# On the Security and Privacy of delegated computation

Anca Nitulescu
DI ENS - Cascade

# Outline

## Introduction

Cryptography

Primitives

## Motivation

Cloud Computation

Security requirements

## SNARKs

**Arguments of Knowledge**

**SNARK**
Definition, Construction

## Directions

**Quantum SNARKs**

Difficulties
Applications
Open Problems

**Conclusions**

# Cryptography

Much of the cryptography used today offers security properties for **data** and **communication**.

Aspects in information security:

- **data confidentiality**
- **authentication**
- **data integrity**

what about **computations?**

# Cryptographic Primitives

- Primitives = algorithms with basic cryptographic properties
- Theoretical work in cryptography
- Tools used to build more complicated cryptographic protocols
- Provide one functionality at the time:

**privacy**

Encryption schemes

compute a ciphertext to hide a message

**authentication**

Digital signatures

confirm the author of a message

**integrity**

Hash functions

compute a reduced hash for a message (e.g. SHA-256)

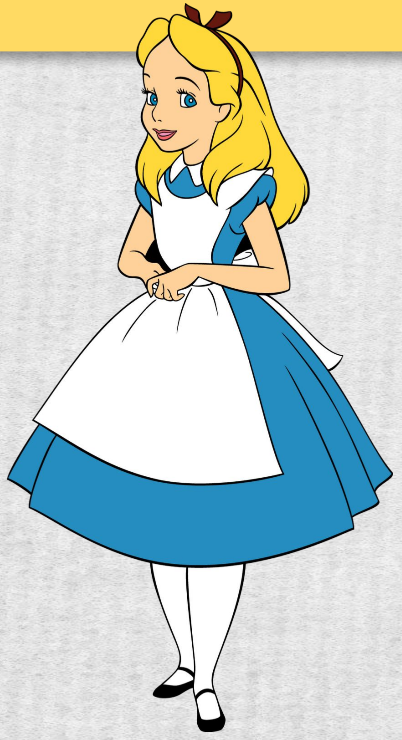
# Privacy

**Encryption schemes**

$m \rightarrow C = Enc(m)$

$C \rightarrow M = Dec(C)$

m




# Privacy

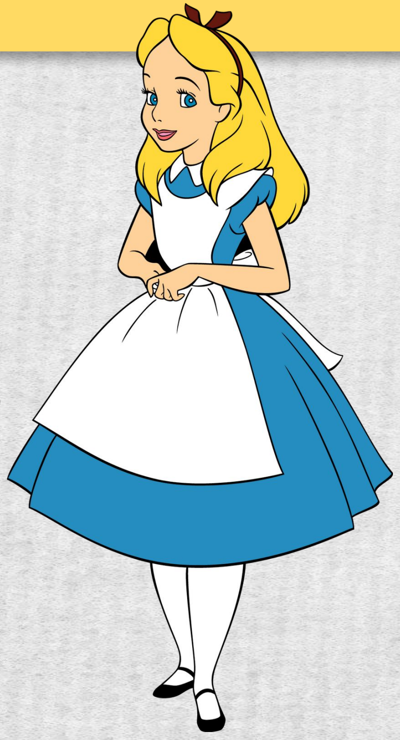
**Encryption schemes**

$m \rightarrow C = Enc(m)$

$C \rightarrow M = Dec(C)$

**m**

C=Enc(m)




**C**

m=Dec(C)

# Privacy

**Encryption schemes**

$m \longrightarrow C = Enc(m)$

$C \longrightarrow M = Dec(C)$

$m'$

$C' = Enc(m')$


$C'$

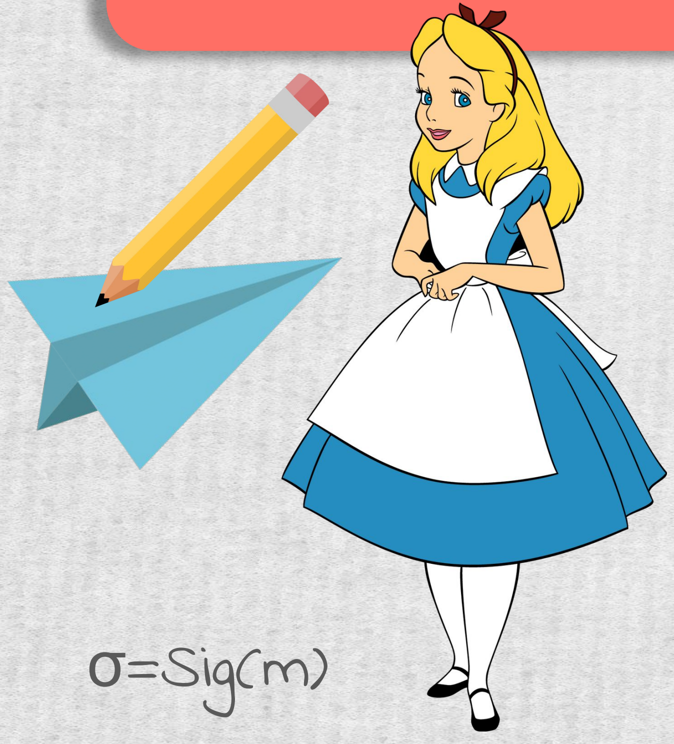
$m' = Dec(C')$

# Authenticity

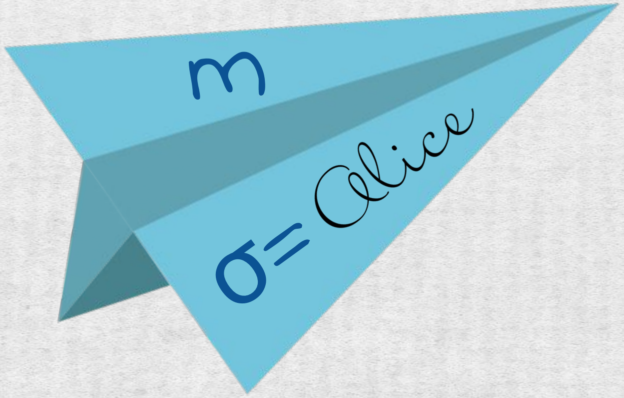
**Signature schemes**

$m \rightarrow \sigma = Sig(m)$

$Ver(\sigma) \rightarrow$ accept/reject



$\sigma = Sig(m)$

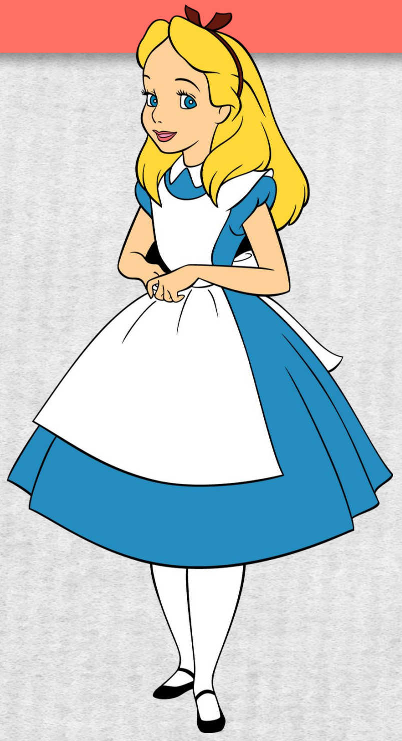



# Authenticity

**Signature schemes**

$m \rightarrow \sigma = \mathrm{Sig}(m)$

$\mathrm{Ver}(\sigma) \rightarrow$ accept/reject



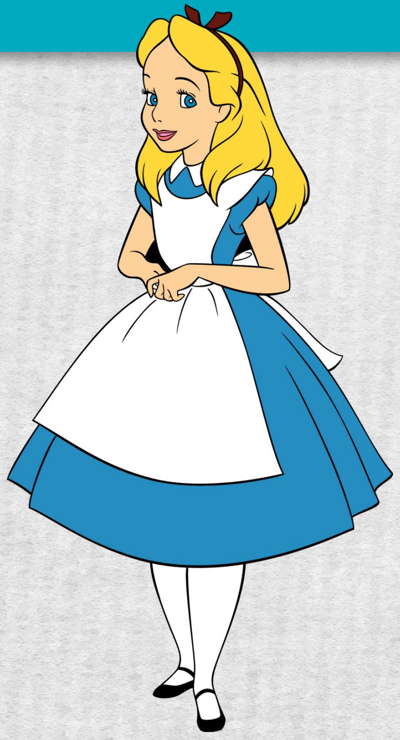
$\mathrm{Ver}(\sigma)$

# Data Integrity

**Attack on Integrity**

Adversary:

intercepts the message




# Data Integrity

**Attack on Integrity**

Adversary:

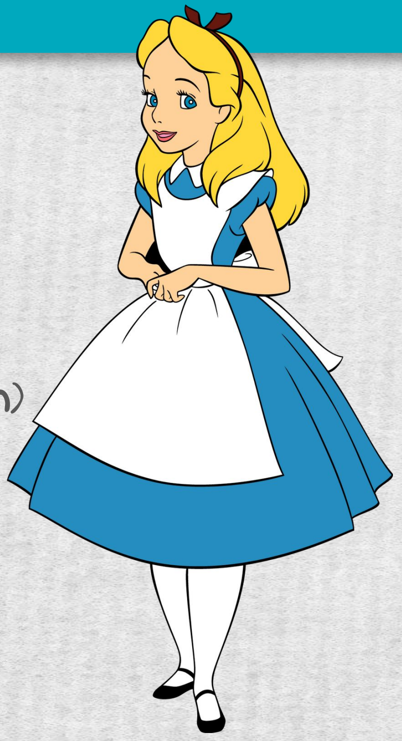
changes the message

# Data Integrity

One-way Hash Functions

m → H = Hash(m)

H → ?m'     H = Hash(m')



m

H = Hash(m)



