Lattice-Based zk-SNARKs from Square Span Programs

joint work with Rosario Gennaro, Michele Minelli, Michele Orrù

Anca Nitulescu
ENS Paris

ENS

cnrs

Inria
INVENTORS FOR THE DIGITAL WORLD

PSL
RESEARCH UNIVERSITY

Outline

Motivation

Cloud Computing

SNARKs

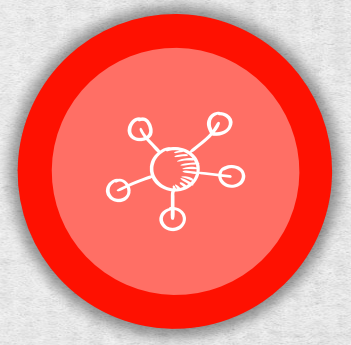
Definition

Construction Steps

Encodings

Examples
Properties
Assumptions

Security Reduction



Our Result

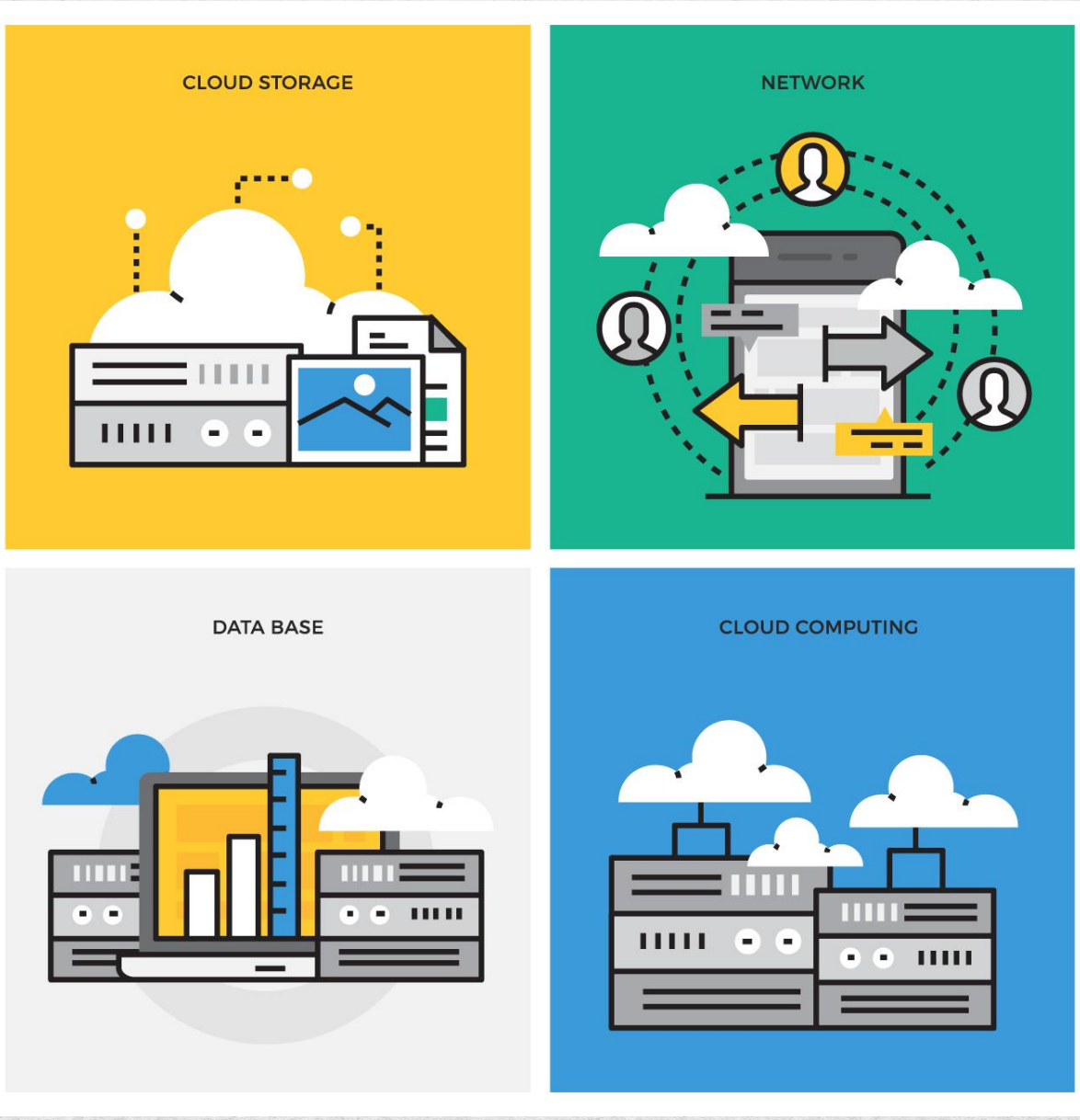
Post-quantum SNARK

Construction
Difficulties
Comparison
Open Problems

Conclusions

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

- safely store the data
- securely process the data
- answer correctly to 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

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?**

Delegated Computation

Client

Worker

Delegated Computation

Client

Compute f(x)

Worker

Delegated Computation

Client

"I have the

result **y**=f(**x**)"

