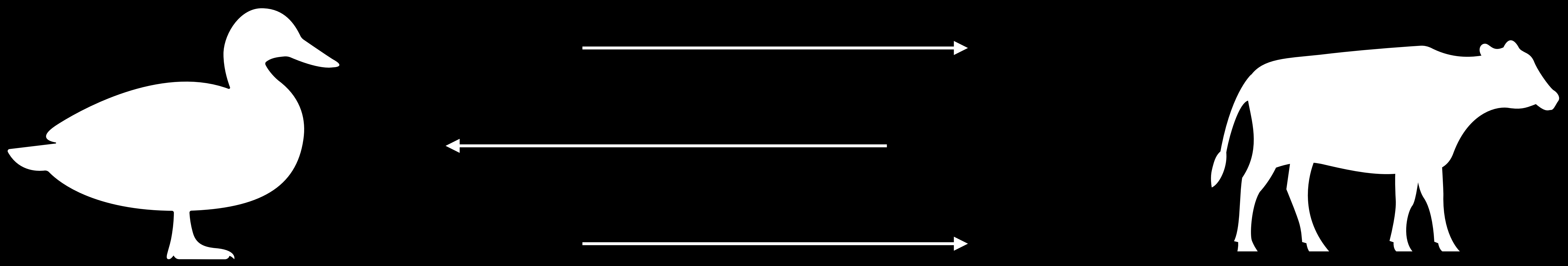
$$g^z = uh^c$$

Lookup Tables



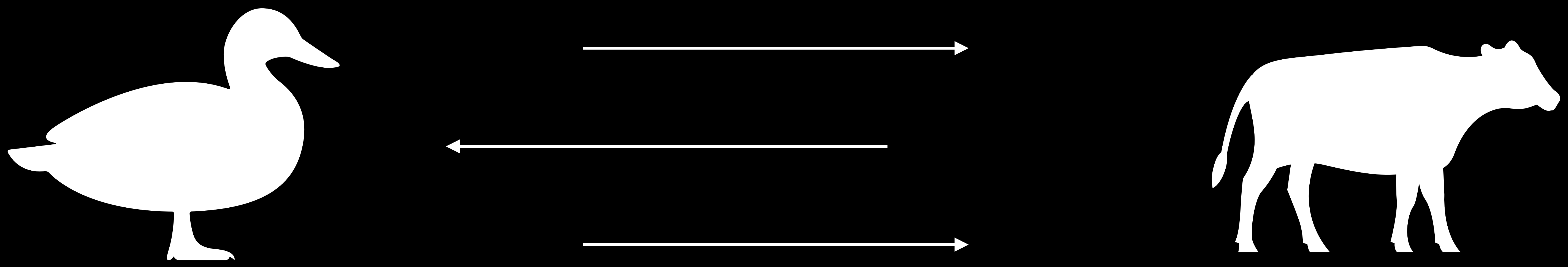
Pedrinho Valeria

$$\overrightarrow{T} = (v_1, v_2, v_3, \ldots, v_m)$$



$$\overrightarrow{T} = (v_1, v_2, v_3, \ldots, v_m)$$

C *is a commitment to elements* $s_i \in \overrightarrow{T}$



Pedrinho Valeria

Importance

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

Some examples

Some examples

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

Some examples

C *is your age*

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

Some examples

C *is your age*

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

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

Some examples

C *is your age*

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

C *is* (x_i, y_i)

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

Some examples

C *is your age*

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

C *is* (x_i, y_i)

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

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

Some examples

C *is your age* $\quad \vec{T} = (18, 19, \ldots, 120)$

C *is* (x_i, y_i)

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

C *is my user name* $\quad \vec{T} = (user_1, \ldots, user_m)$

Some examples

C *is your age*

C *is* (x_i, y_i)

C *is my user name*

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

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

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

Membership proofs from Lookup tables

C *is my user name*

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

Membership proofs from Lookup tables

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

C *is my user name*

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

Membership proofs from Lookup tables

C *is my user name*

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

Membership proofs from Lookup tables

C *is my user name*

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

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

Membership proofs from Lookup tables

C *is my user name*

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

$sk \leftarrow \mathbb{Z}_q$

$pk = g^{sk}$

Membership proofs from Lookup tables

C *is my user name*

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

$$pk = g^{sk}$$

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

Membership proofs from Lookup tables

C *is my user name*

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

$$pk = g^{sk}$$

$$\vec{T} = (pk_1, \ldots, pk_m)$$

$$\mathrm{C} = Com(pk) = g^{x+r.sk}$$

Membership proofs from Lookup tables

C *is my user name*

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

$$pk = g^{sk}$$

$$\mathrm{C} = Com(pk) = g^{x+r.sk}$$

$$\vec{T} = (pk_1, \ldots, pk_m)$$

"I am authorized":

Membership proofs from Lookup tables

C *is my user name*

$sk \leftarrow \mathbb{Z}_q$

$pk = g^{sk}$

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

C $= Com(pk) = g^{x+r.sk}$

"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*

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

$$pk = g^{sk}$$

$$\vec{T} = (pk_1, \ldots, pk_m)$$

$$\mathsf{C} = Com(pk) = g^{x+r.sk}$$

"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

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

$$pk = g^{sk}$$

$$\vec{T} = (pk_1, \ldots, pk_m)$$

$$\mathsf{C} = Com(pk) = g^{x+r.sk}$$

"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