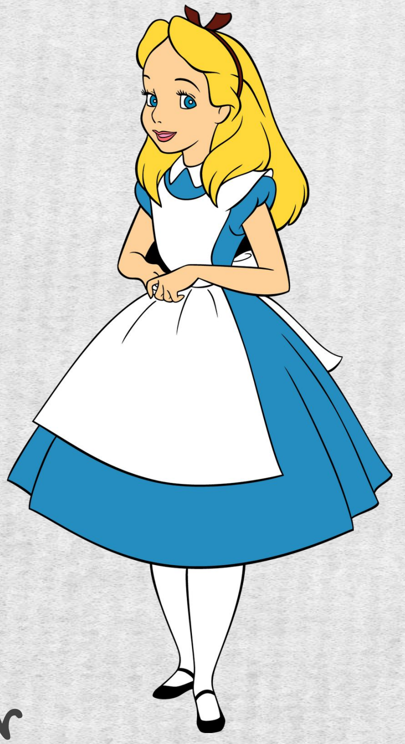
# Delegate Computation to Cloud

User

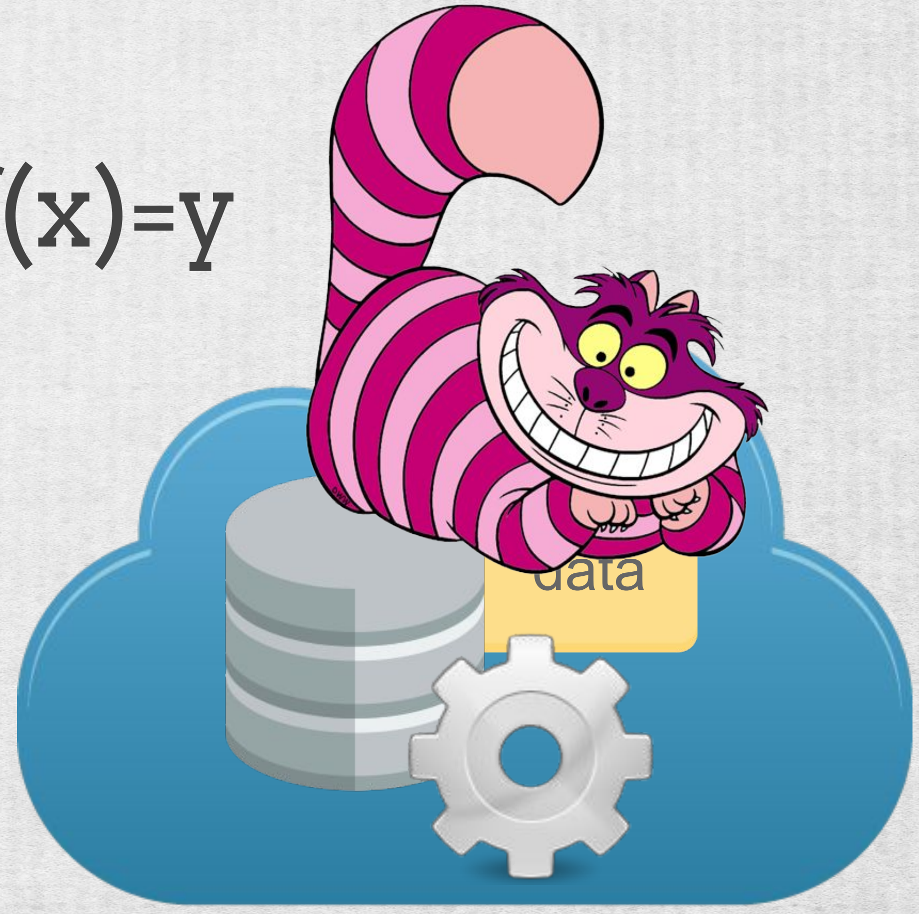

Server

# Delegate Computation to Cloud

$f(x)=y$



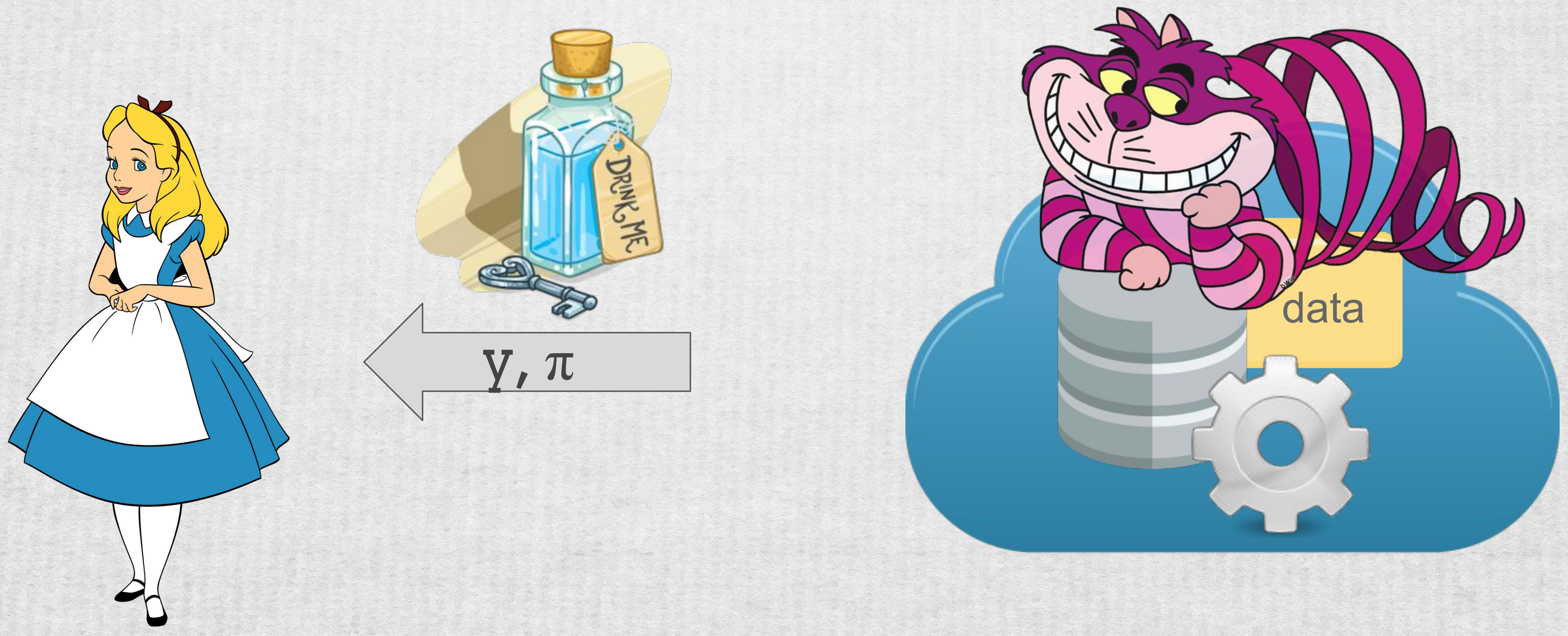
$x$



User

Server

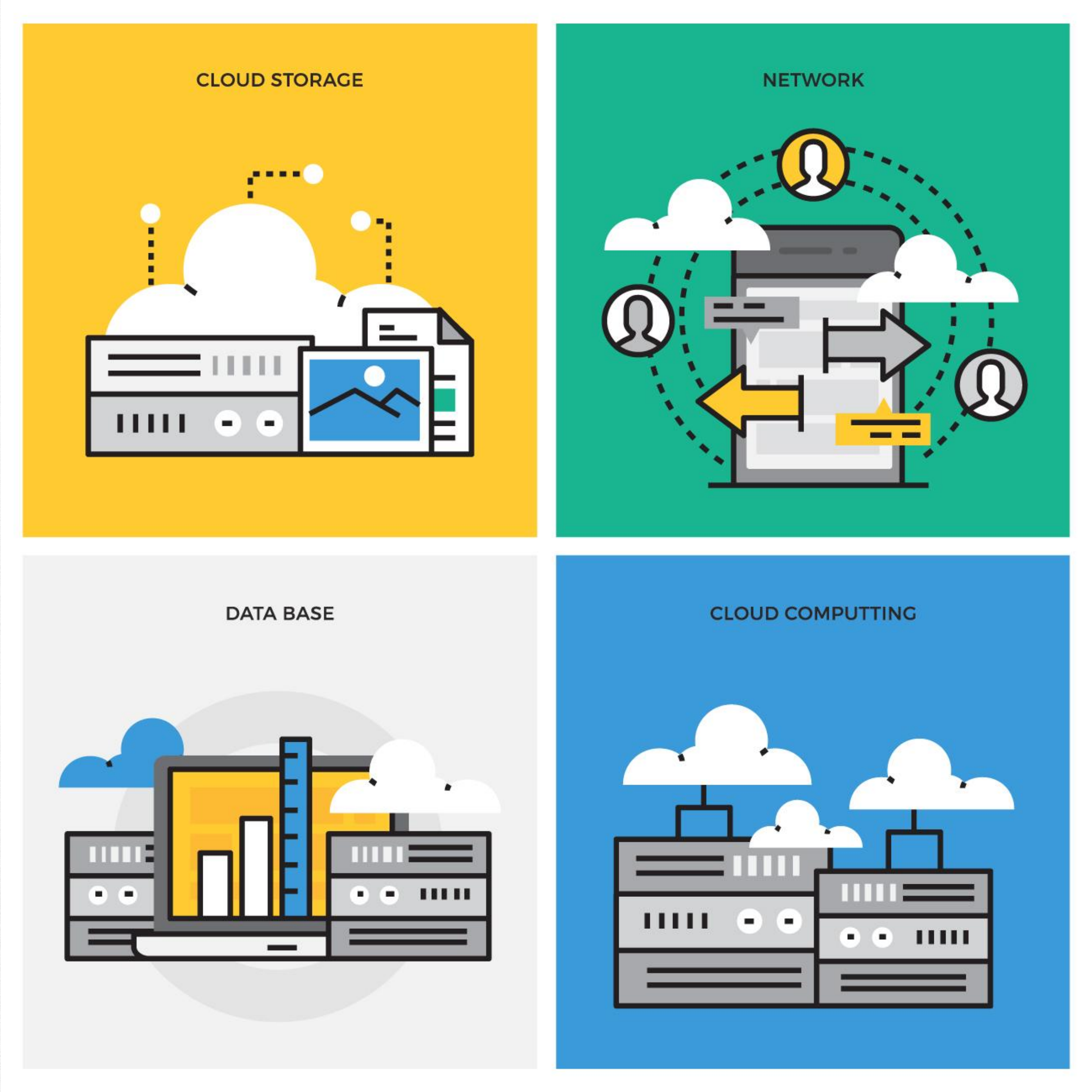
# Integrity of Delegated Computation?




trust the server / ask for a proof

# CLOUD - Available for Everything

**Store** documents, photos, videos, etc




**Share** them with colleagues, friends, family

**Ask queries** on the data

**Process** the data

# Outsourced Processing

The Cloud Provider:

- **knows the content**
- **performs the computations**

**Claims to**

- identify users
- apply access rights
- safely store the data
- securely process the data
- answer correct our queries
- protect privacy

# Risks

For economical reasons, by accident, or attacks

- data can get deleted
- results of computation can be modified
- one can use your private data to analyze and sell/negotiate the information