Worker

Unreliable worker

Client

$y^* \neq f(x)$

Corrupted Worker

Ask for a proof

Verify π

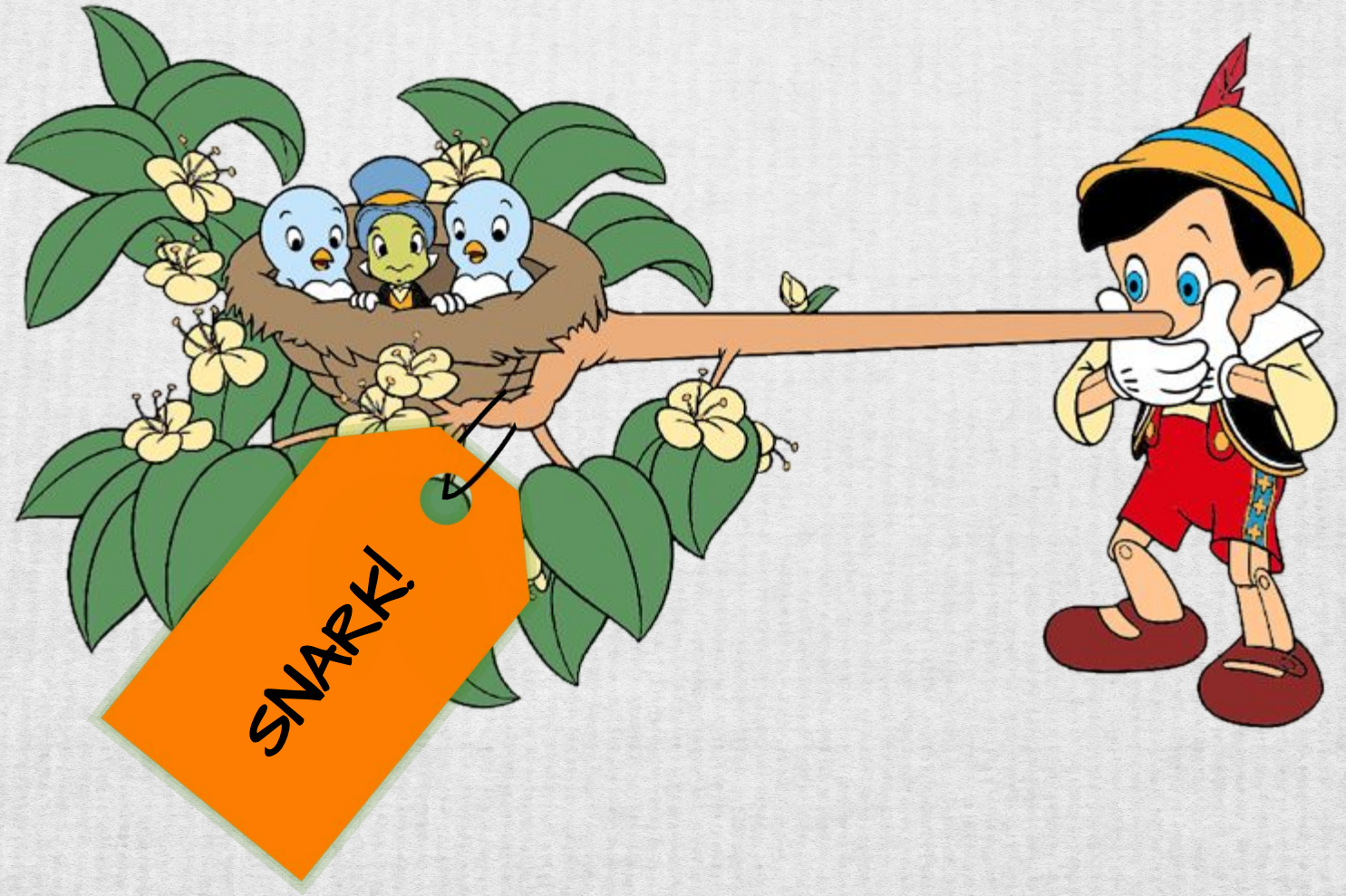
$y^* \neq f(x)$

Proof π

Client
Verifier

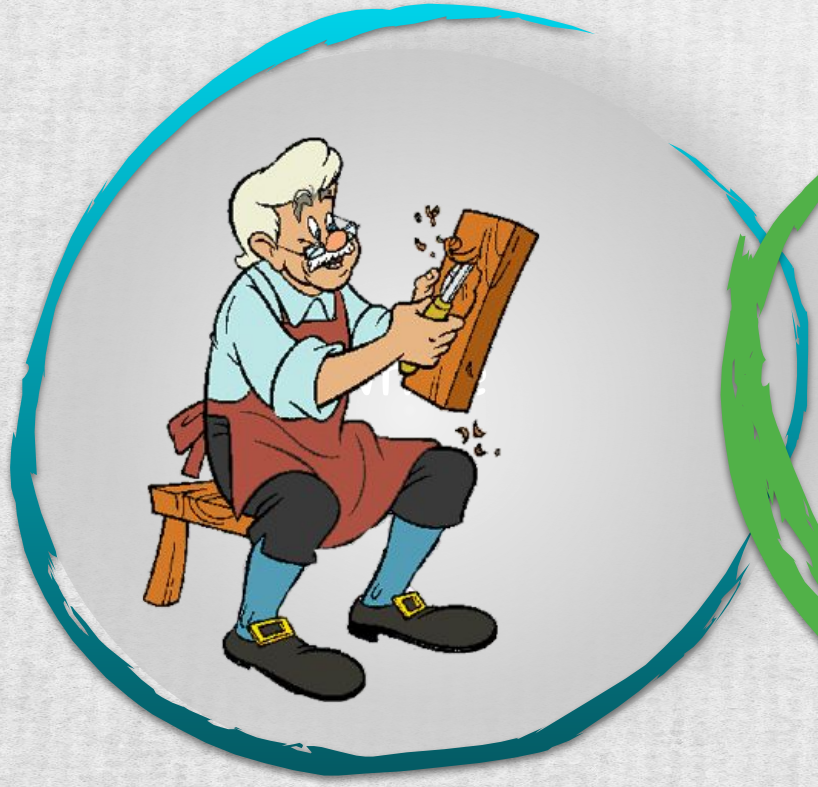
Worker
Prover

Integrity for Computation

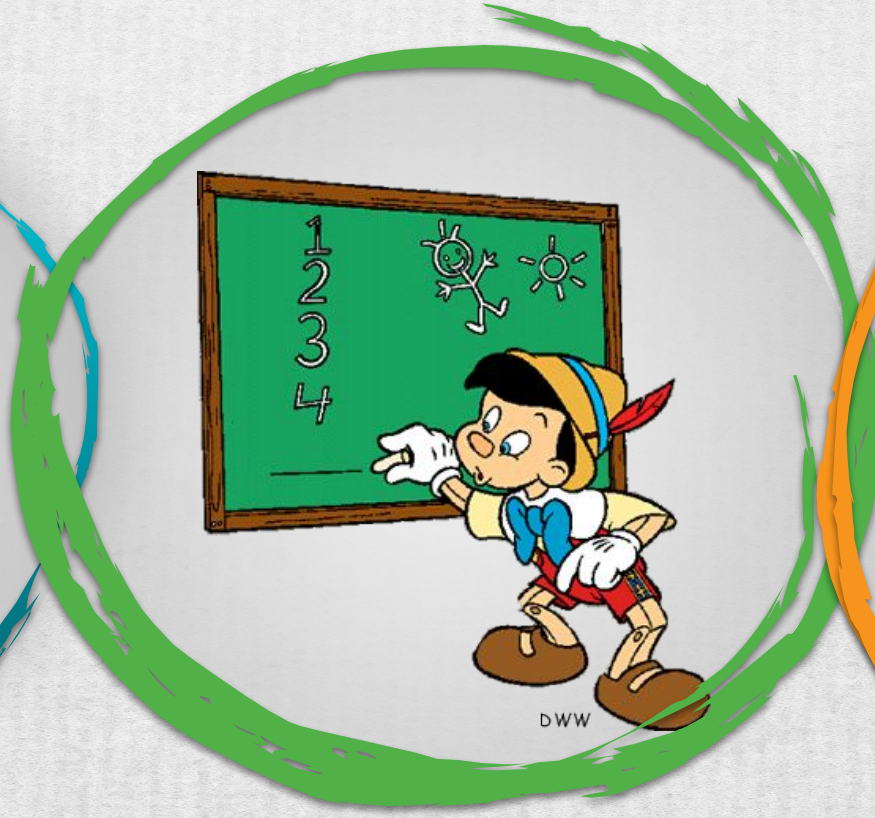


SNARK: Properties of a Proof

Fast

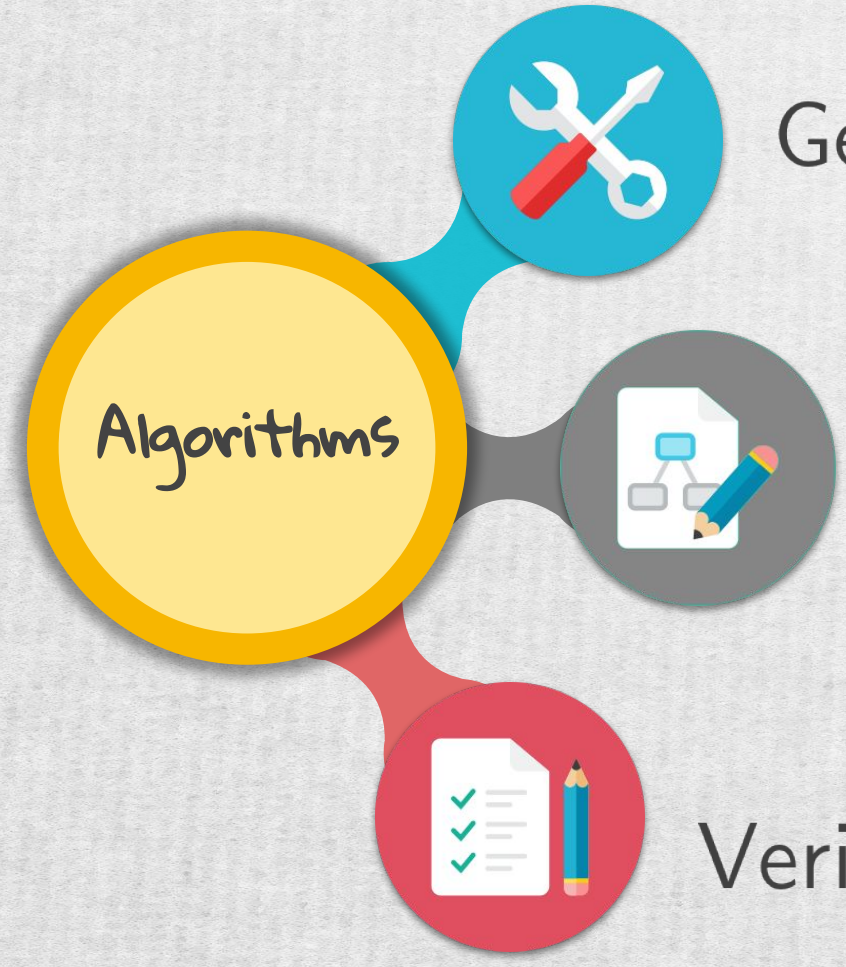


Sound



Succinct

SNARK: Succinct Non-interactive ARgument of Knowledge



$$\mathsf{Gen}(1^\lambda, \mathcal{R}) \to (\mathsf{crs}, \mathsf{vk})$$

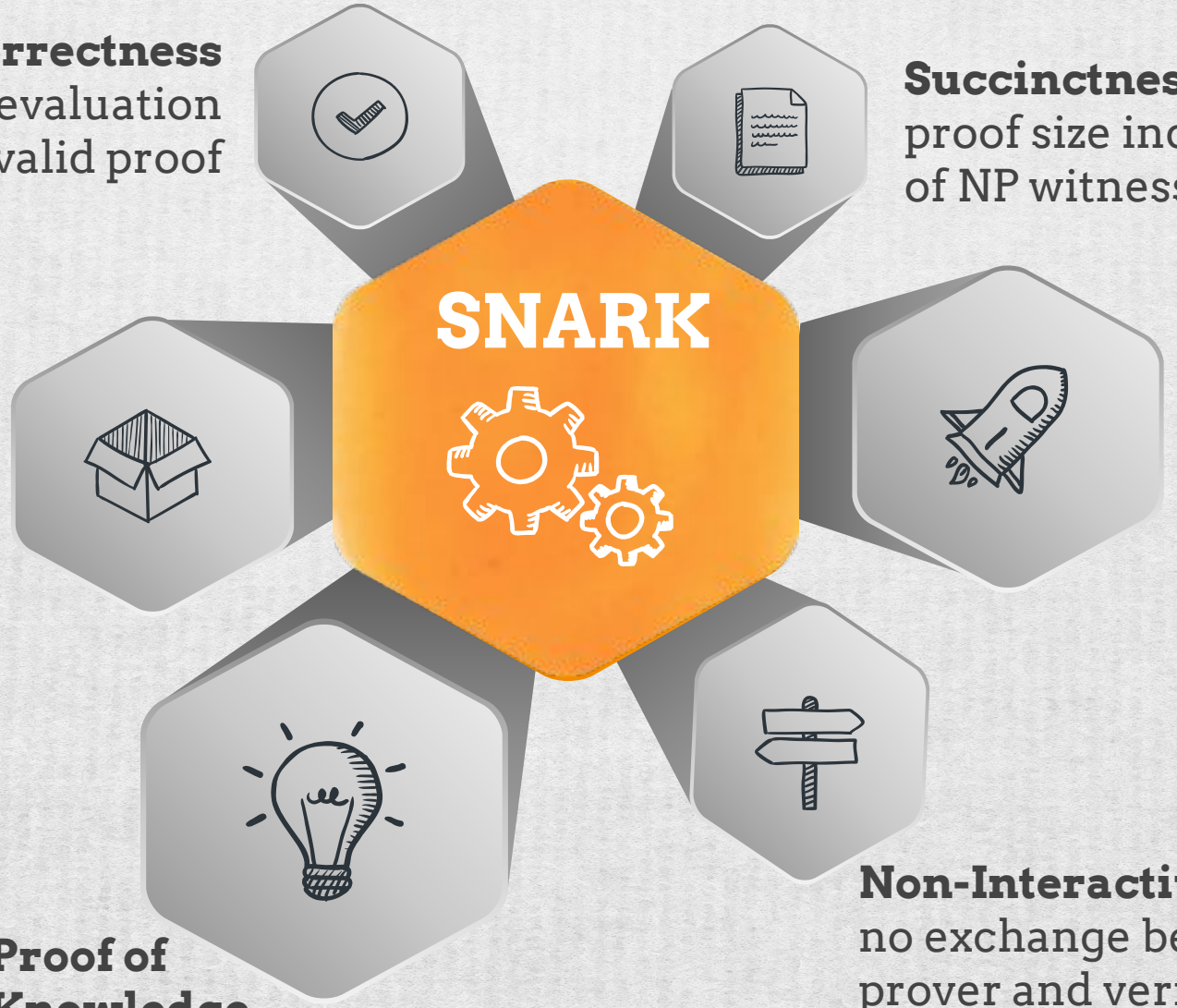
$$\mathsf{Prove}(\mathsf{crs}, y, w) \to \pi : (y, w) \in R$$

$$\mathsf{Verify}(\mathsf{vk}, y, \pi) \to 0/1$$

SNARK

Correctness
any correct evaluation **f(x)** has a valid proof

Succinctness
proof size independent of NP witness size



Zero-Knowledge
does not leak anything about the witness

Efficiency
verification faster than computing **f(x)**

Proof of Knowledge

Non-Interactivity
no exchange between prover and verifier

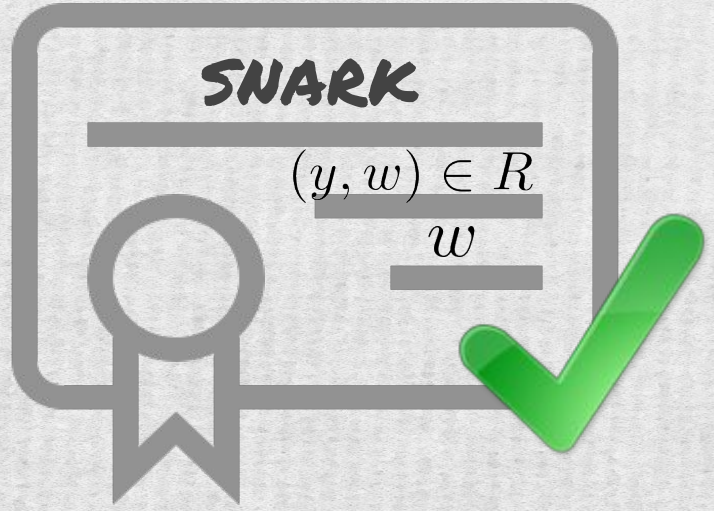
Argument of Knowledge

$\mathcal{A}$

crs, aux

Adversary

π



Argument of Knowledge

$\mathcal{A}$

crs, aux

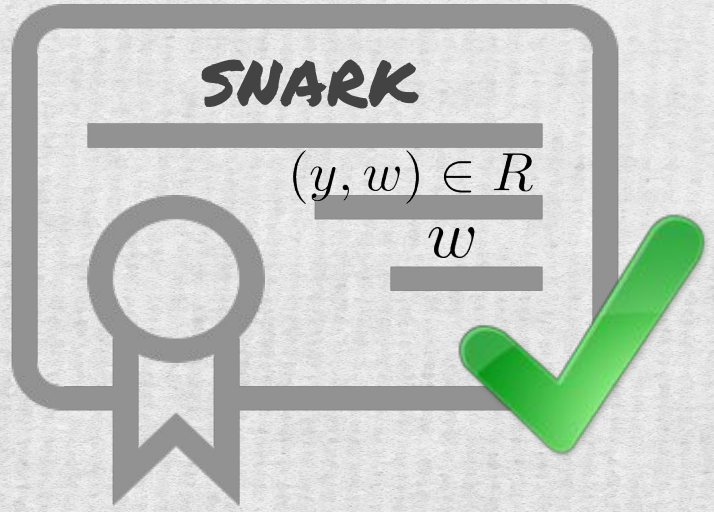
Adversary

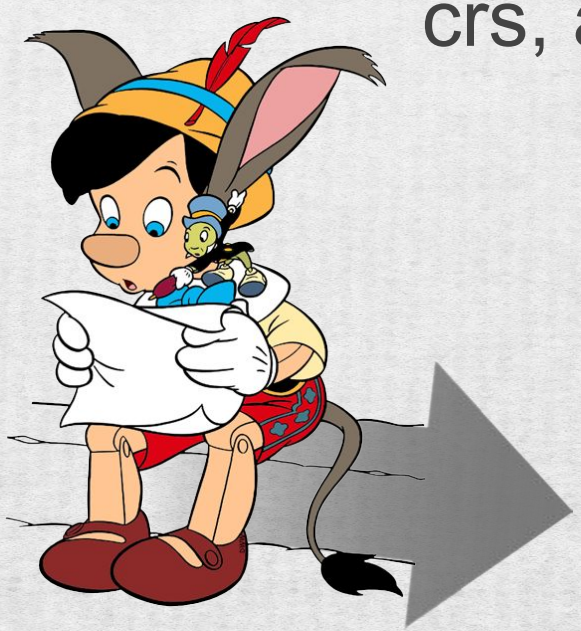
$\mathcal{E}$

crs, aux

extractor

π





Efficient Constructions: Pinocchio, Geppetto

Pinocchio:
Nearly Practical
Verifiable
Computation

Bryan Parno,
Jon Howell,
Craig Gentry,
Mariana Raykova

Geppetto: Versatile
Verifiable
Computation

Craig Costello,
Cédric Fournet,
Jon Howell,
Markulf Kohlweiss,
Benjamin Kreuter, Michael
Naehrig,
Bryan Parno,
Samee Zahur

Quantum Attacks

Existent SNARKs:

- zero-knowledge
- publicly-verifiable (only **crs**)
- based on DLog in EC groups
- **not quantum resistant**

Quantum Attacks

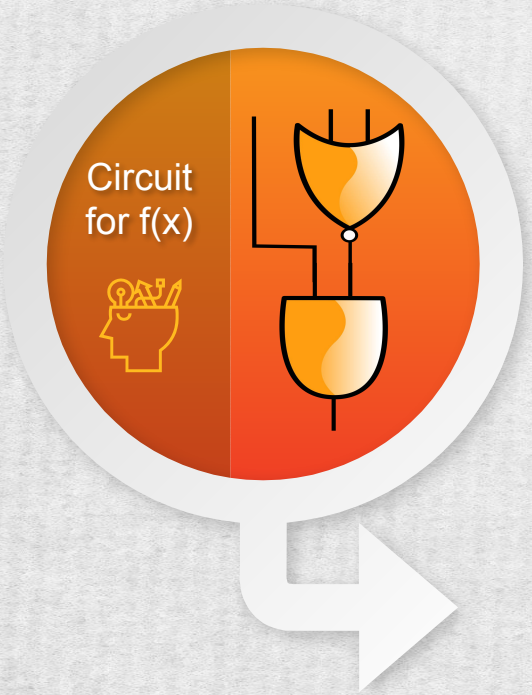
Existent SNARKs:

- zero-knowledge
- publicly-verifiable (only **crs**)
- based on DLog in EC groups
- **not quantum resistant**