# Delegated Computation - Requirements

**Confidentiality**

Medical Record

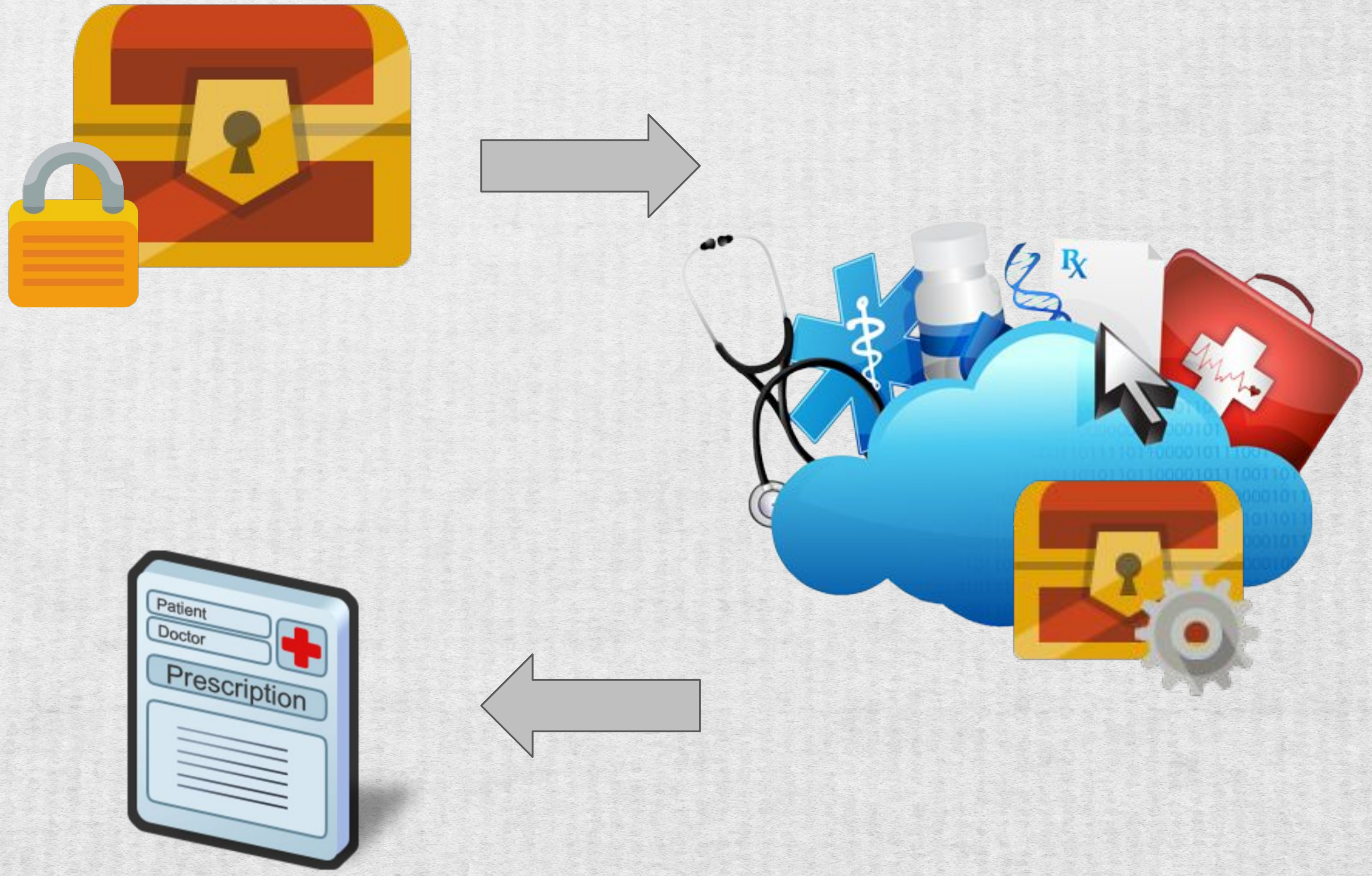

**Integrity**

Verify Computation Result

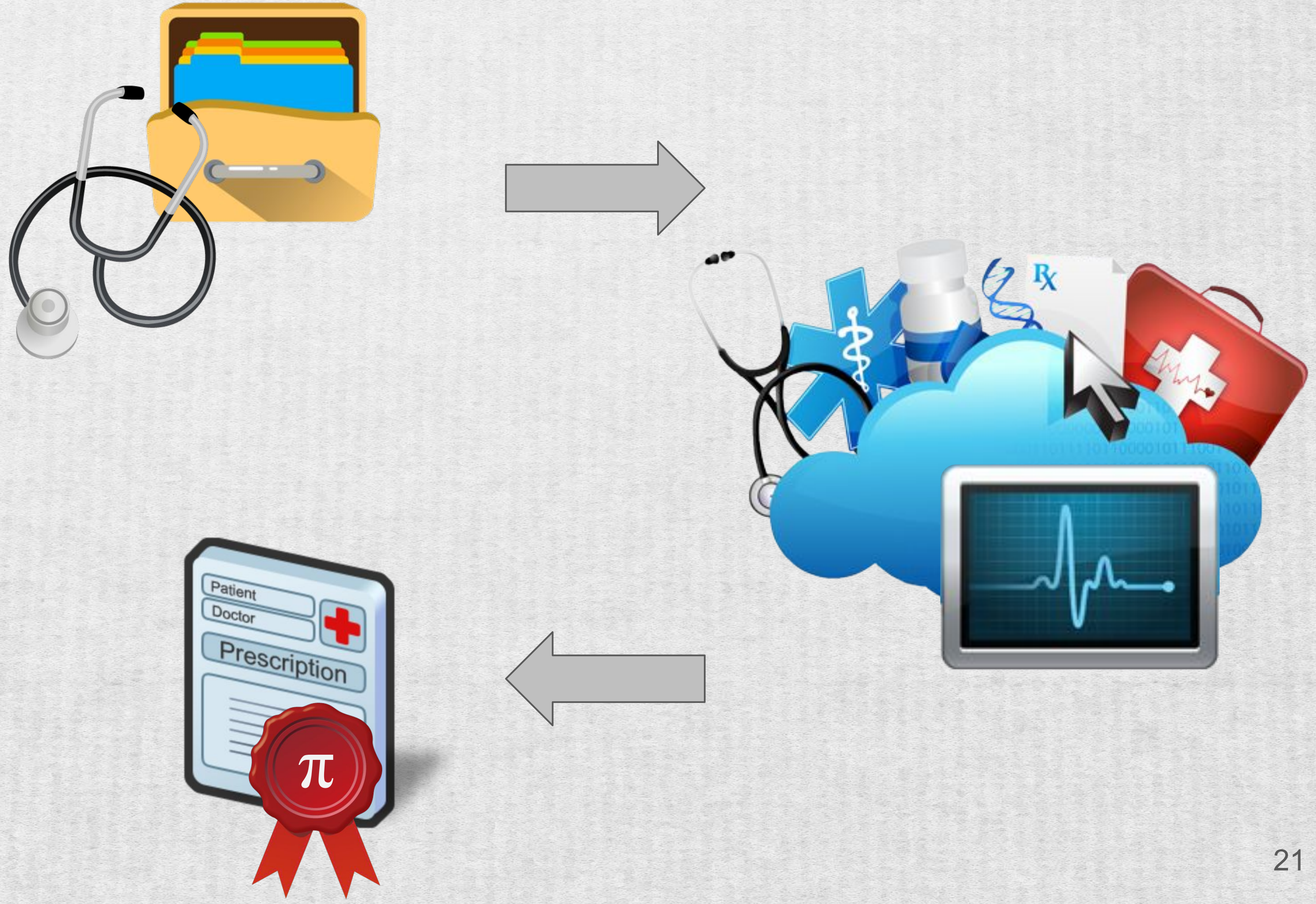
# Delegated Computation - Requirements

## Confidentiality

Fully Homomorphic Encryption



# Delegated Computation - Requirements




**Integrity**

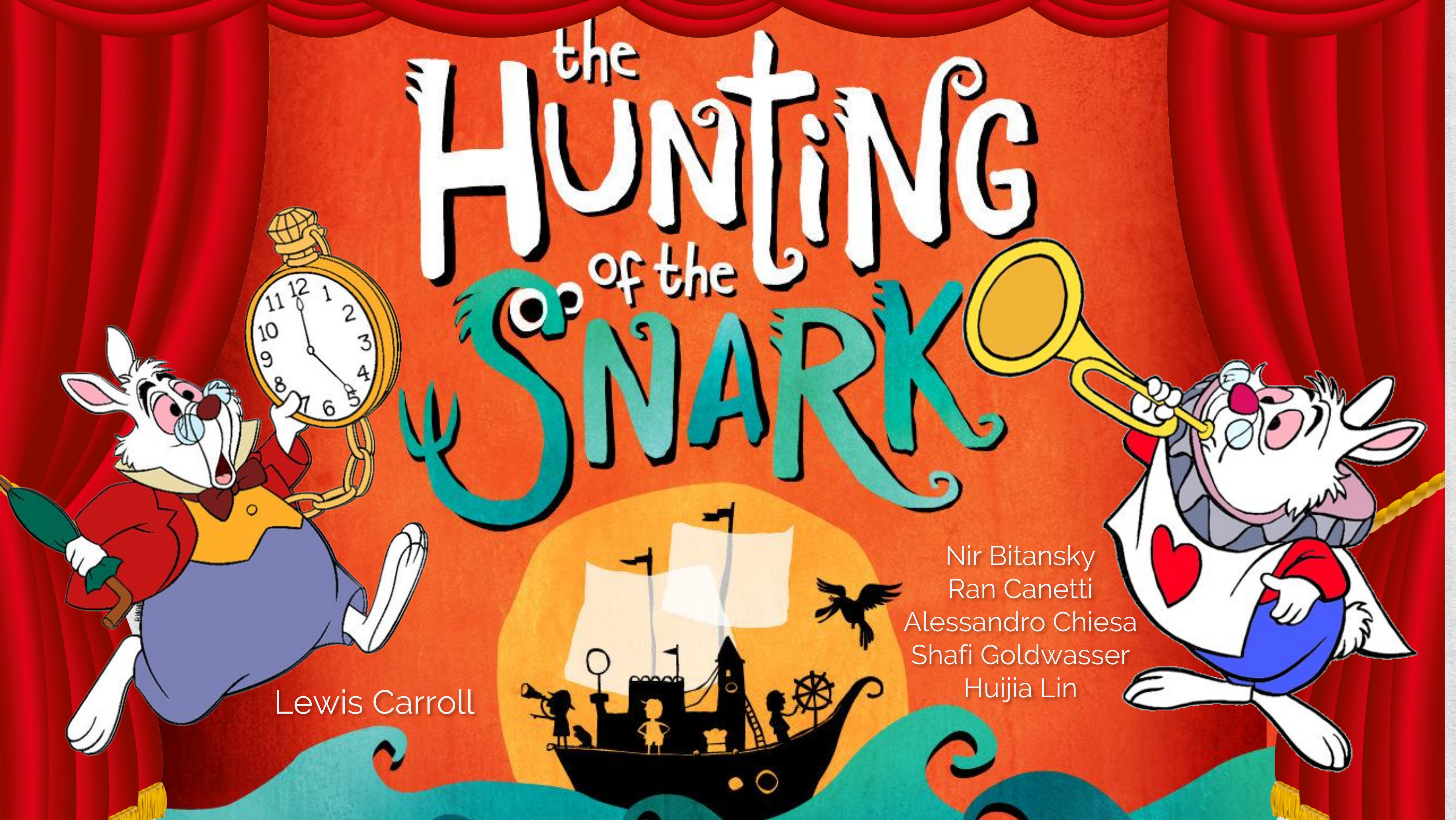
Proof of Knowledge

# Properties for the new tool

Fast

Succinct


Sound

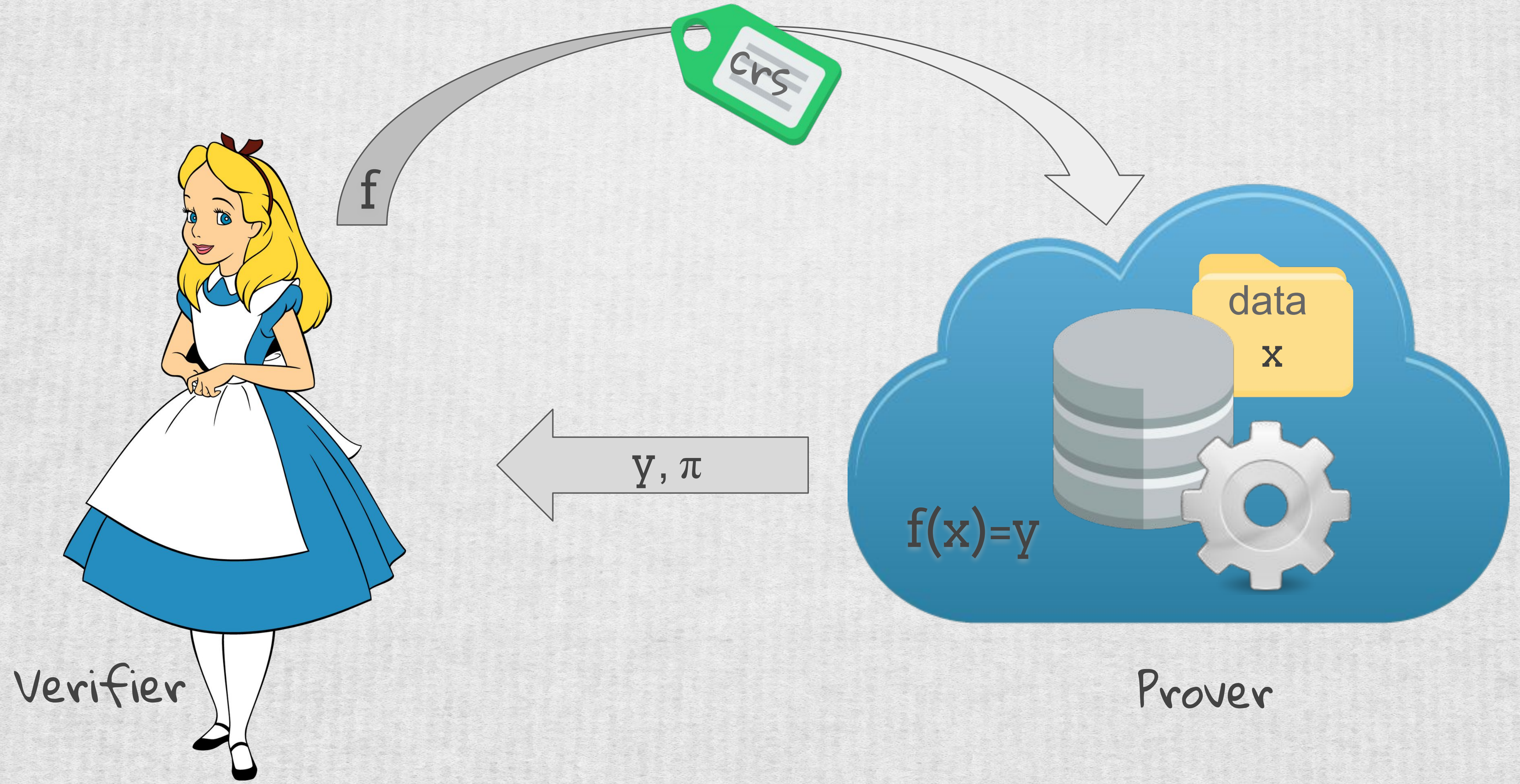


# the Hunting of the Snark

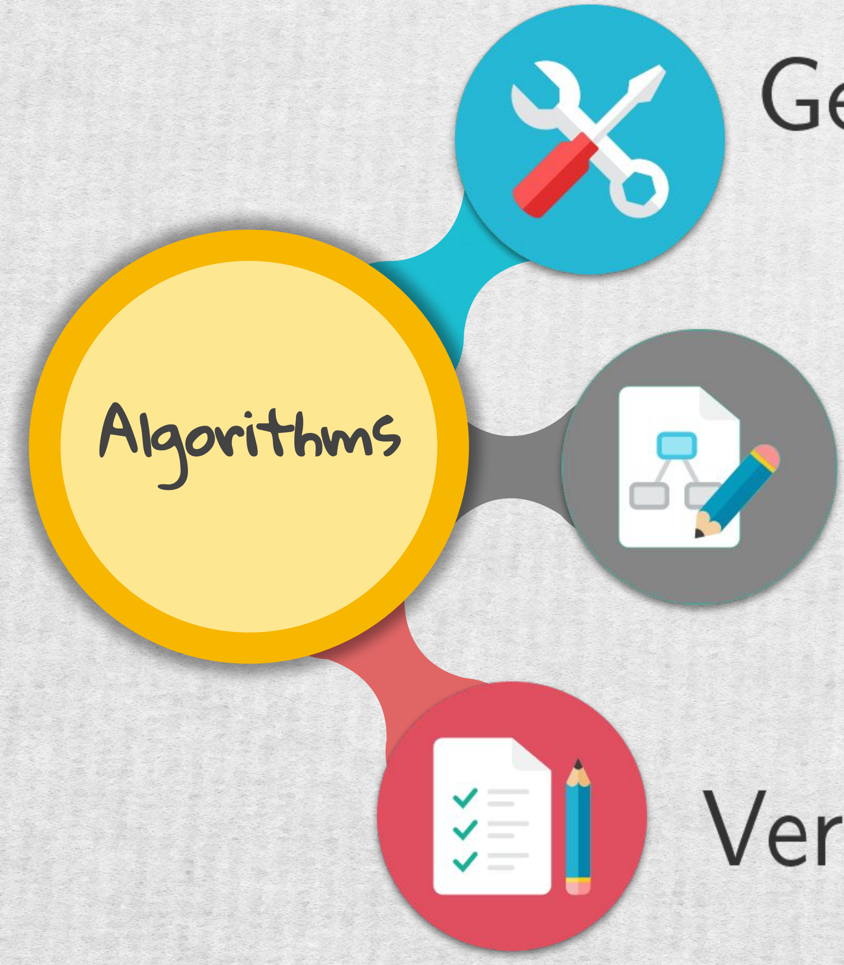
Lewis Carroll

Nir Bitansky
Ran Canetti
Alessandro Chiesa
Shafi Goldwasser
Huijia Lin

# Non-Interactive proofs



# Algorithms of a SNARK

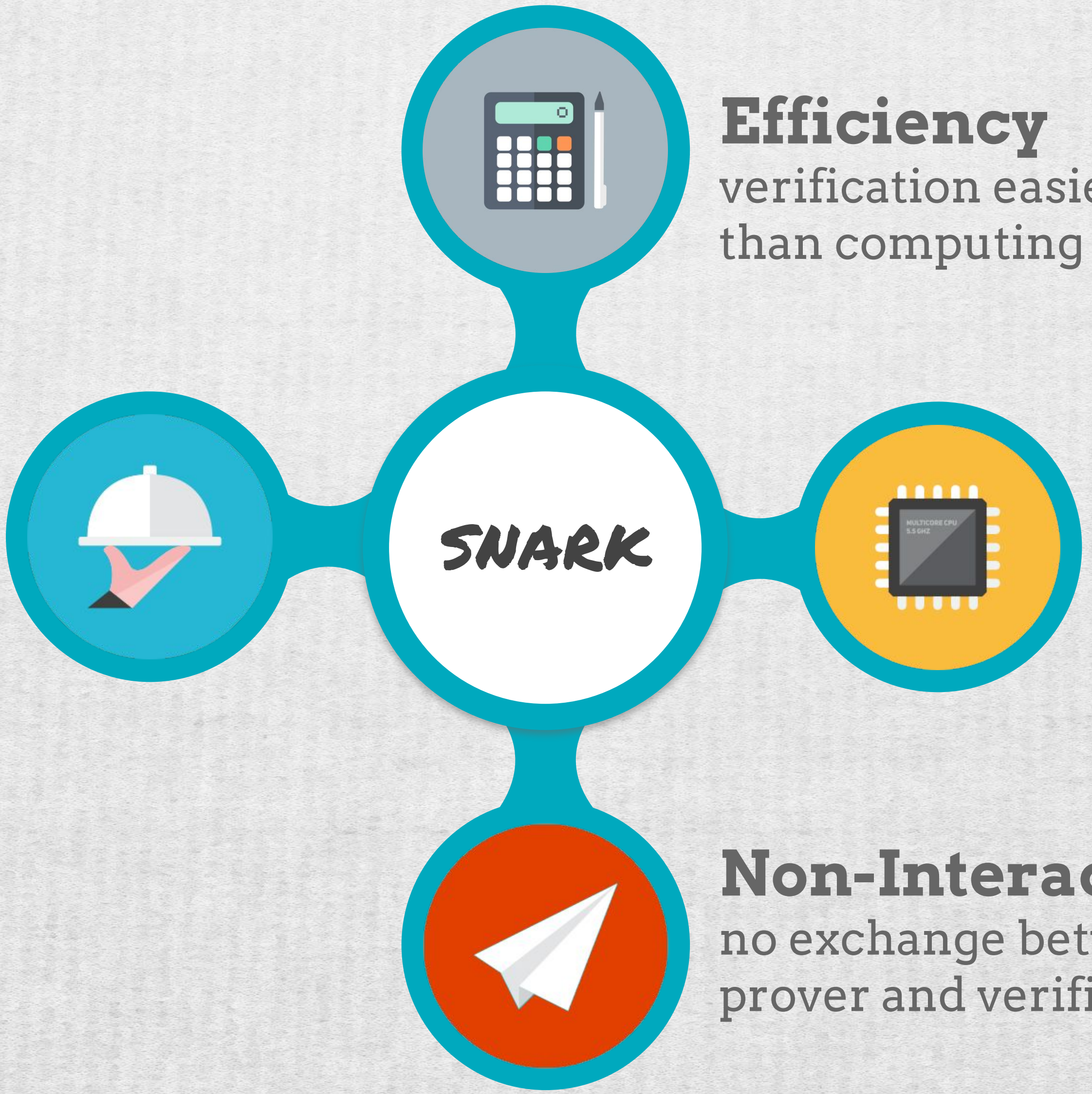



$\mathsf{Gen}(1^\lambda, T) \to \mathsf{crs}$

$\mathsf{Prove}(\mathsf{crs}, y, w) \to \pi : (y, w) \in R$

$\mathsf{Ver}(\mathsf{crs}, y, \pi) \to 0/1$

# SNARK: Succinct Non-interactive ARgument of Knowledge




**Efficiency**
verification easier than computing $f$

**Zero-Knowledge**
does not leak information about the witness

**Succinctness**
proof size independent of NP witness size

**Non-Interactivity**
no exchange between prover and verifier

# Argument of Knowledge Property

$\mathcal{A}$

crs, aux

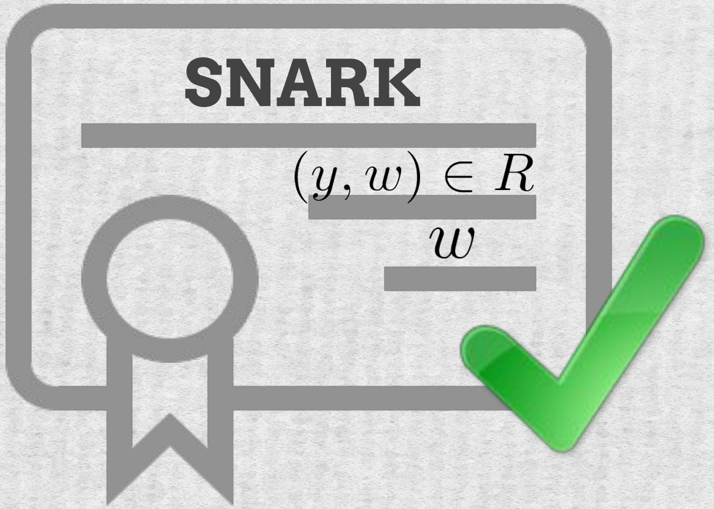
Adversary

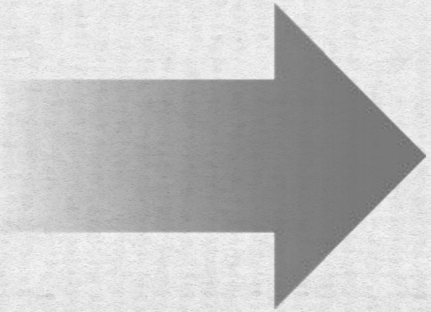
$\mathcal{E}$