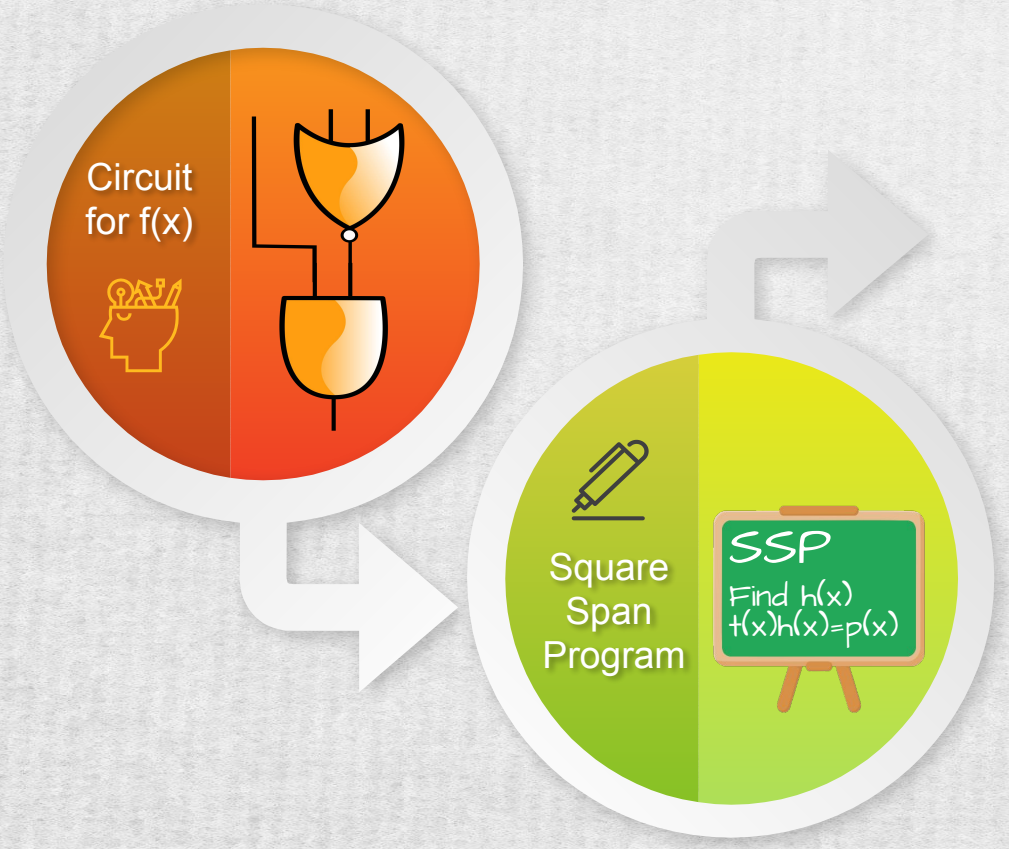
Post-Quantum SNARKs:

- based on **lattice assumptions**
- designated-verifiable (**vk**)
- zero-knowledge

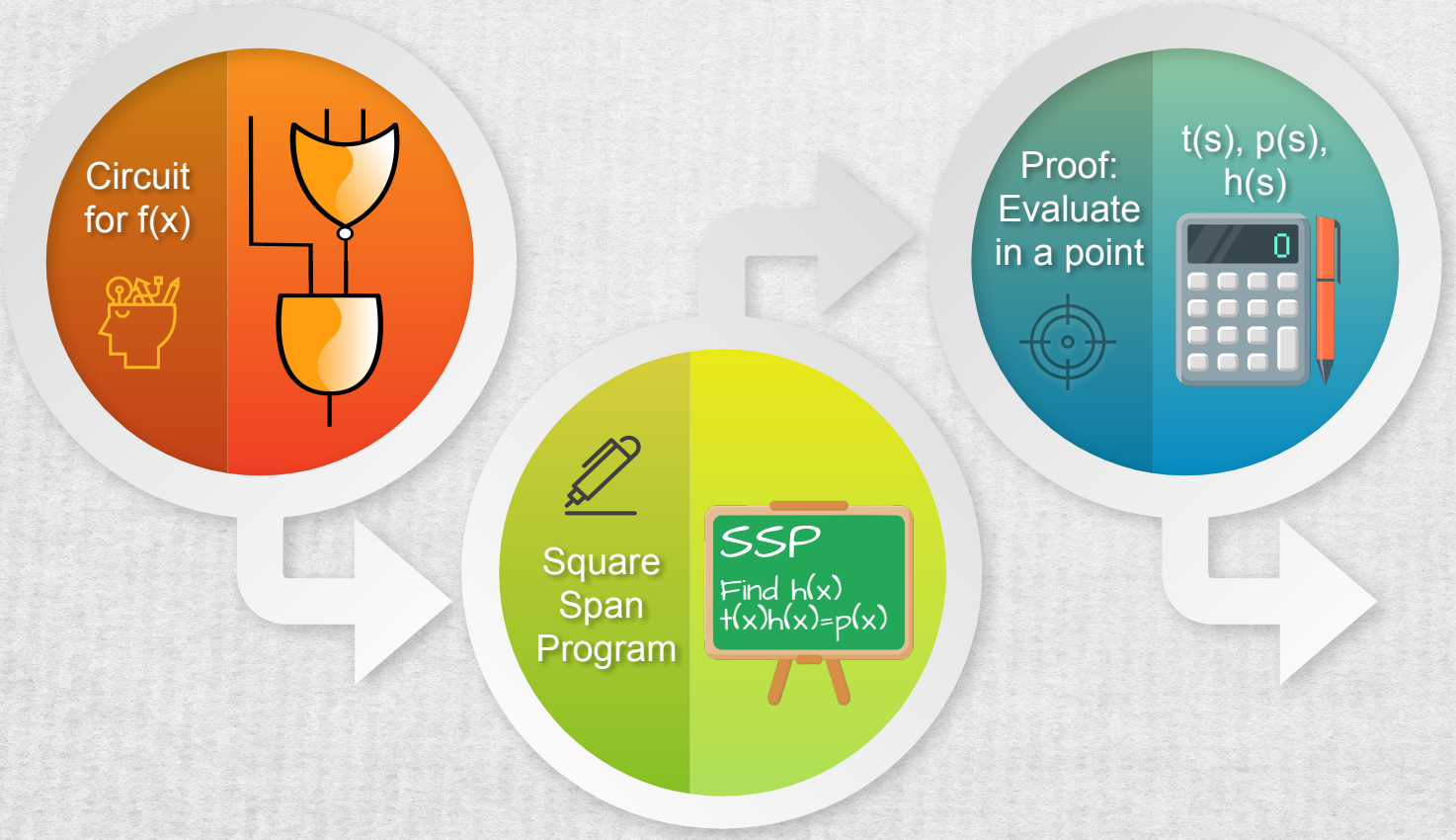
SNARK: overview of Toolchain



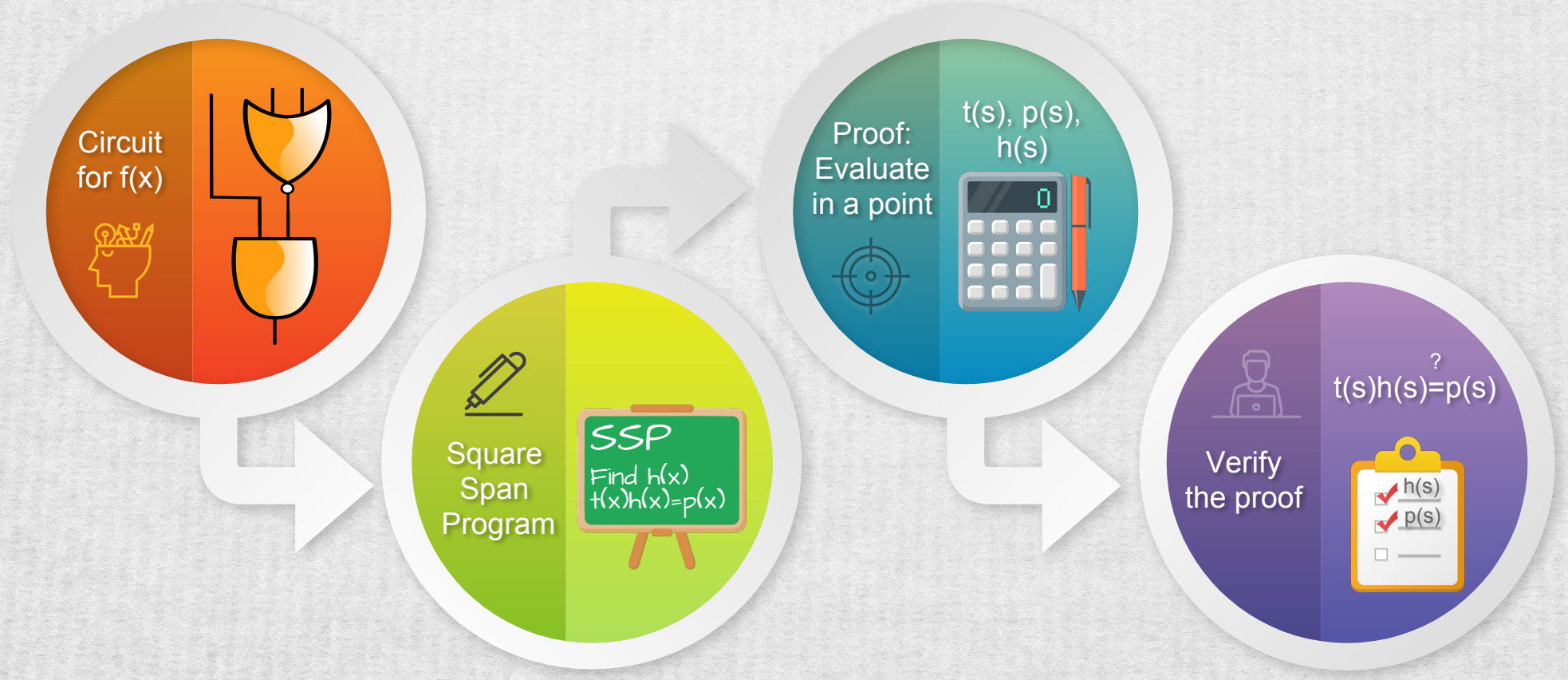
SNARK: overview of Toolchain



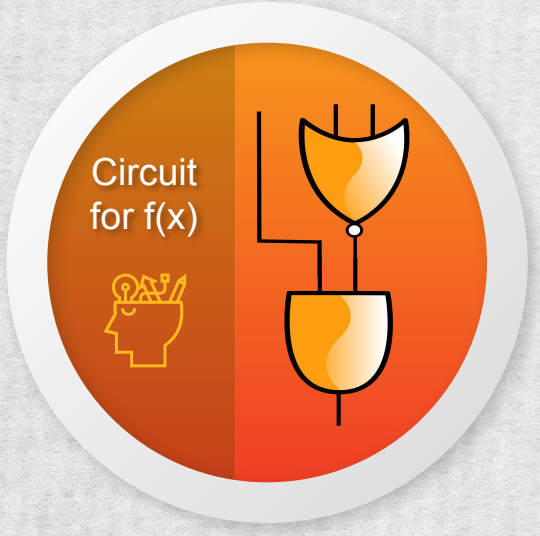
SNARK: overview of Toolchain



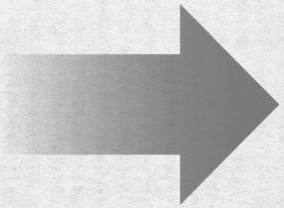
SNARK: overview of Toolchain

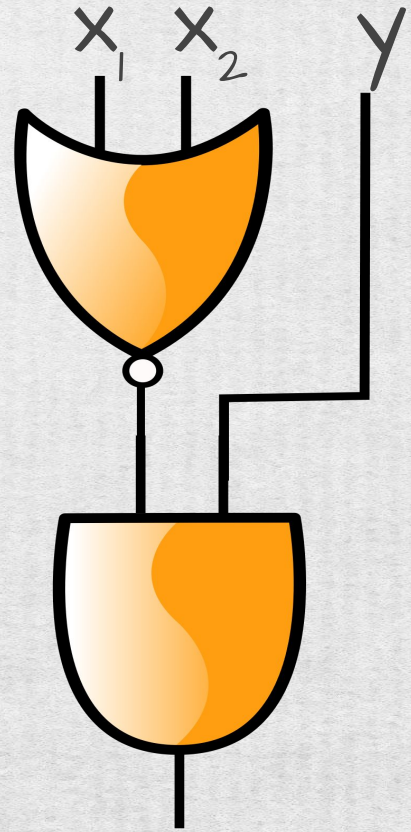


Circuit Satisfiability Problem



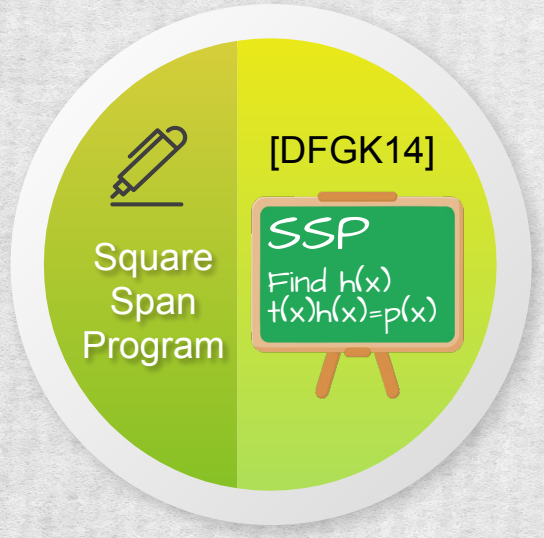
$f(x_1, x_2) = y$

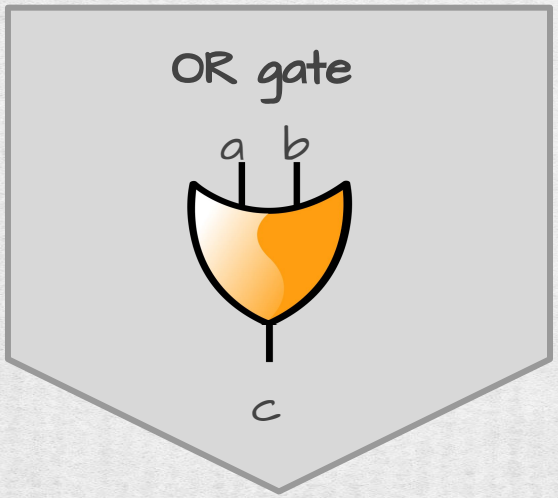


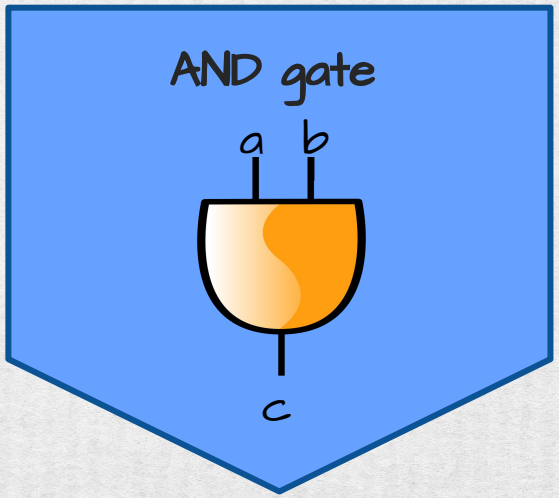


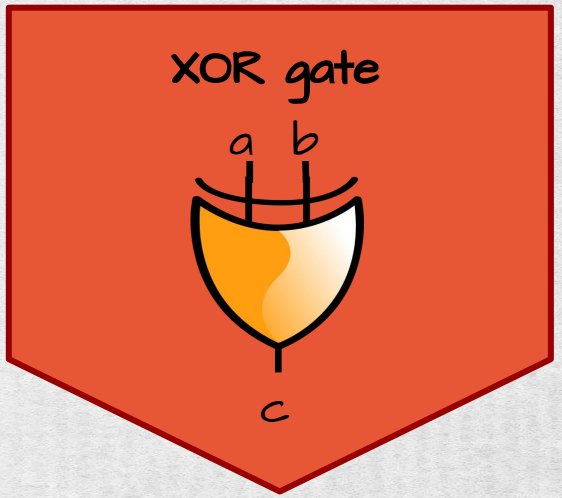
$C(x_1, x_2, y)$

Step 1. Linearization of logic gates





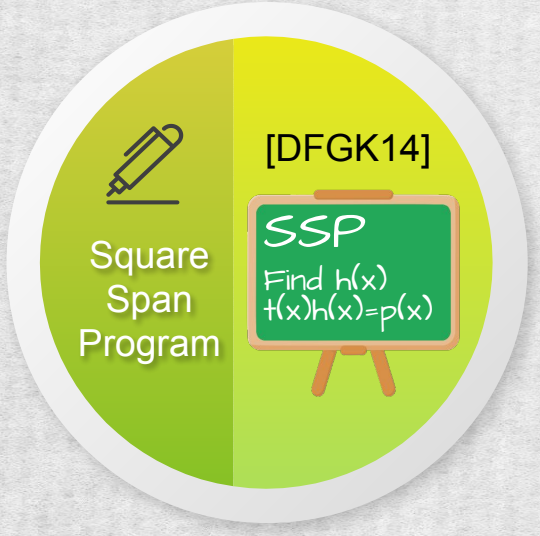




a b c	a b c	a b c
0 0 0	0 0 0	0 0 0
0 1 1	0 1 0	0 1 1
1 0 1	1 0 0	1 0 1
1 1 1	1 1 1	1 1 0
$-a-b+2c \in \{0,1\}$	$a+b-2c \in \{0,1\}$	$a+b+c \in \{0,2\}$

Step 2. Matrix equation for circuit

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

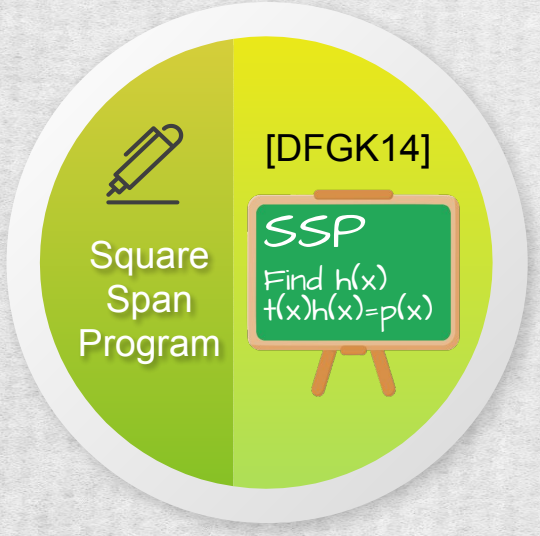


$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$

$$V \begin{bmatrix} a \\ \delta \end{bmatrix} \in \{0,2\}^d$$

$$\left(V \begin{bmatrix} a \\ \delta \\ 1 \end{bmatrix} \right) \circ \left(V \begin{bmatrix} a \\ \delta \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 \end{bmatrix}$$

Square Span Program



SSP

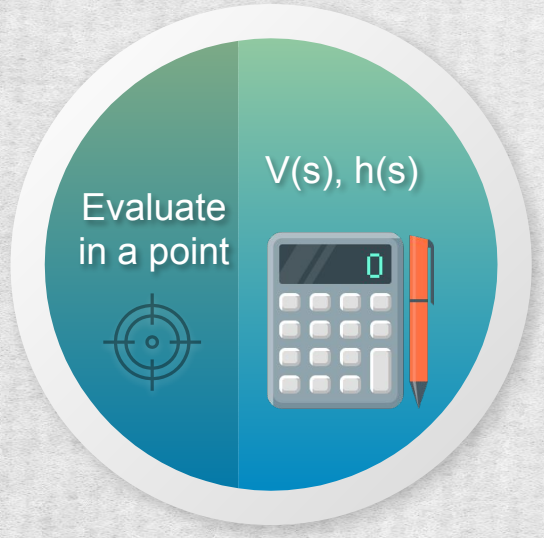
For $\{v_i(x)\}_{i=1,m}$, $t(x) \in \mathbb{F}[x]$

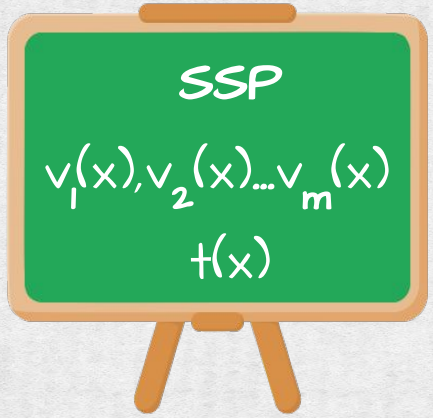
find $V(x), h(x)$ such that

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

$$t(x)h(x) = V(x)^2 - 1$$

Proving on top of SSP: Idea





$$V(s), h(s) = ?$$

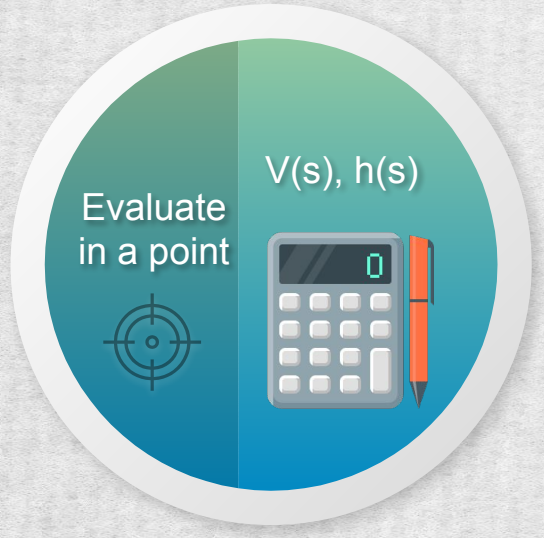
$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

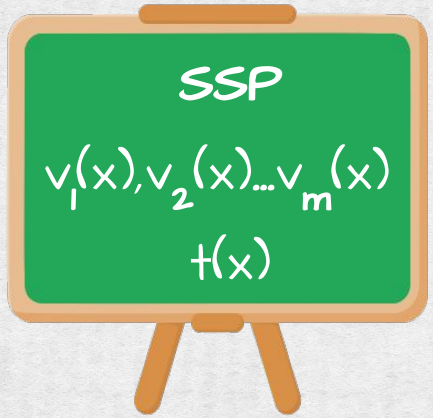
$$t(s)h(s) = V(s)^2 - 1$$

Prover: Evaluate the solution in a random unknown point **s**

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

Proving on top of SSP: Idea

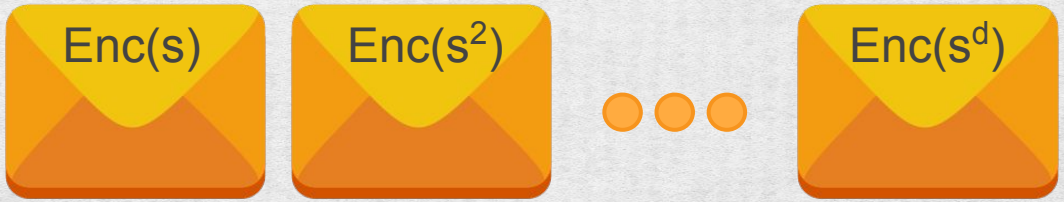




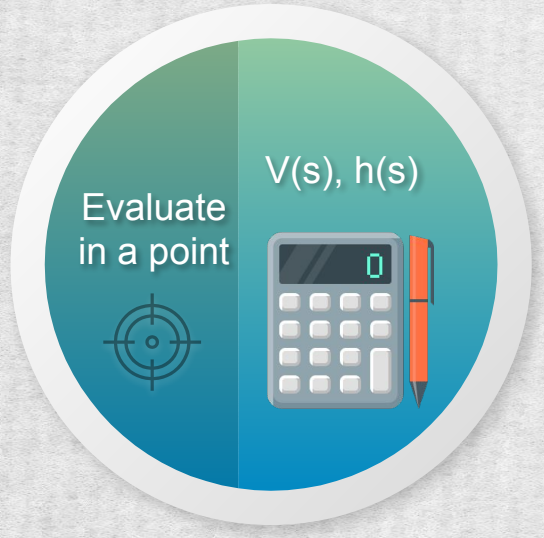
$$V(s), h(s) = ?$$

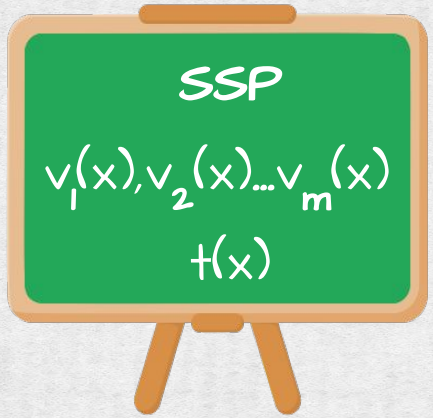
$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

$$t(s)h(s) = V(s)^2 - 1$$



Proving on top of SSP: Idea

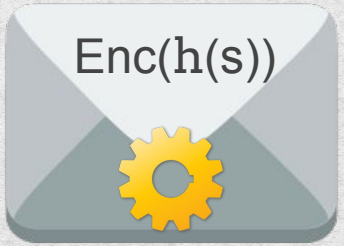




$$V(s), h(s) = ?$$

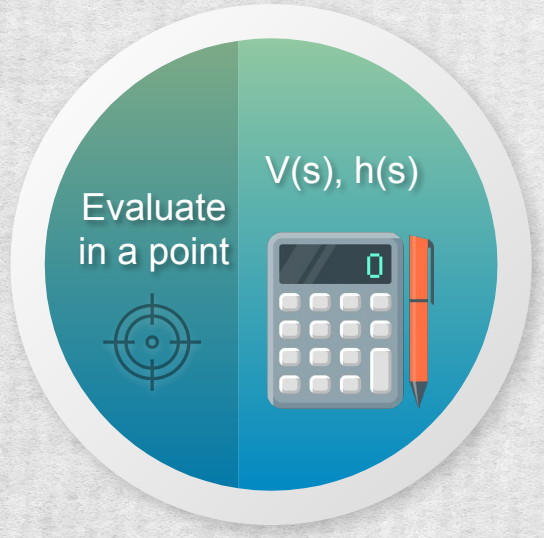
$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

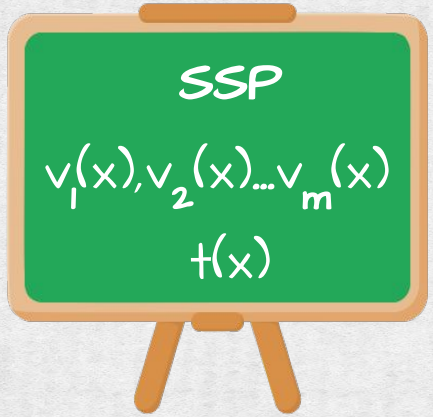
$$t(s)h(s) = V(s)^2 - 1$$



$= \mathrm{Enc}(\Sigma\, h_i s^i)$

Proving on top of SSP: Idea





$$V(s), h(s) = ?$$

$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

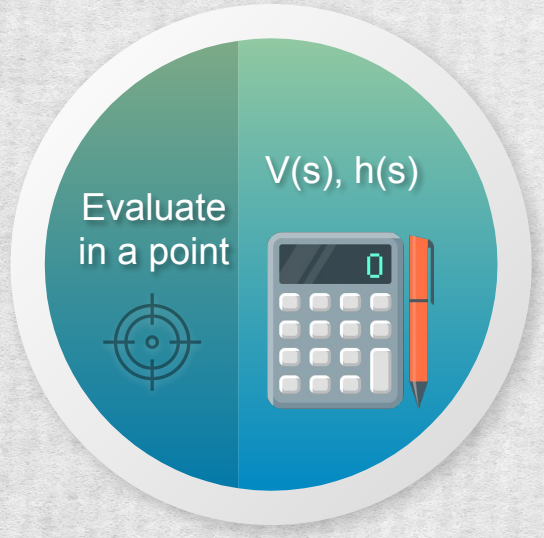
$$t(s)h(s) = V(s)^2 - 1$$

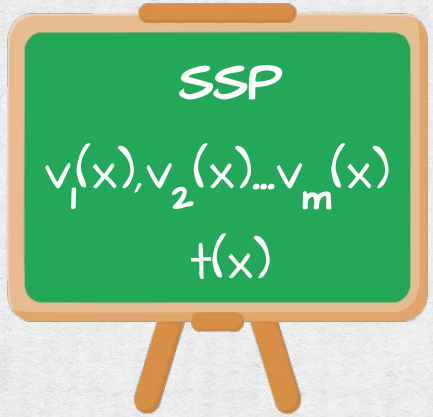
Encoding:

- ***linearly*** homomorphic

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

Proving on top of SSP: Idea

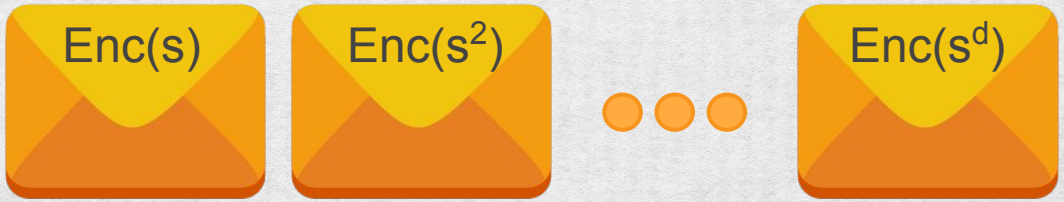




$$V(s), h(s) = ?$$

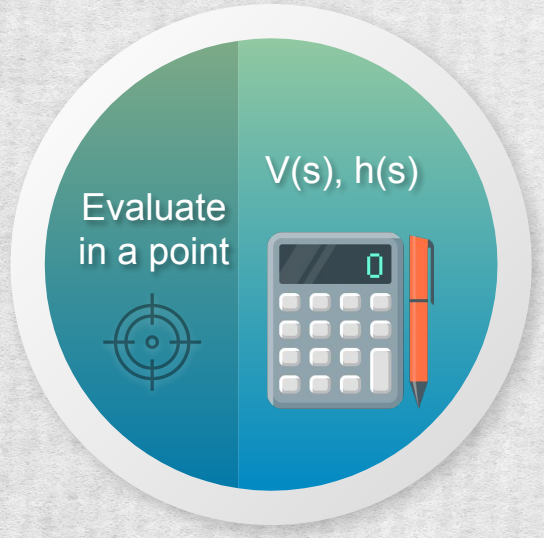
$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

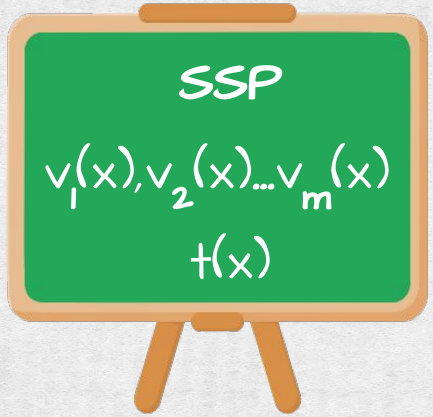
$$t(s)h(s) = V(s)^2 - 1$$



Proof  = 

Not an argument of Knowledge!

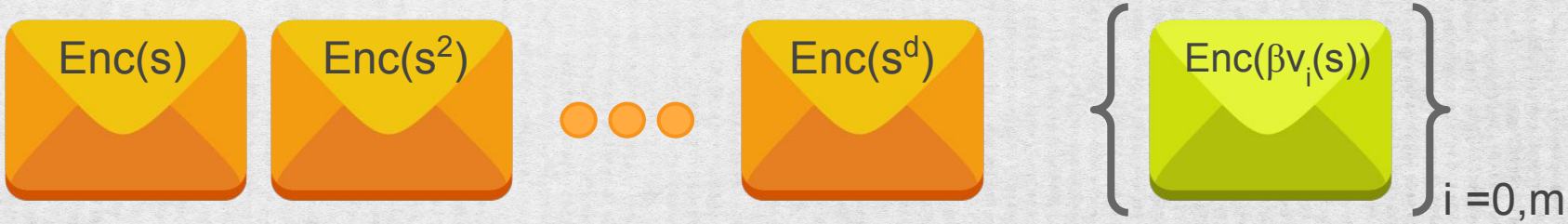


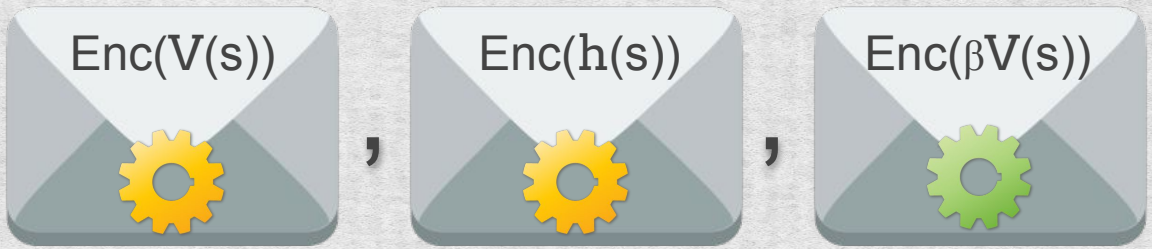


$$V(s), h(s) = ?$$

$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

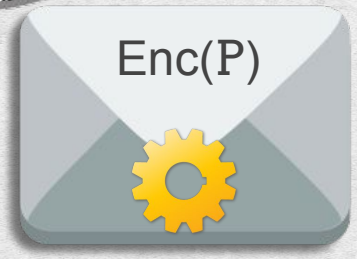
$$t(s)h(s) = V(s)^2 - 1$$

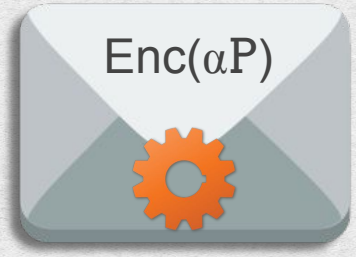


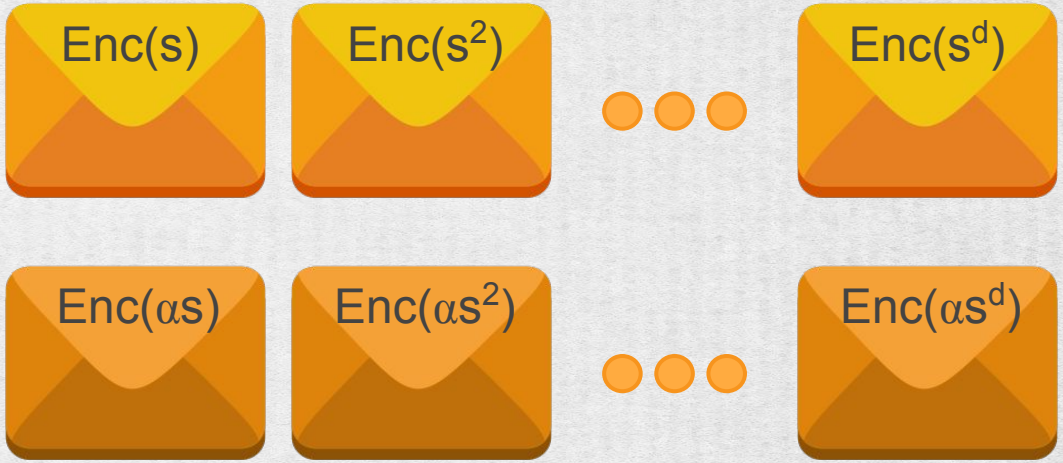
Proof  = 

Assumption PKE: Power Knowledge of Exponent

d-PKE

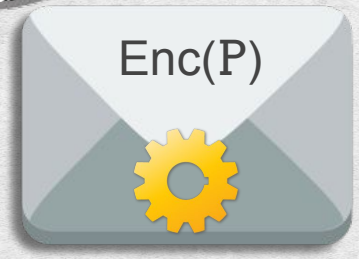


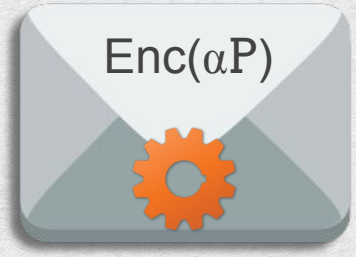


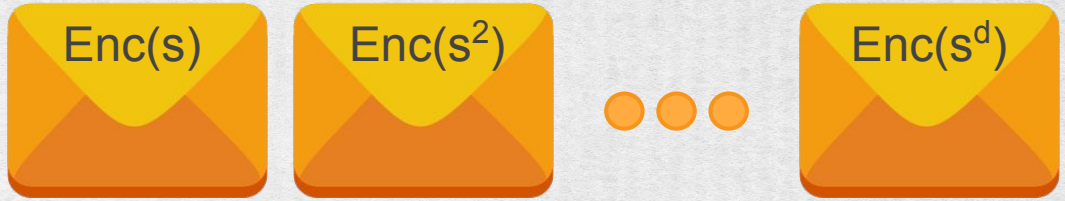


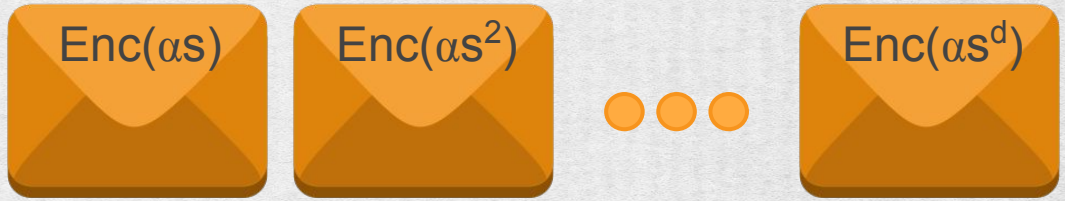
Assumption PKE: Power Knowledge of Exponent

d-PKE

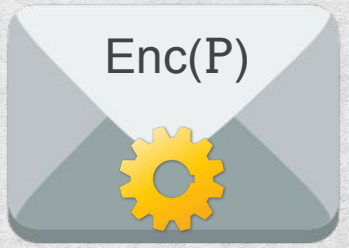






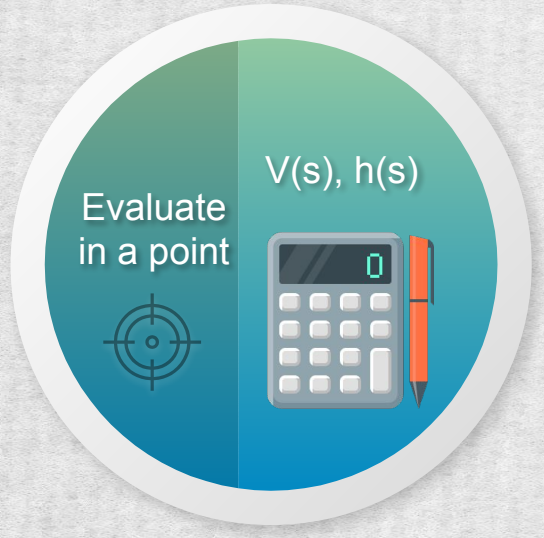


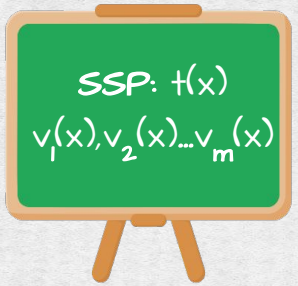
$\mathcal{E}$



$= \mathrm{Enc}(\Sigma\, p_i s^i)$

Setup and Proof





$\{ Enc(\beta v_i(s)) \}$ i =1,m

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

Enc(αs) Enc(αs^2) ... Enc(αs^d)

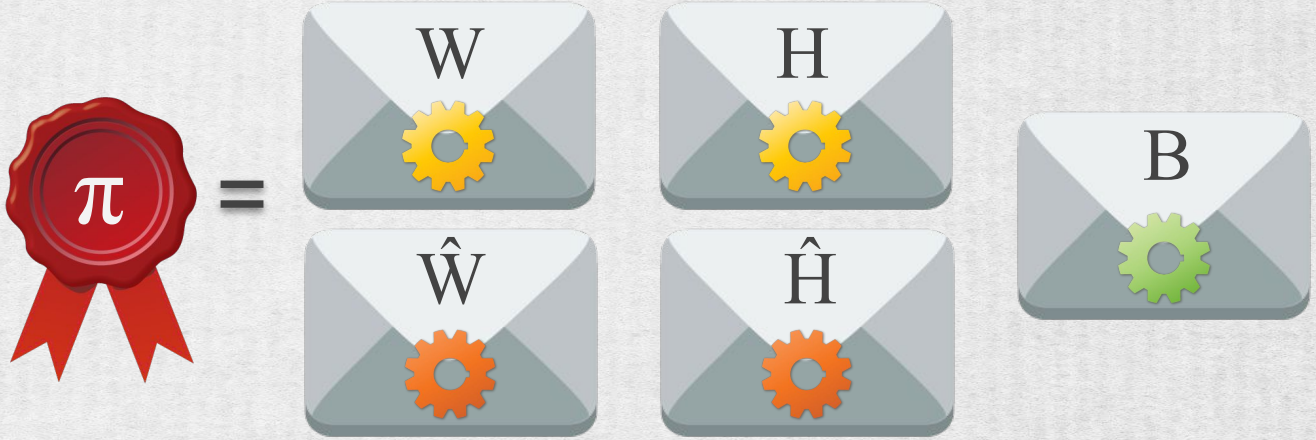
π =

Enc(V(s)) Enc(h(s))

Enc(αV(s)) Enc(αh(s))

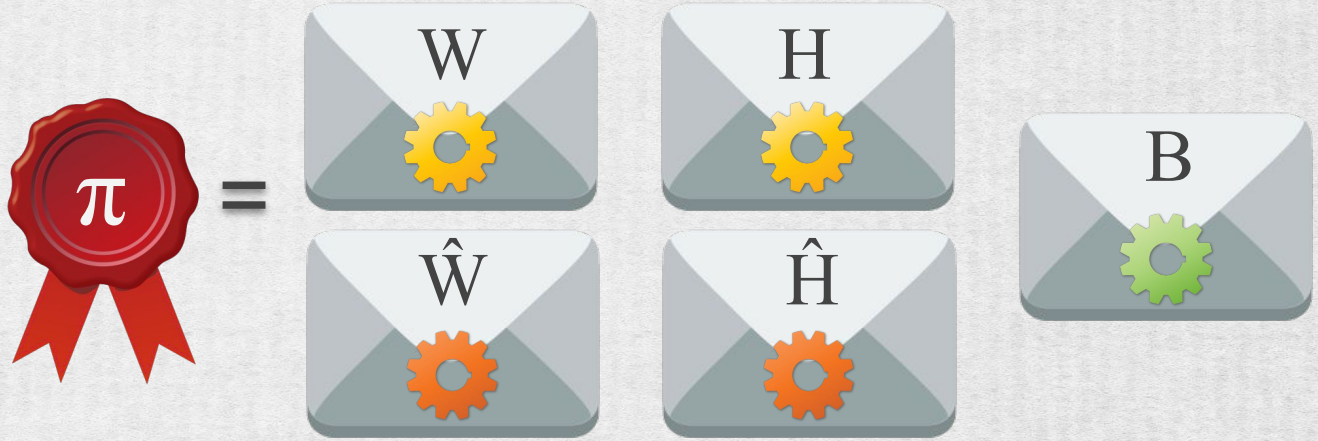
Enc(βV(s))

Proving on top of SSP: Verifier



Verifier

Proving on top of SSP: Verifier



Verifier ?