crs, aux

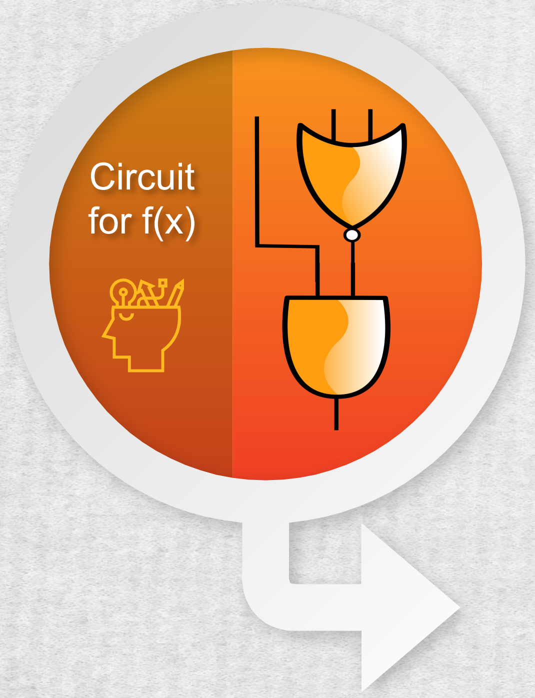
extractor

$\pi$

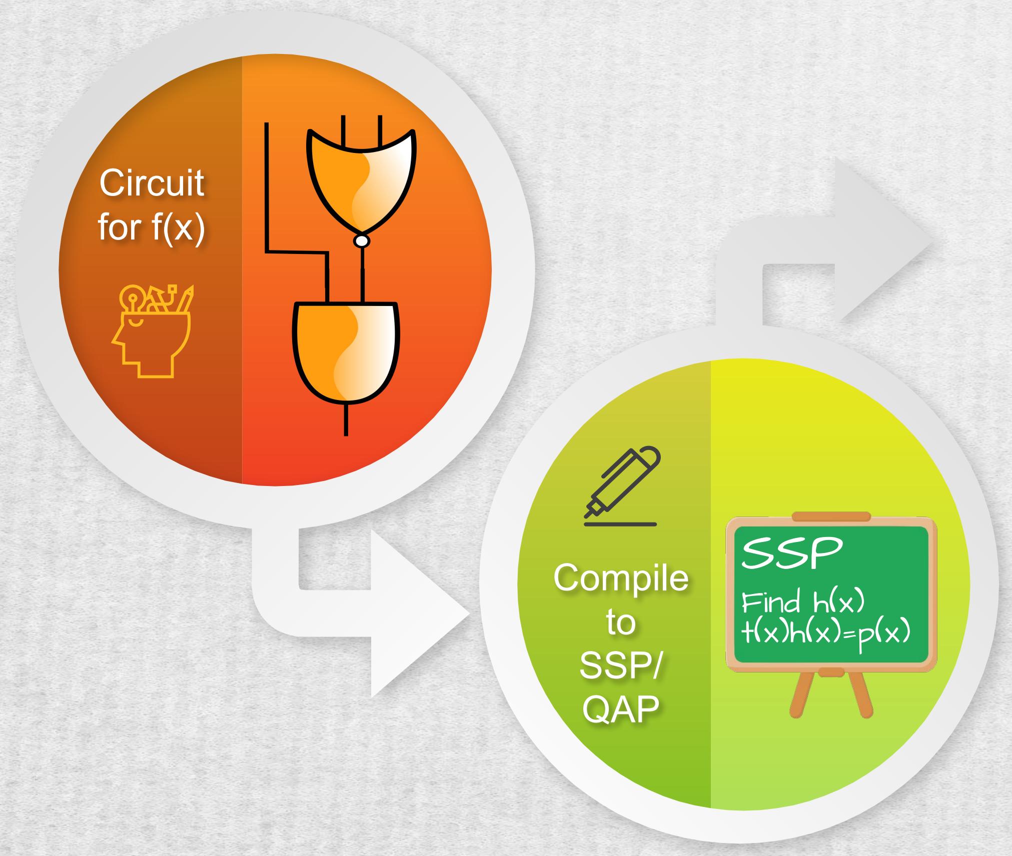


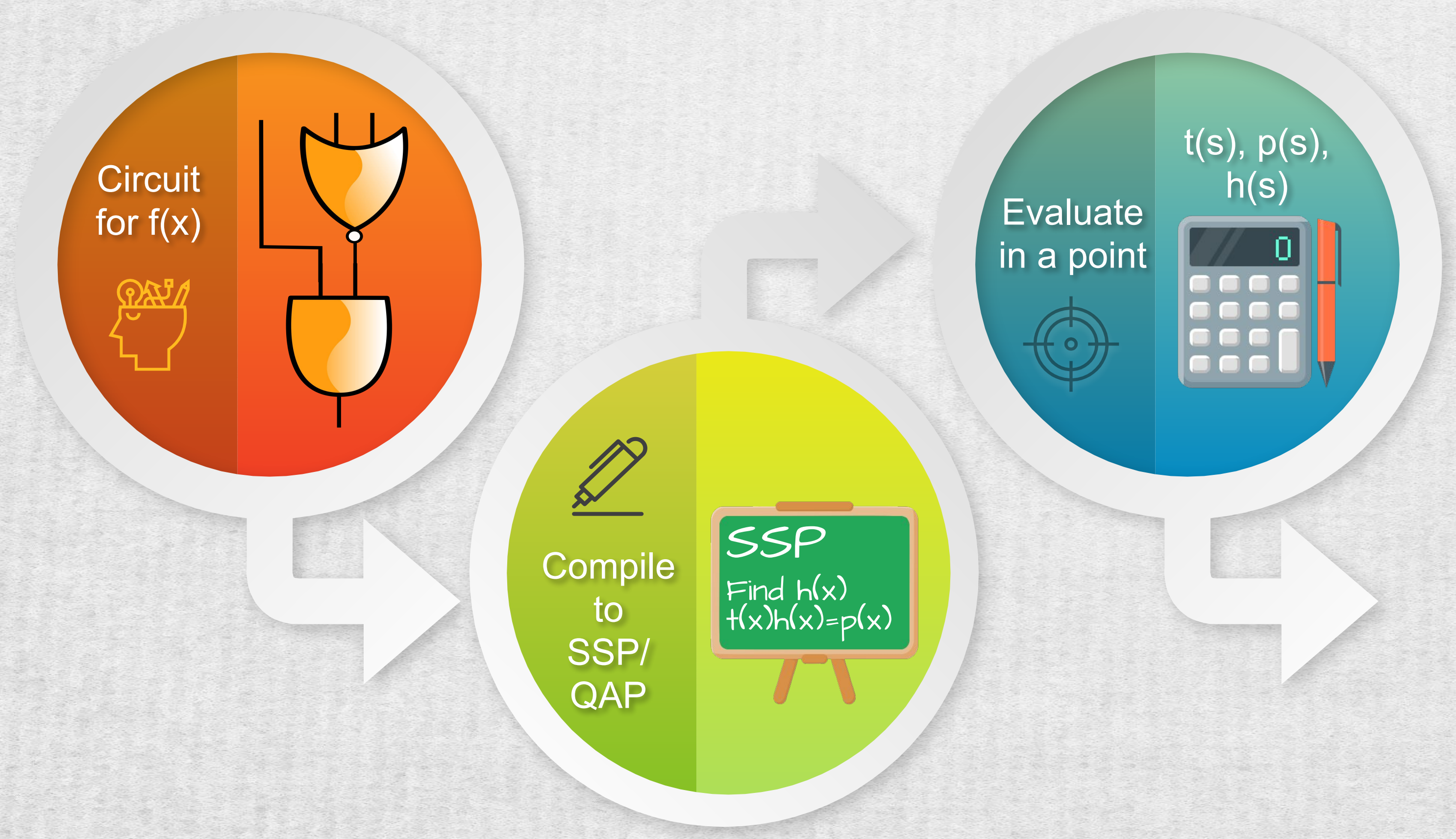
# SNARK: overview of Toolchain



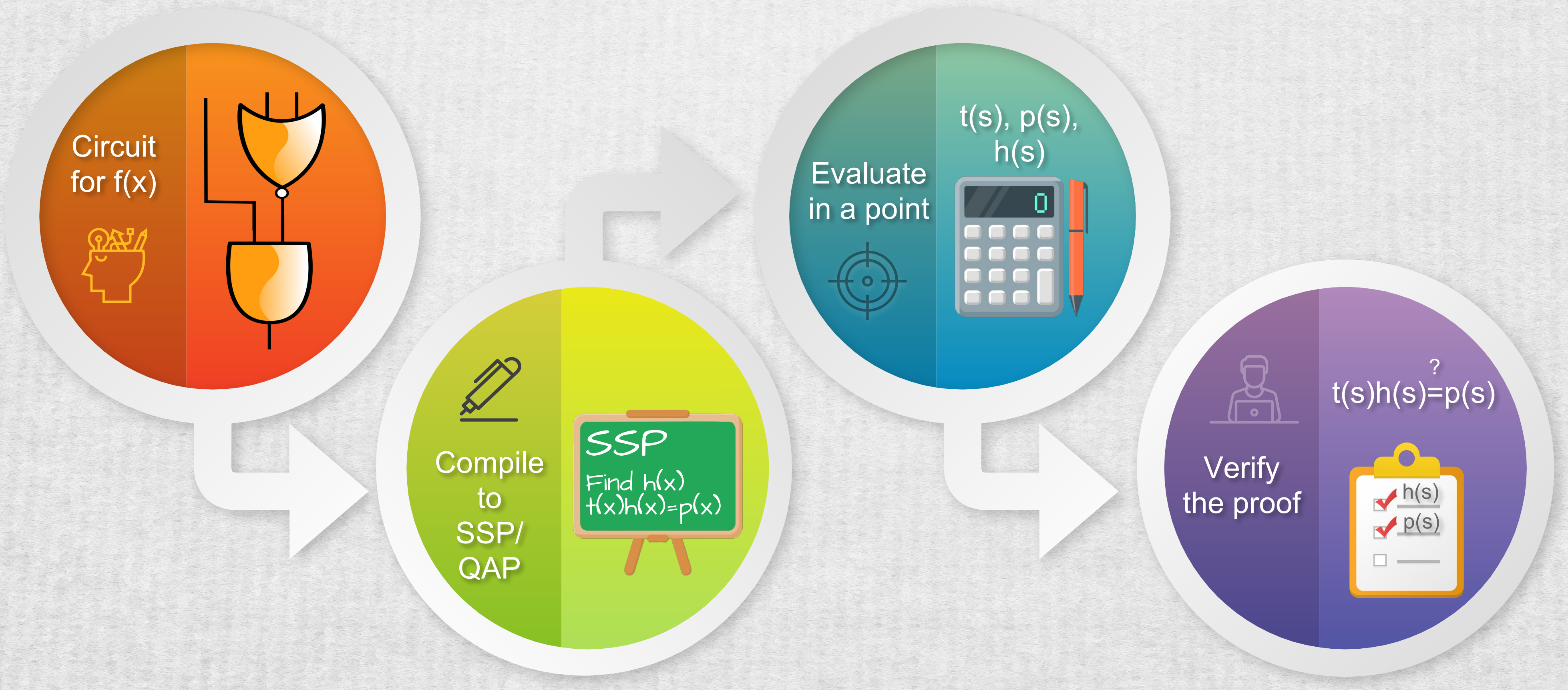
# SNARK: overview of Toolchain



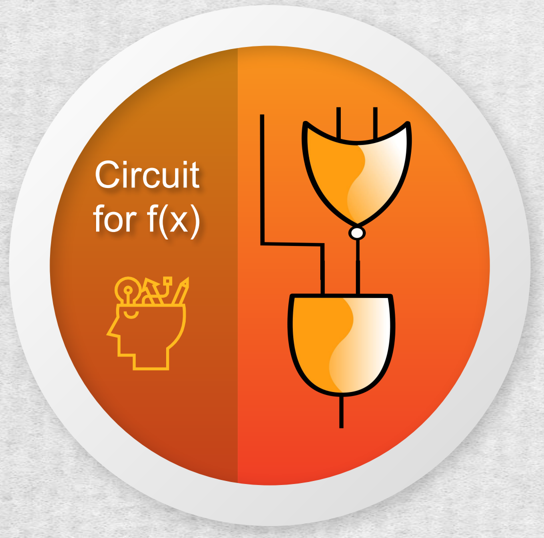
# SNARK: overview of Toolchain



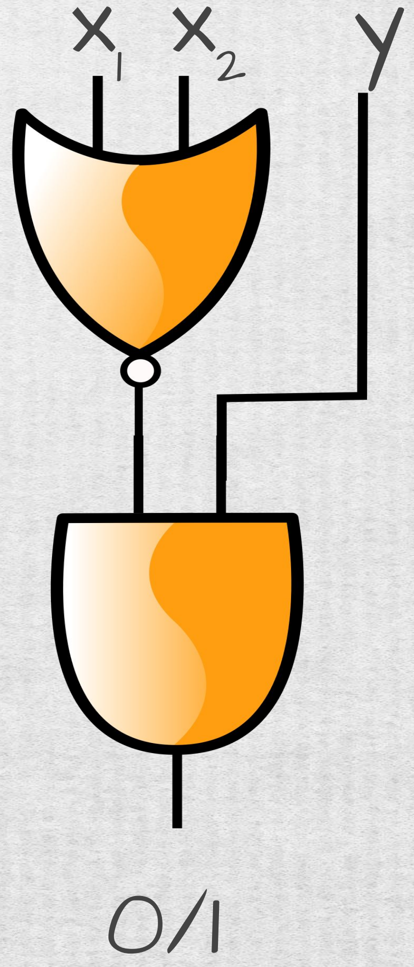
# SNARK: overview of Toolchain



# From Functions to Circuits

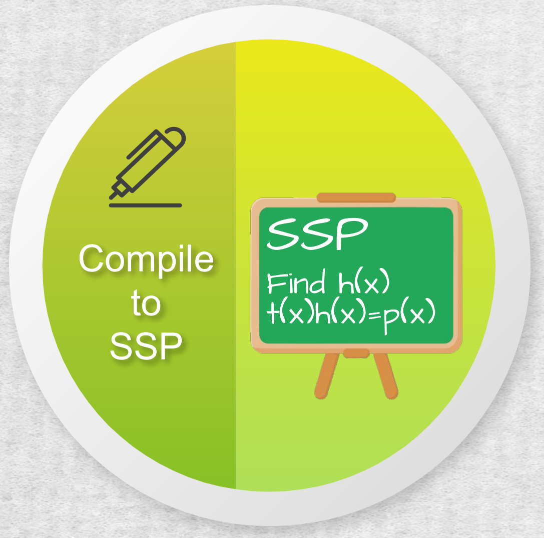


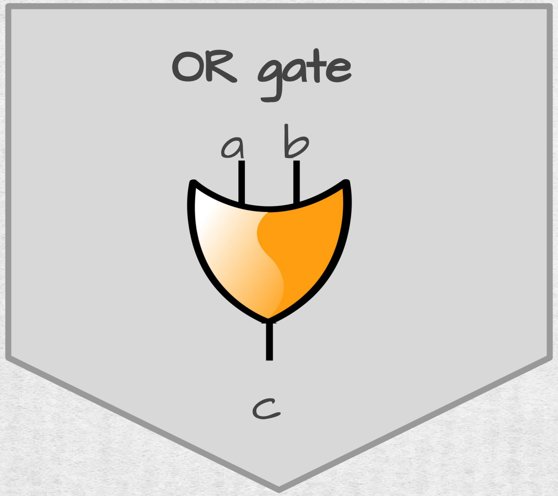

$f(x_1, x_2) = y$

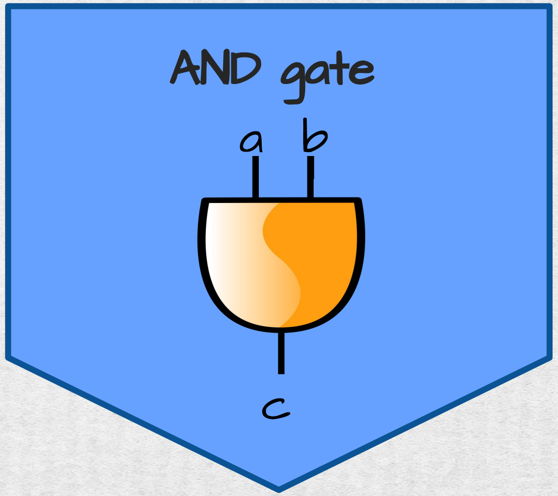


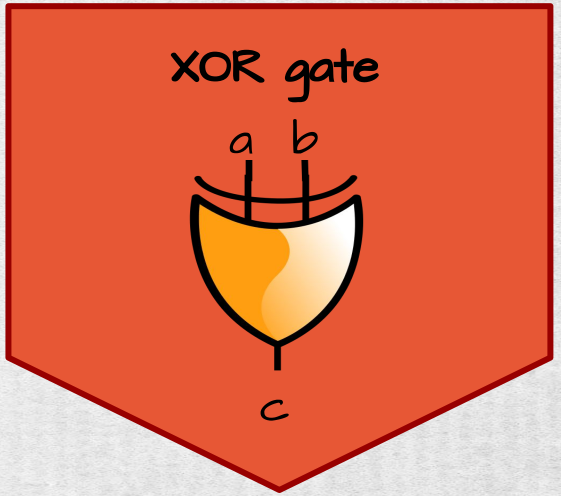

$C(x_1, x_2, y)$

# Step 1. Linearization of logic gates




| a b c | a b c | a b c |
|---|---|---|
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 1 1 | 0 1 0 | 0 1 1 |
| 1 0 1 | 1 0 0 | 1 0 1 |
| 1 1 1 | 1 1 1 | 1 1 0 |
| $-a-b+2c \in \{0,1\}$ | $a+b-2c \in \{0,1\}$ | $a+b+c \in \{0,2\}$ |

# Step 2. Matrix equation for circuit

| OR gate | AND gate | XOR gate | Output gate |
|---|---|---|---|
| $-a-b+2c \in \{0,1\}$ | $a+b-2c \in \{0,1\}$ | $a+b+c \in \{0,2\}$ | $3-3c \in \{0,1\}$ |

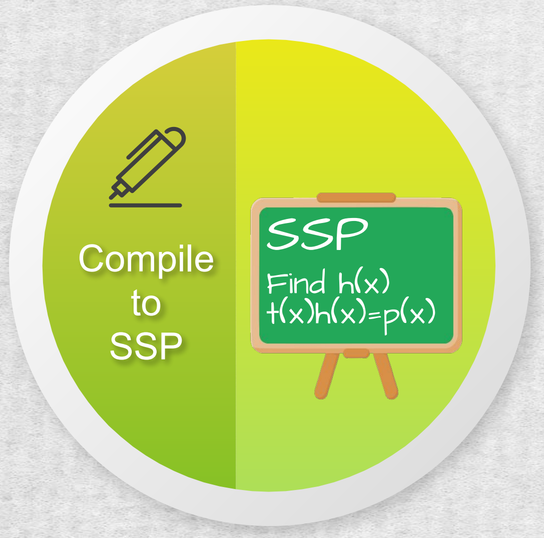
$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$


$$V \, a + \delta \in \{0,2\}^d$$

# Step 2. Matrix equation for circuit

| OR gate | AND gate | XOR gate | Output gate |
|---|---|---|---|
| $-a-b+2c \in \{0,1\}$ | $a+b-2c \in \{0,1\}$ | $a+b+c \in \{0,2\}$ | $3-3c \in \{0,1\}$ |

$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$




$$V\,\mathbf{a} + \boldsymbol{\delta} \in \{0,2\}^d$$

$$\left(V\,\mathbf{a} + \boldsymbol{\delta}\right) \circ \left(V\,\mathbf{a} + \boldsymbol{\delta} - \mathbf{2}\right) = \mathbf{0}$$

# Step 2. Matrix equation for circuit

| OR gate | AND gate | XOR gate | Output gate |
|---|---|---|---|
| $-a-b+2c \in \{0,1\}$ | $a+b-2c \in \{0,1\}$ | $a+b+c \in \{0,2\}$ | $3-3c \in \{0,1\}$ |


$$\left(V\,a + \delta\right) \circ \left(V\,a + \delta - 2\right) = 0$$

$$\left(V\,a + \delta - 1\right) \circ \left(V\,a + \delta - 1\right) = 1$$

# Step 3. Polynomial Problem SSP

$$\left( V\, a + \delta - 1 \right) \circ \left( V\, a + \delta - 1 \right) = 1$$


$$\Downarrow \quad \forall \{r_j\} \in \mathbb{F}^d$$

$$\left( v_0(r_j) + \sum_{i=1}^{m} a_i v_i(r_j) \right)^2 - 1 = 0$$

$$v_0(r_j) = \delta_j - 1 \qquad v_i(r_j) = V_{ji}$$

# Step 3. Polynomial Problem SSP

$$\left(V\mathbf{a} + \boldsymbol{\delta} - \mathbf{1}\right) \circ \left(V\mathbf{a} + \boldsymbol{\delta} - \mathbf{1}\right) = \mathbf{1}$$

$$\Big\Downarrow \ \forall \{r_j\} \in \mathbb{F}^d$$

$$\prod_{j=1}^{d}(x - r_j) \,\Big|\, \left(v_0(x) + \sum_{i=1}^{m} a_i v_i(x)\right)^2 - 1$$

# Step 3. Polynomial Problem SSP

$$\left( V \, a + \delta - 1 \right) \circ \left( V \, a + \delta - 1 \right) = 1$$