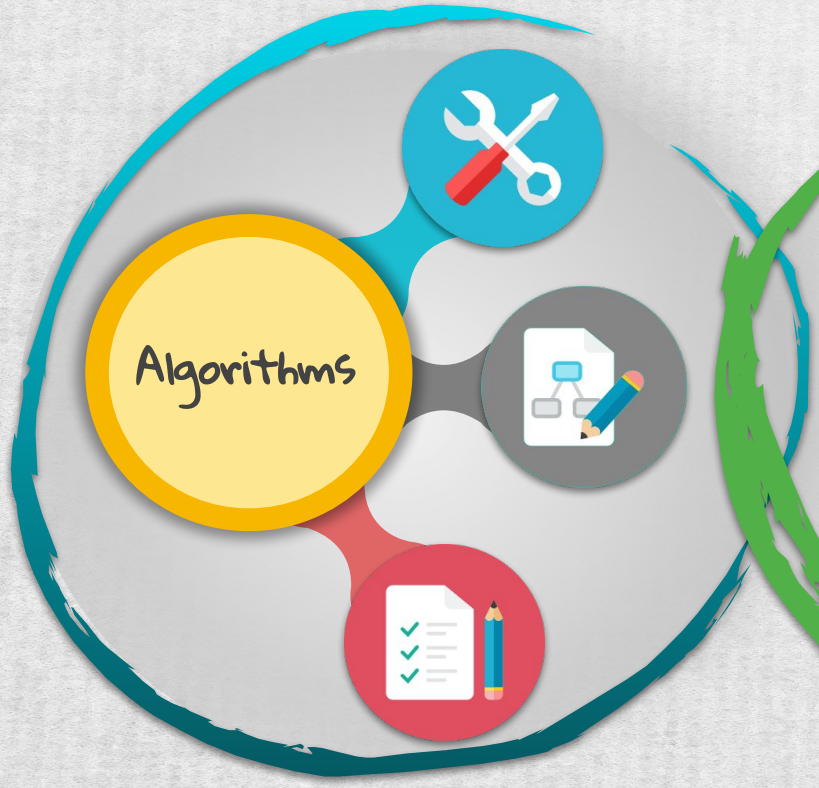
Proving on top of SSP: Verifier

Encoding:

- *linearly* homomorphic
- quadratic root detection
- image verification

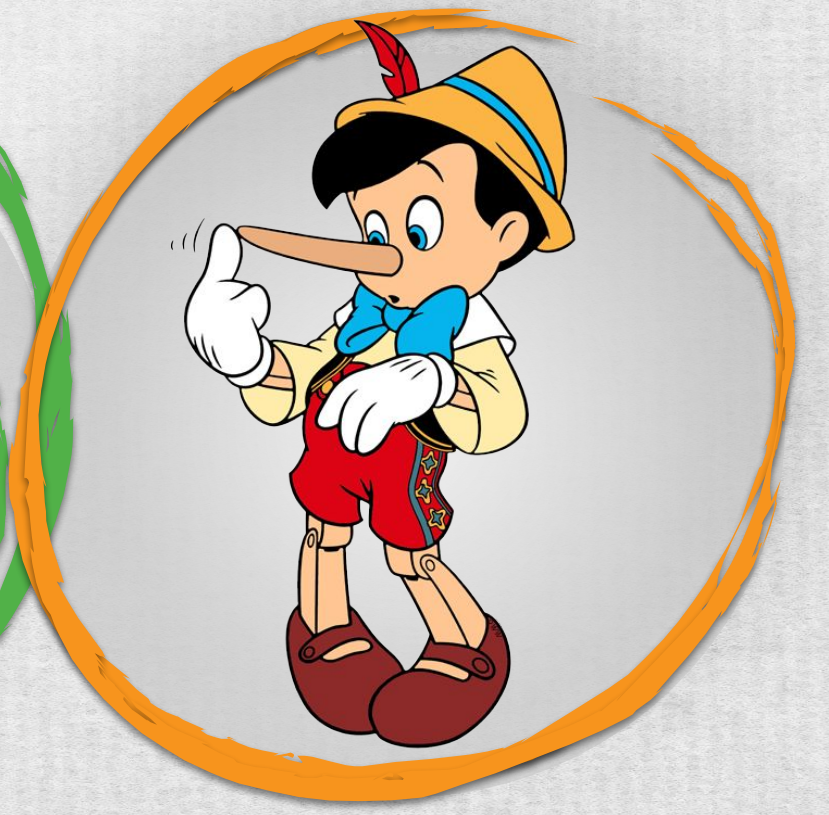
Formal Construction and Security

Protocol

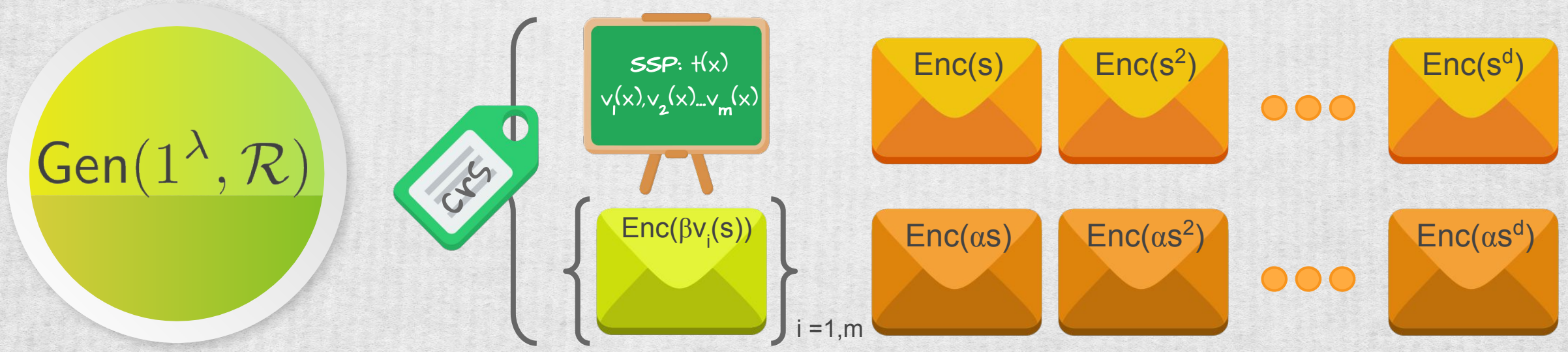


Assumptions

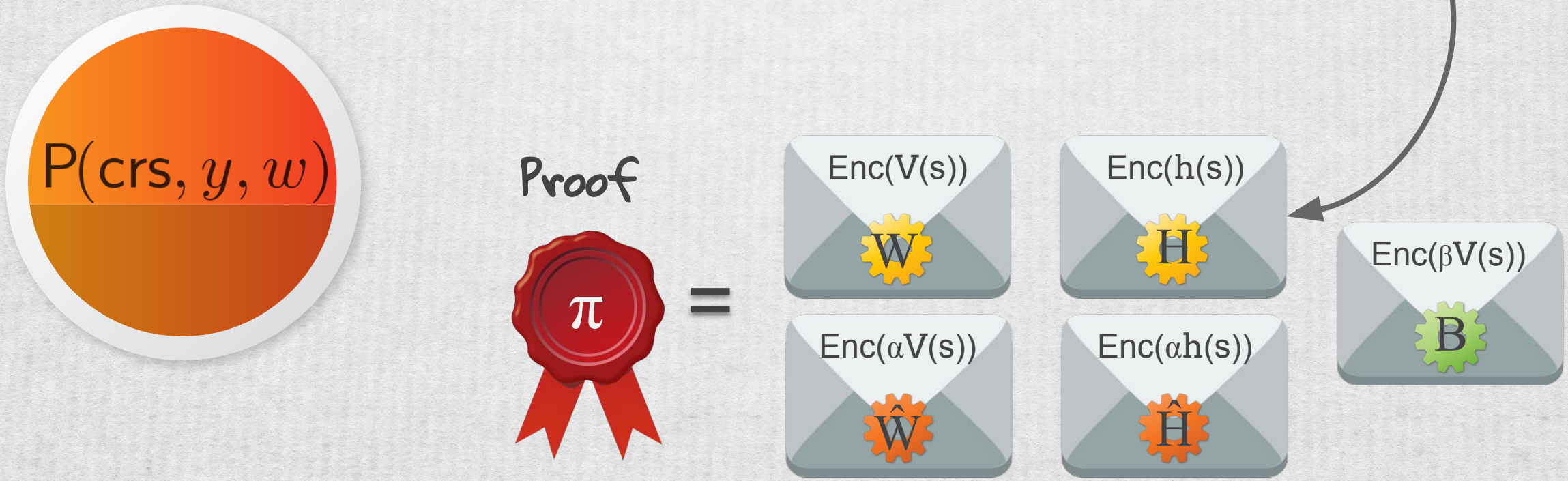
Security



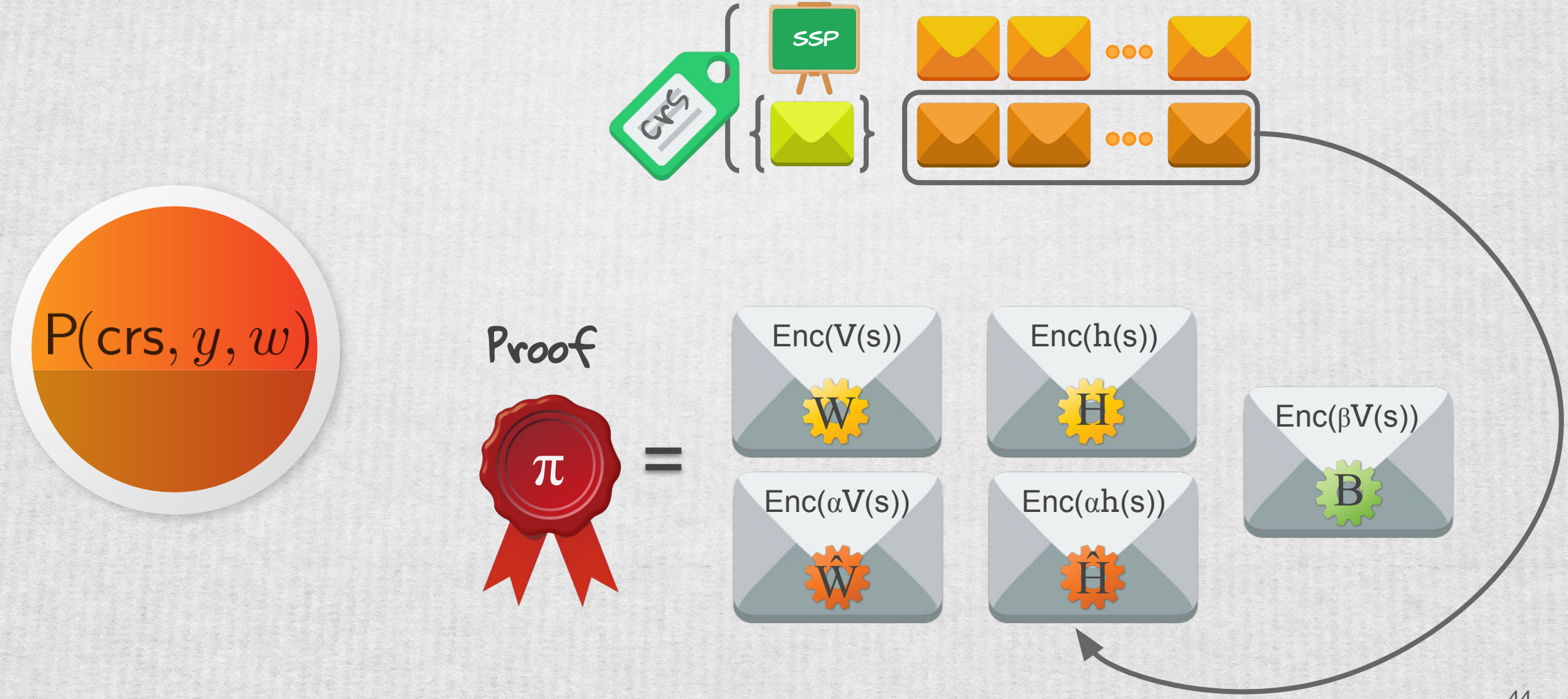
Setup Algorithm



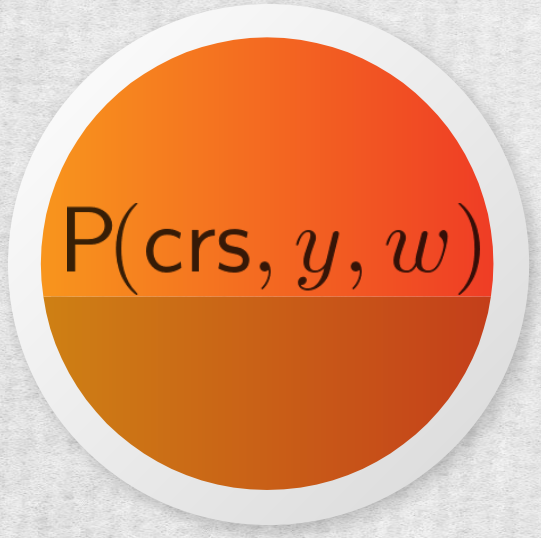
Prover Algorithm

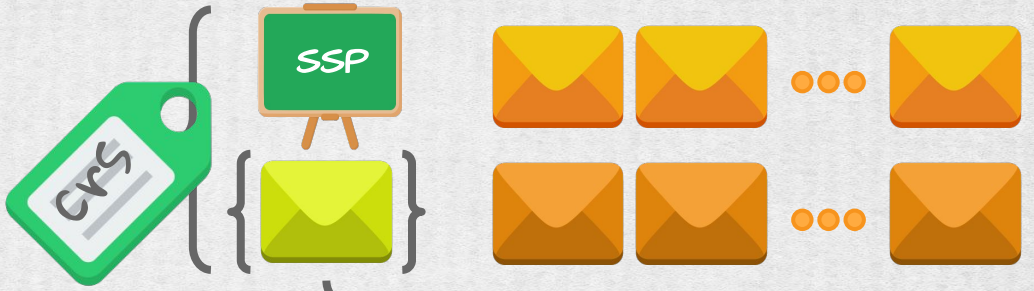


Prover Algorithm

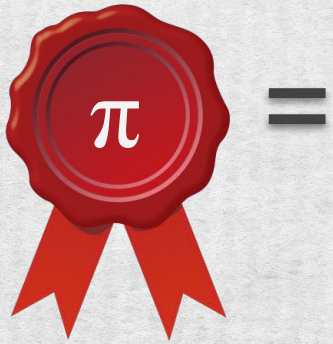


Prover Algorithm

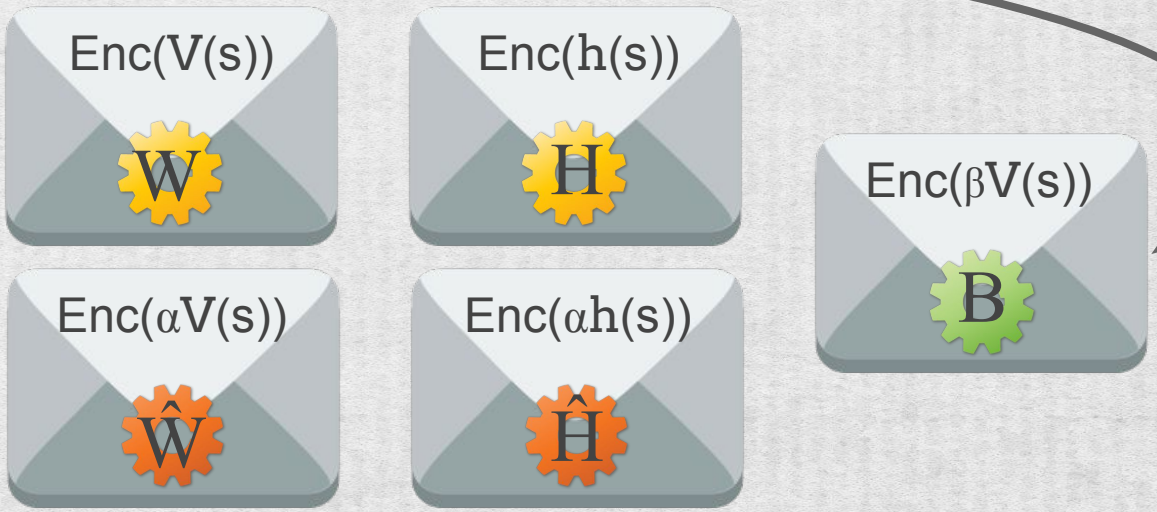




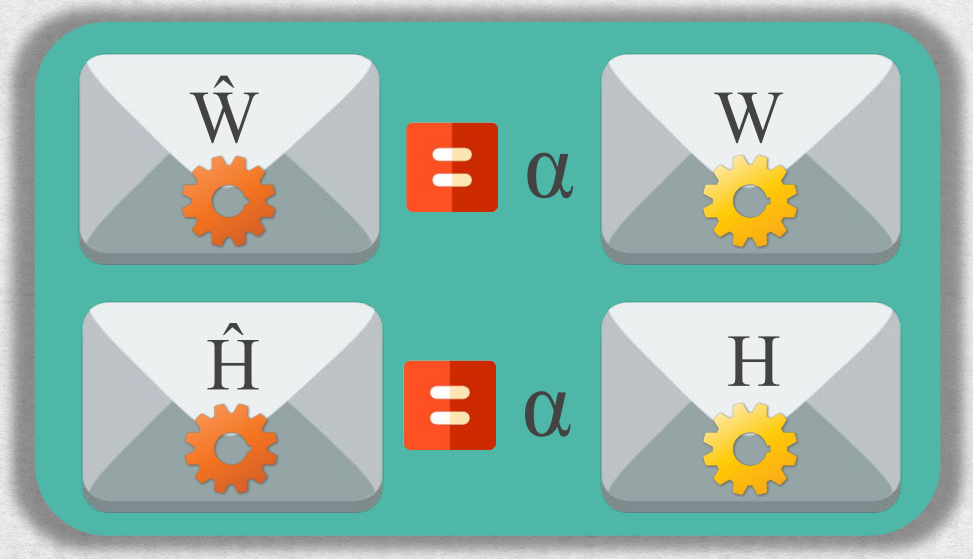
Proof

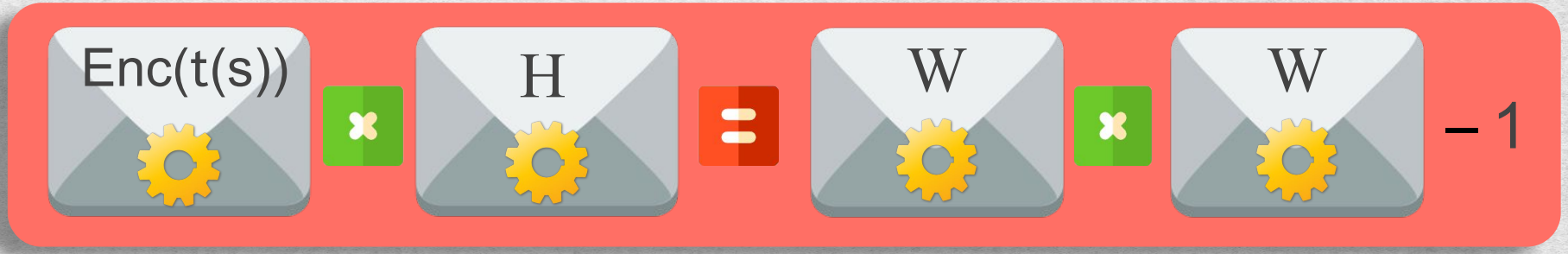


=



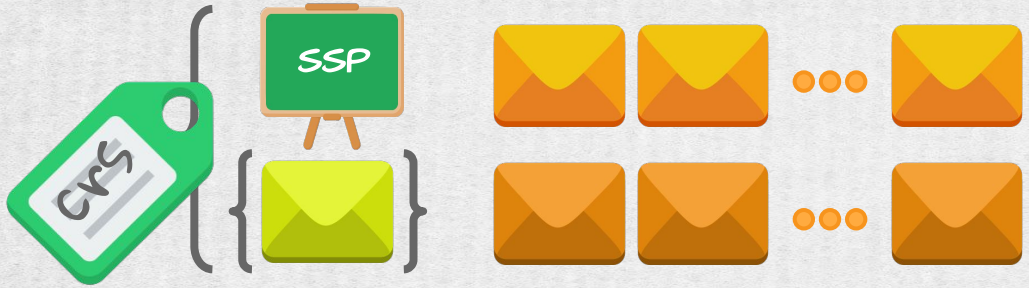
Verifier Algorithm



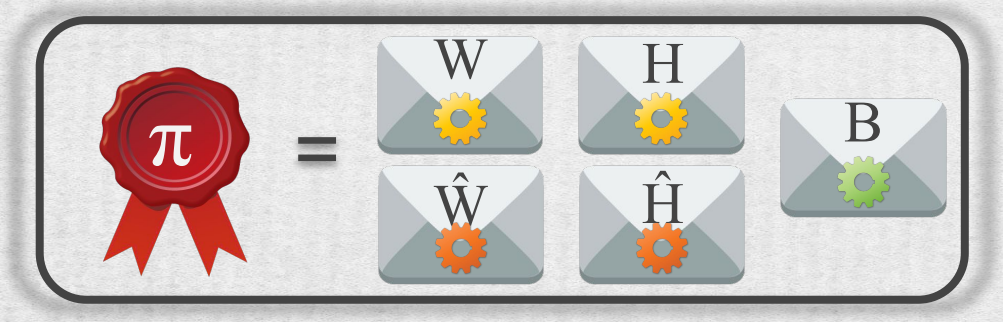