$$\Big\Downarrow \ \forall \{r_j\} \in \mathbb{F}^d$$


$$t(x) \;\Big|\; V(x)^2 - 1$$

$$V(x) = v_0(x) + \sum_{i=1}^{m} a_i v_i(x) \qquad t(x) = \prod_{j=1}^{d} (x - r_j)$$

# Step 3. Polynomial Problem SSP

$$\left( V \; a + \delta - 1 \right) \circ \left( V \; a + \delta - 1 \right) = 1$$


SSP: $v_i(x), t(x) \qquad h(x) = ?$

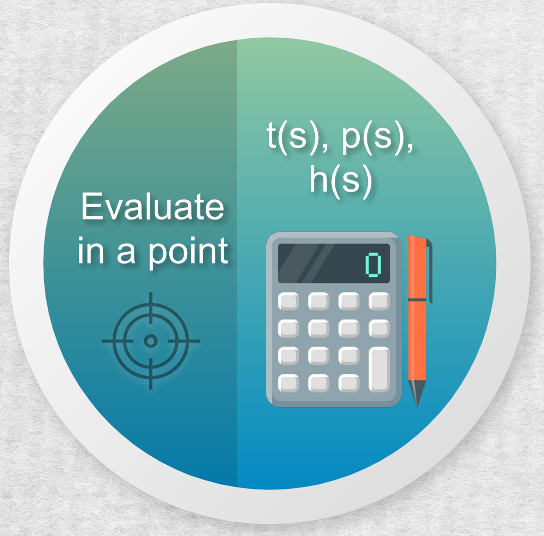
$$h(x)t(x) = p(x)$$

$$p(x) = V(x)^2 - 1 \qquad V(x) = v_0(x) + \sum_{i=1}^{m} a_i v_i(x)$$

# Proving on top of SSP: Setup

SSP: $v_i(x), t(x) \quad h(x) =?$

$$h(x)t(x) = p(x)$$



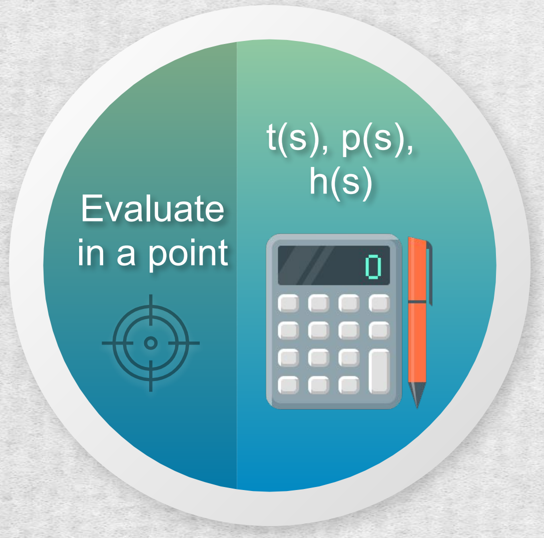

**Prover:** Evaluate the solution in a random <u>unknown</u> point **s**

**Preprocessing:** Publish all necessary powers of **s** (hidden from the **Prover**)

# Proving on top of SSP: Setup

SSP: $v_i(x), t(x) \qquad h(x) = ?$

$$h(x)t(x) = p(x)$$




Enc(s) Enc($s^2$) ... Enc($s^d$)

# Proving on top of SSP: Setup

SSP: $v_i(x), t(x) \qquad h(x) = ?$

$$h(x)t(x) = p(x)$$

Evaluate in a point

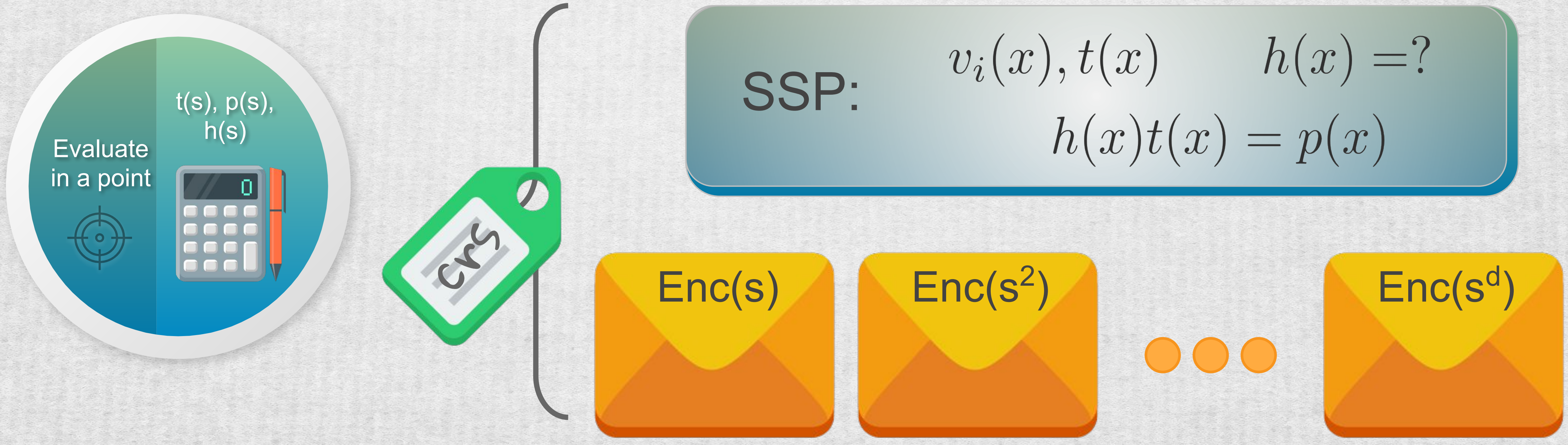
t(s), p(s), h(s)

Enc(s) Enc($s^2$) ... Enc($s^d$)

Encoding:

- ***linear-only*** homomorphic (affine)
- quadratic root detection
- image verification

# Proving on top of SSP: Setup



# Proving on top of SSP: Prover

SSP: $v_i(x), t(x)$ $\quad h(x) = ?$

$$h(x)t(x) = p(x)$$

crs

Enc(s) Enc($s^2$) ... Enc($s^d$)

Evaluate in a point

t(s), p(s), h(s)

Prover

Enc(p(s)) $= \Sigma\, p_j\, \mathrm{Enc}(s^j)$

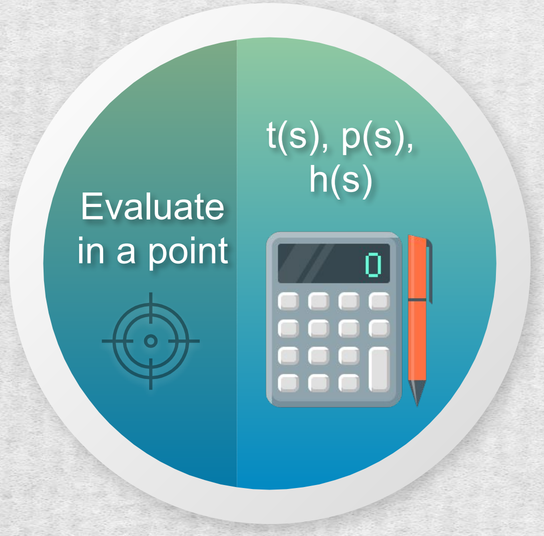
Enc(h(s)) $= \Sigma\, h_j\, \mathrm{Enc}(s^j)$

# Proving on top of SSP: Prover


SSP: $v_i(x), t(x) \qquad h(x) =?$

$$h(x)t(x) = p(x)$$

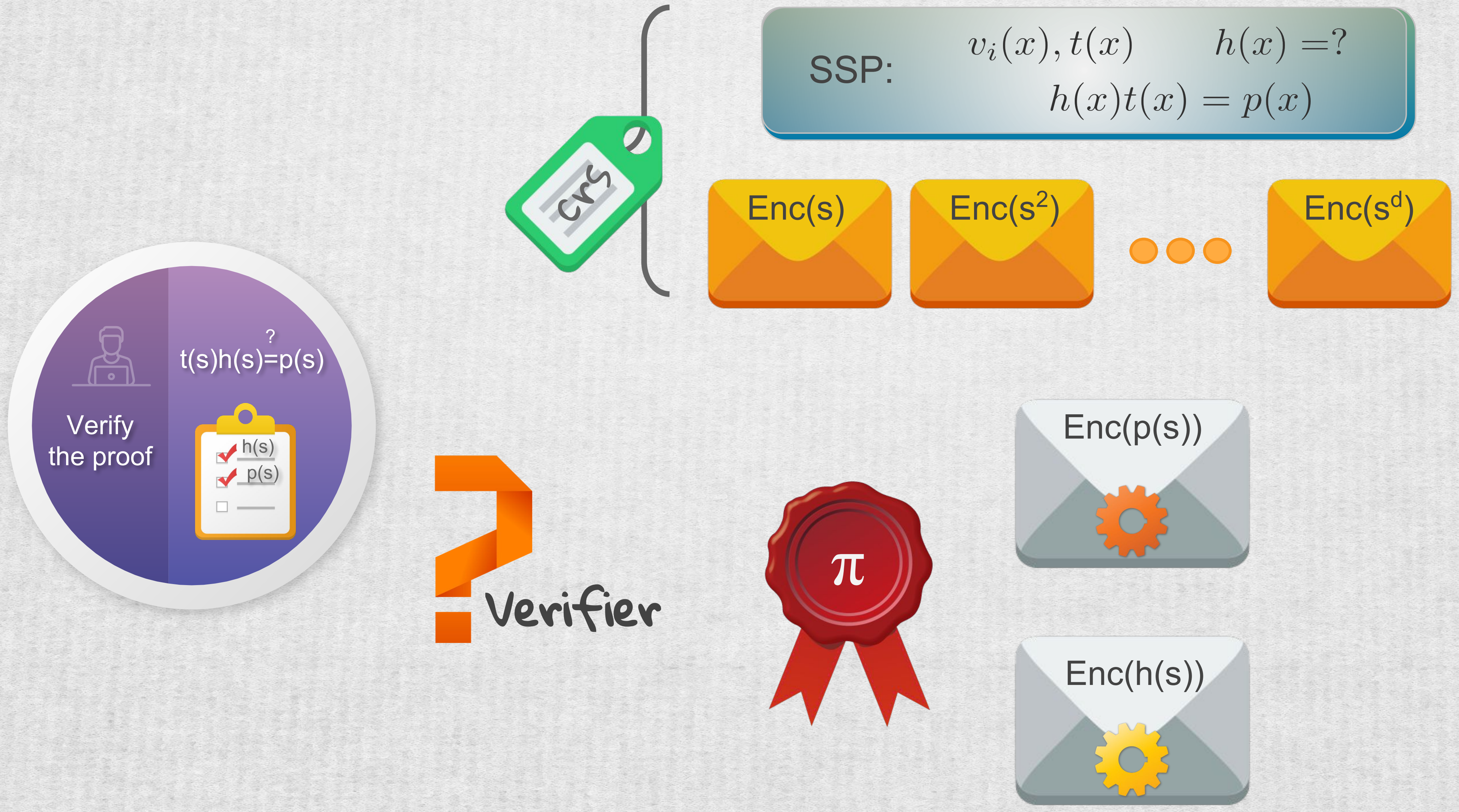
Enc(s) Enc($s^2$) ... Enc($s^d$)




Proof


= Enc(p(s)), Enc(h(s))

# Proving on top of SSP: verifier



# Proving on top of SSP: verifier

SSP: $v_i(x), t(x)$ $\quad h(x) =?$

$$h(x)t(x) = p(x)$$


$Enc(s)$ $\quad Enc(s^2)$ $\quad \dots \quad$ $Enc(s^d)$



$Enc(p(s)) = (\Sigma\, a_i\, Enc(v_i(s)))^2 - 1 \quad ?$

$Enc(h(s)) = Enc(p(s)) / Enc(t(s)) \quad ?$

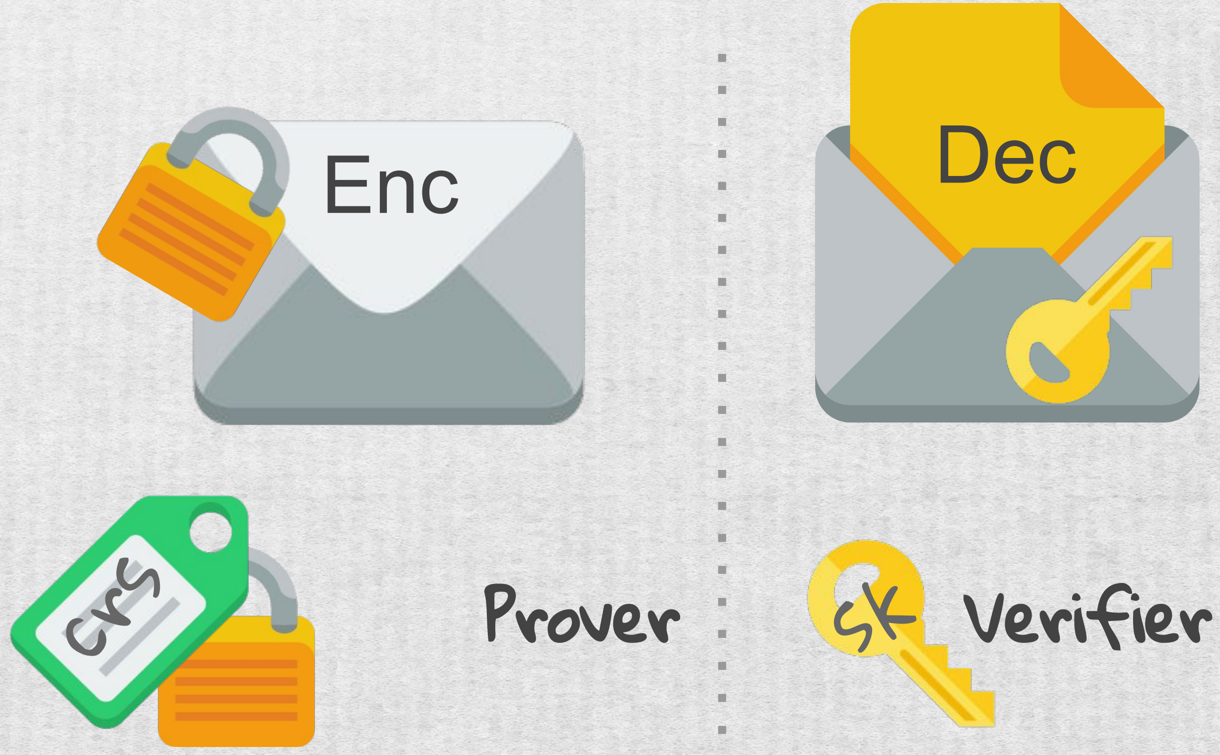
# Security: Types of encodings

Public Verifiable Encoding:

- affine operation using **crs**
- quadratic root detection using **crs**
- image verification using **crs**

Designated Verifiable Encoding:

- affine operation using **crs**
- quadratic root detection needs **sk**
- image verification using **crs**

# Security: Publicly Verifiable Encoding

$$Enc(s) = g^s$$

$$\langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$

$$e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$

$$e(g^a, g^b) = \tilde{g}^{ab}$$


SSP: $v_i(x), t(x)$ $\quad h(x) = ?$

$$h(x)t(x) = p(x)$$


Prover


$$Enc(p(s)) = g^{p(s)} = g^{\sum_i p_i s^i} = \prod (g^{s^i})^{p_i}$$

# Security: Publicly verifiable Encoding

$$Enc(s) = g^s$$

$$\langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$

$$e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$

$$e(g^a, g^b) = \tilde{g}^{ab}$$

crs

SSP: $v_i(x), t(x)$ $\quad h(x) = ?$

$$h(x)t(x) = p(x)$$

$g^s$ $\quad g^{s^2}$ $\quad \dots$ $\quad g^{s^d}$

Prover

crs

$$Enc(p(s)) = g^{p(s)} = g^{\sum_i p_i s^i} = \prod (g^{s^i})^{p_i}$$

Verifier

crs

$$e(g^{t(s)}, g^{h(s)}) \stackrel{?}{=} e(g^{p(s)}, g)$$

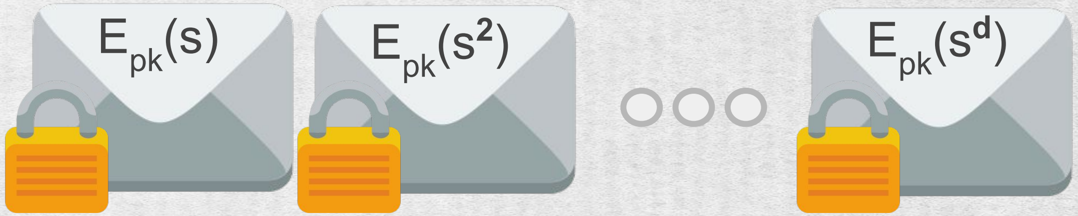
# Security: Designated Verifiable Encoding

Encryption: $E_{pk}(m) = c$

Decryption: $D_{sk}(c) = m$


SSP: $v_i(x), t(x)$ $\quad h(x) = ?$

$h(x)t(x) = p(x)$




Prover


$$Enc(p(s)) = E_{pk}(p(s)) = E_{pk}(\sum p_i s^i) = \sum p_i E_{pk}(s^i)$$

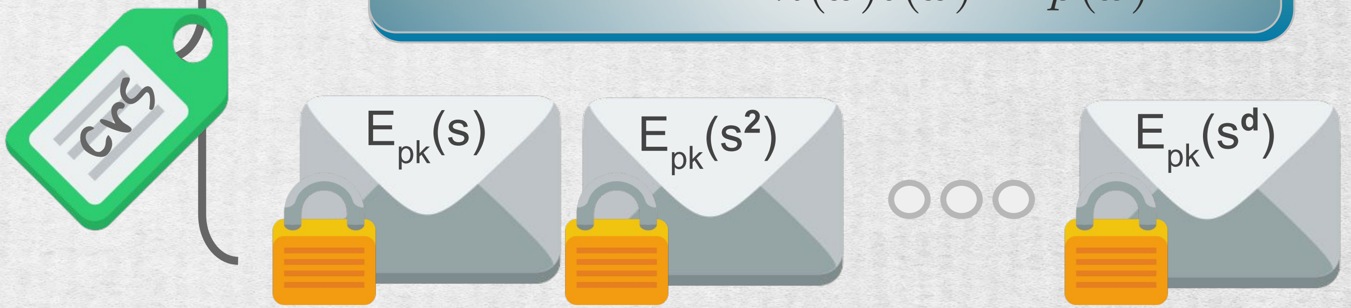
# Security: Designated Verifiable Encoding

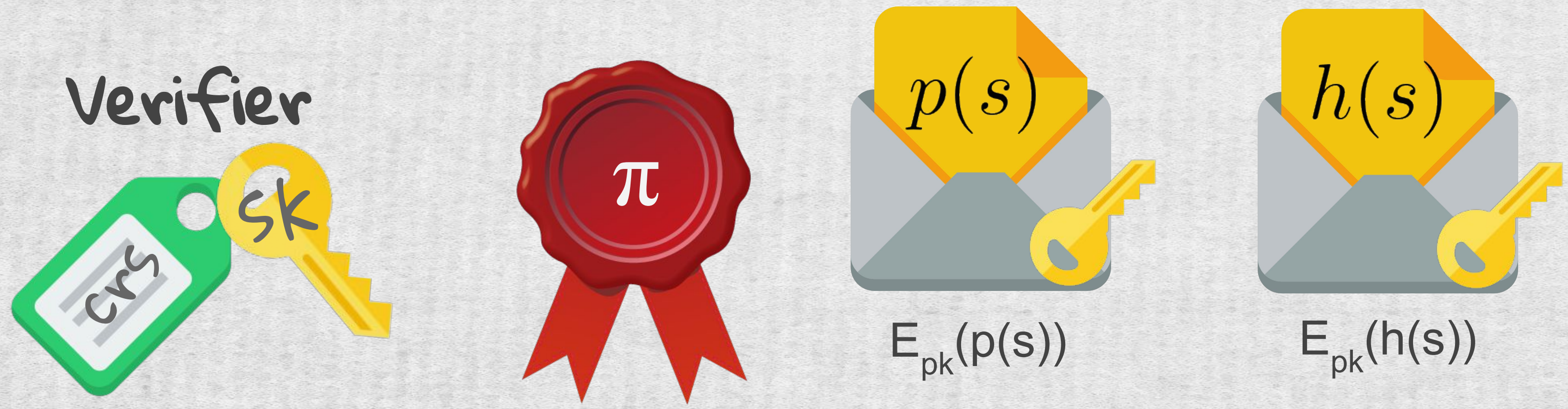
Encryption: $E_{pk}(m) = c$

Decryption: $D_{sk}(c) = m$

SSP: $v_i(x), t(x)$ $\quad$ $h(x) = ?$

$h(x)t(x) = p(x)$




$$t(s)h(s) \stackrel{?}{=} p(s)$$

# SNARKs: Further Directions

**Standard SNARKs**

based on DLog in EC groups
not quantum resistant
publicly-verifiable
zero-knowledge

**Post-Quantum SNARKs**

based on lattice assumptions
designated-verifiable
zero-knowledge

# Post-Quantum SNARKs from Lattice-based Encodings

error

Encryption: $E_{\vec{s}}(m) = (-\vec{a}, \vec{a}\vec{s} + pe + m), \;\; e \in \chi$

Decryption: $D_{\vec{s}}((\vec{c_0}, c_1)) = \vec{c_0} \cdot \vec{s} + c_1 \pmod{p}$

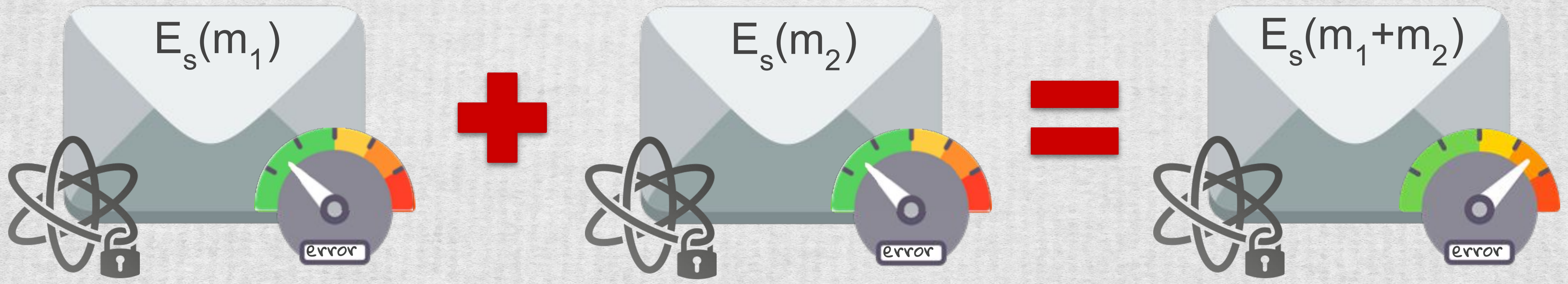
# Post-Quantum SNARKs from Lattice-based Encodings

error

Encryption: $E_{\vec{s}}(m) = (-\vec{a}, \vec{a}\vec{s} + pe + m), \; e \in \chi$

Decryption: $D_{\vec{s}}((\vec{c_0}, c_1)) = \vec{c_0} \cdot \vec{s} + c_1 \quad \pmod{p}$

$$E_{\vec{s}}(m_1) + E_{\vec{s}}(m_2) = (-\vec{a}_1 - \vec{a}_2, (\vec{a}_1 + \vec{a}_1)\vec{s} + p\boxed{(e_1 + e_2)} + m_1 + m_2)$$



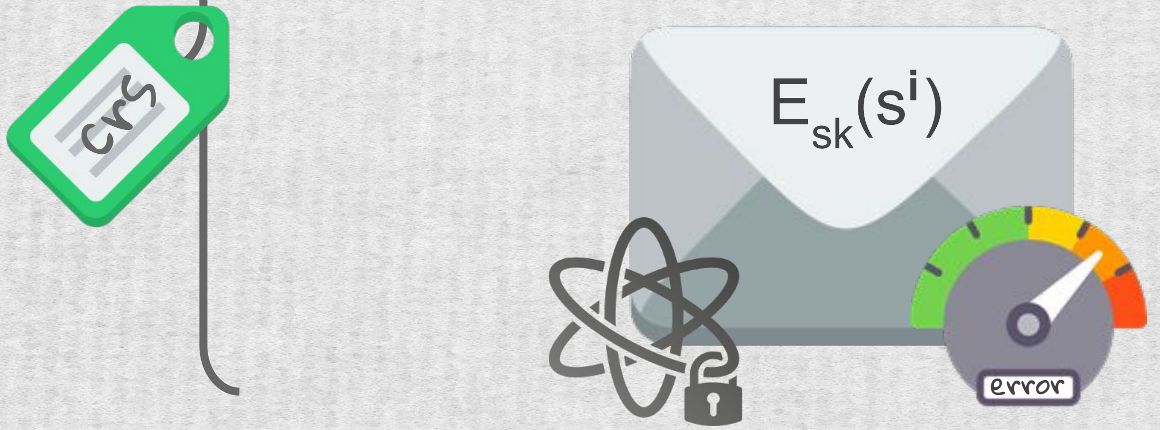
# Post Quantum SNARKs from Lattice-based Encodings

Encryption: $E_{\vec{s}}(m) = (-\vec{a}, \vec{a}\vec{s} + pe + m), \;\; e \in \chi$

Decryption: $D_{\vec{s}}((\vec{c_0}, c_1)) = \vec{c_0} \cdot \vec{s} + c_1 \pmod{p}$

SSP: $v_i(x), t(x) \qquad h(x) = ?$

$h(x)t(x) = p(x)$



# SNARKs: Further Directions

## Standard SNARKs
based on DLog in EC groups
not quantum resistant
publicly-verifiable
zero-knowledge

## Publicly Verifiable
post-quantum SNARKs ???

## Post-Quantum SNARKs
based on lattice assumptions
designated-verifiable
zero-knowledge

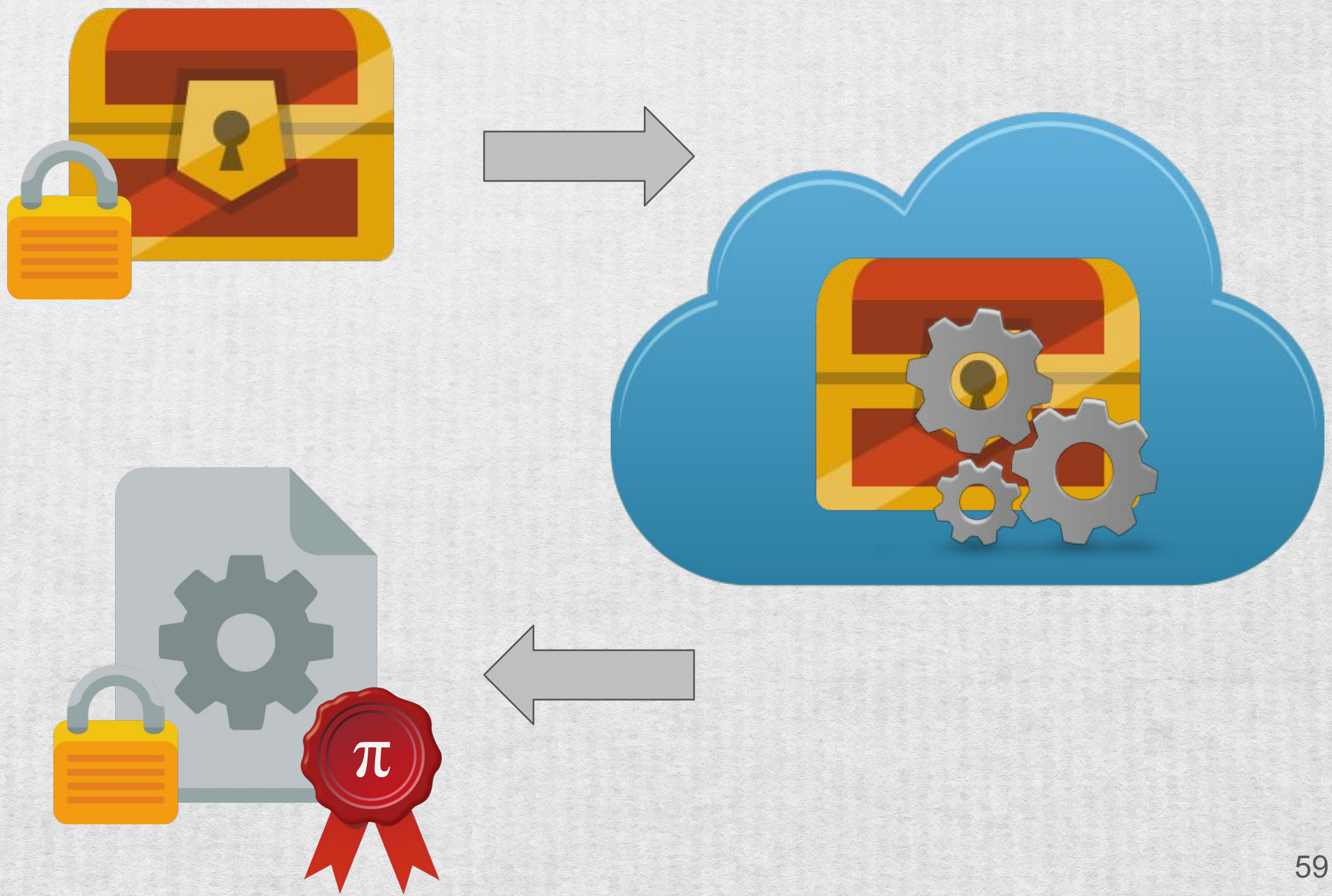
## SNARKs with privacy for the data
for computations over ciphertexts

prove integrity of the result efficiently

# SNARKs for computations on encrypted data

**Confidentiality**

Fully Homomorphic Encryption




**Integrity**

Proof of Knowledge

# More trustful Cloud

# THANK YOU

www.di.ens.fr/~nitulesc