Security Reduction

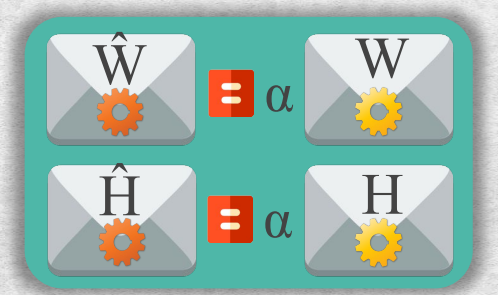
$\mathsf{Gen}(1^\lambda, \mathcal{R})$



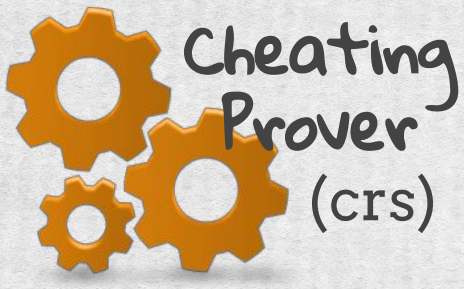
$\mathsf{P}(\mathsf{crs}, y, w)$

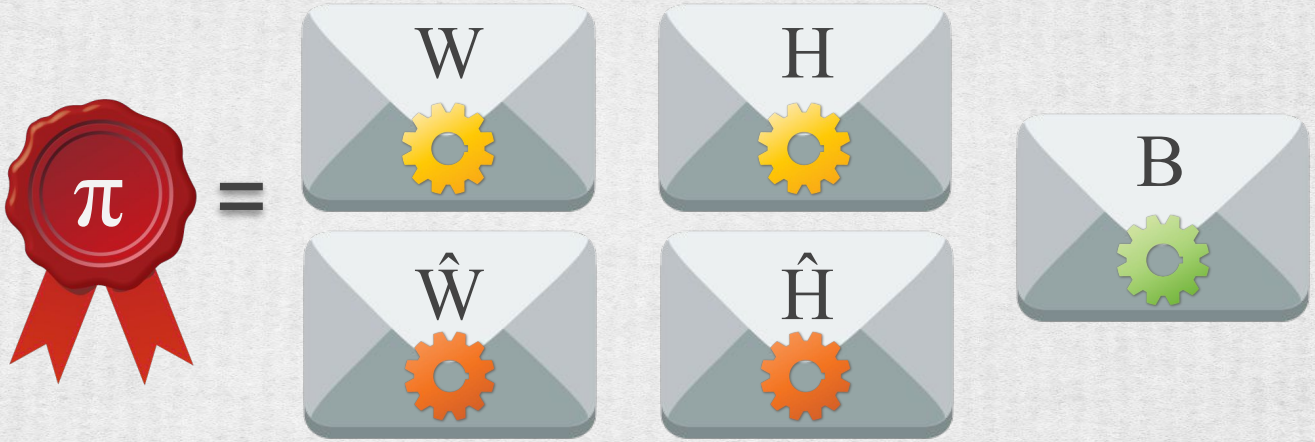


$\mathsf{V}(\mathsf{vk}, y, \pi)$

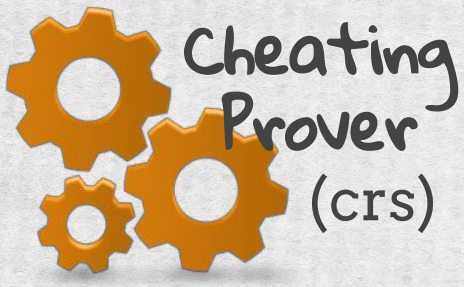


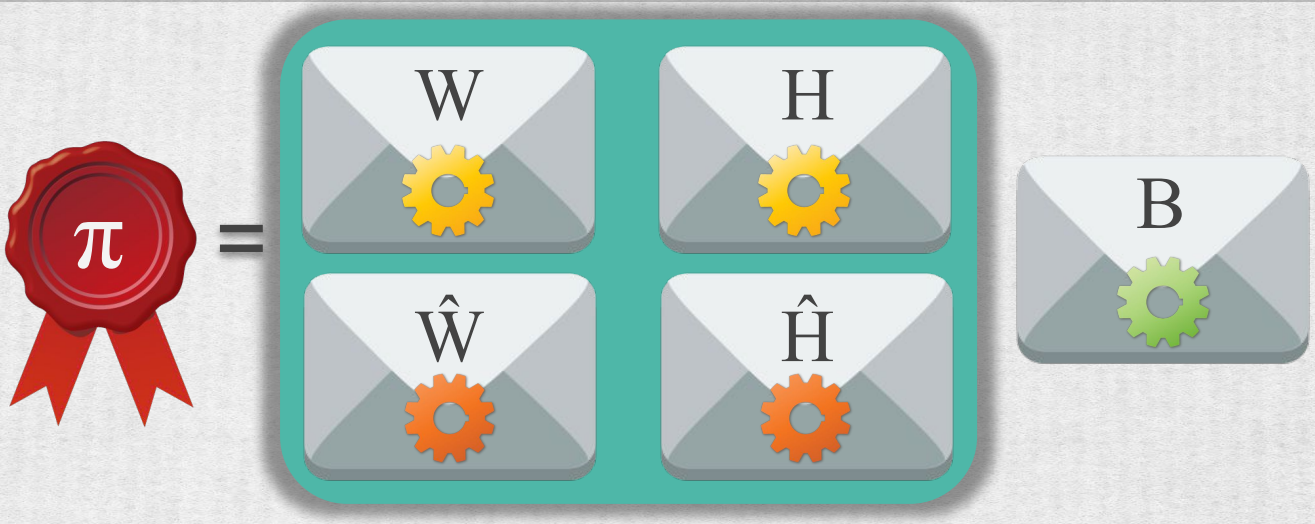
Cheating Strategy



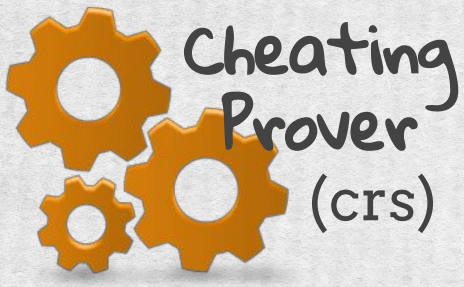


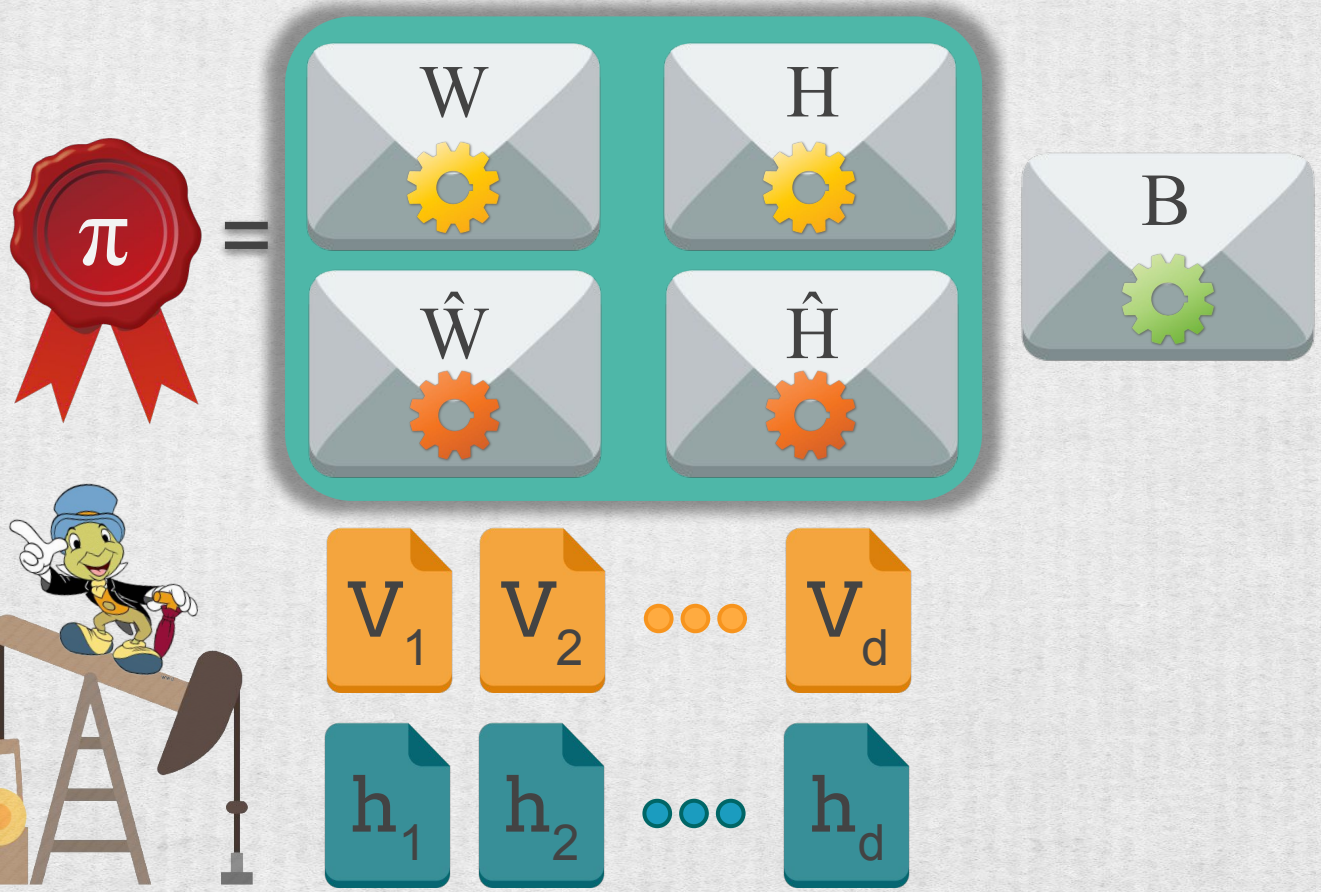
Cheating Strategy



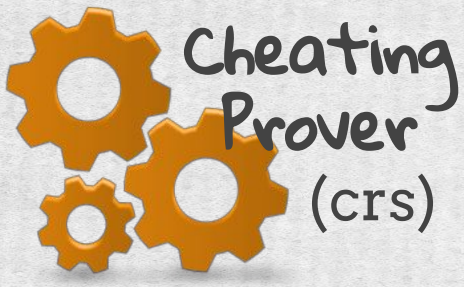


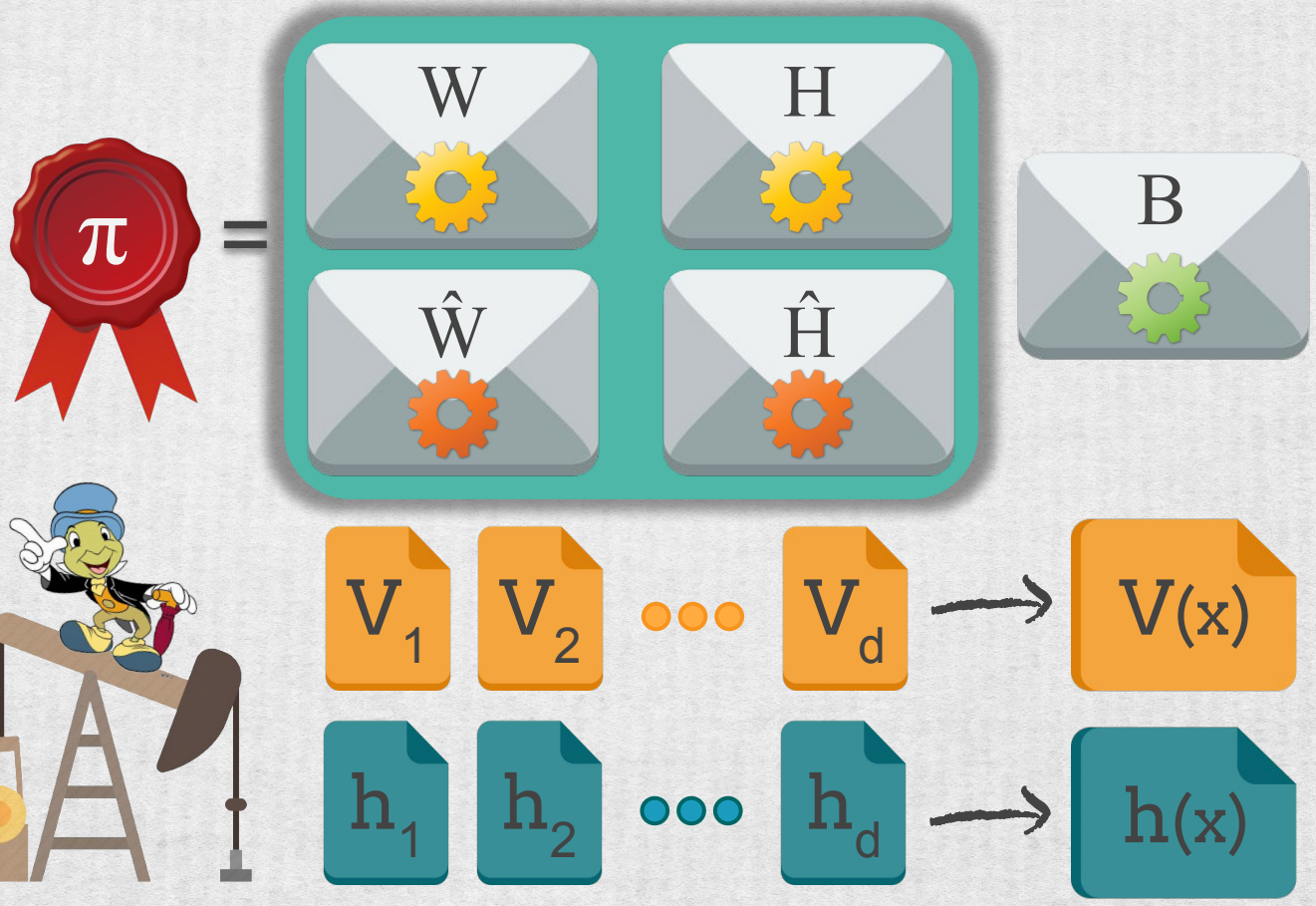
Cheating Strategy



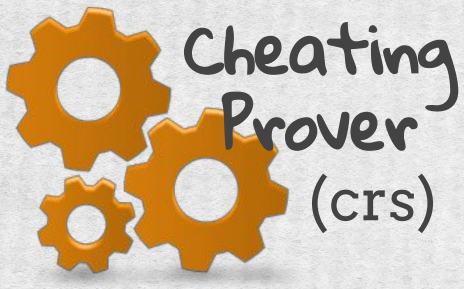


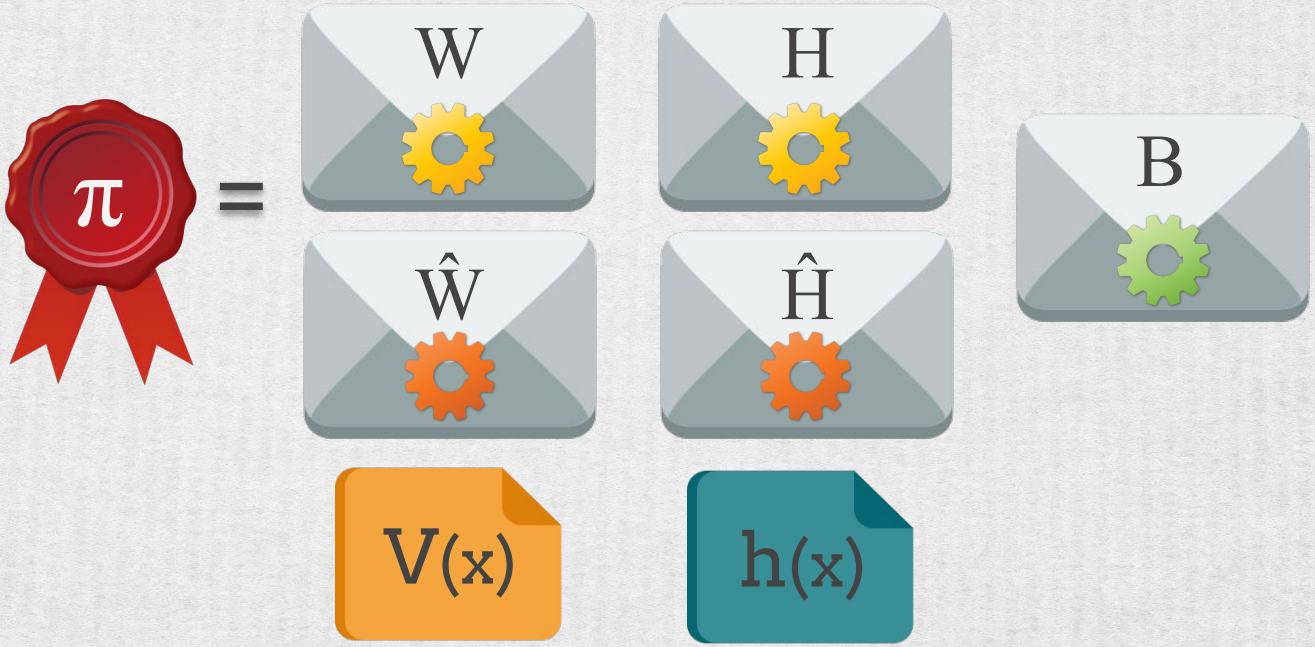
Cheating Strategy



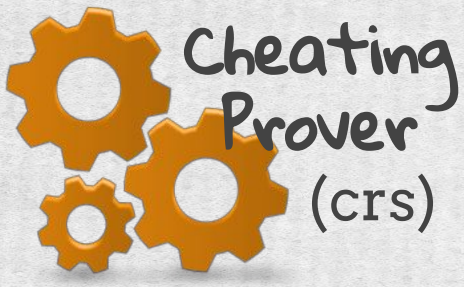


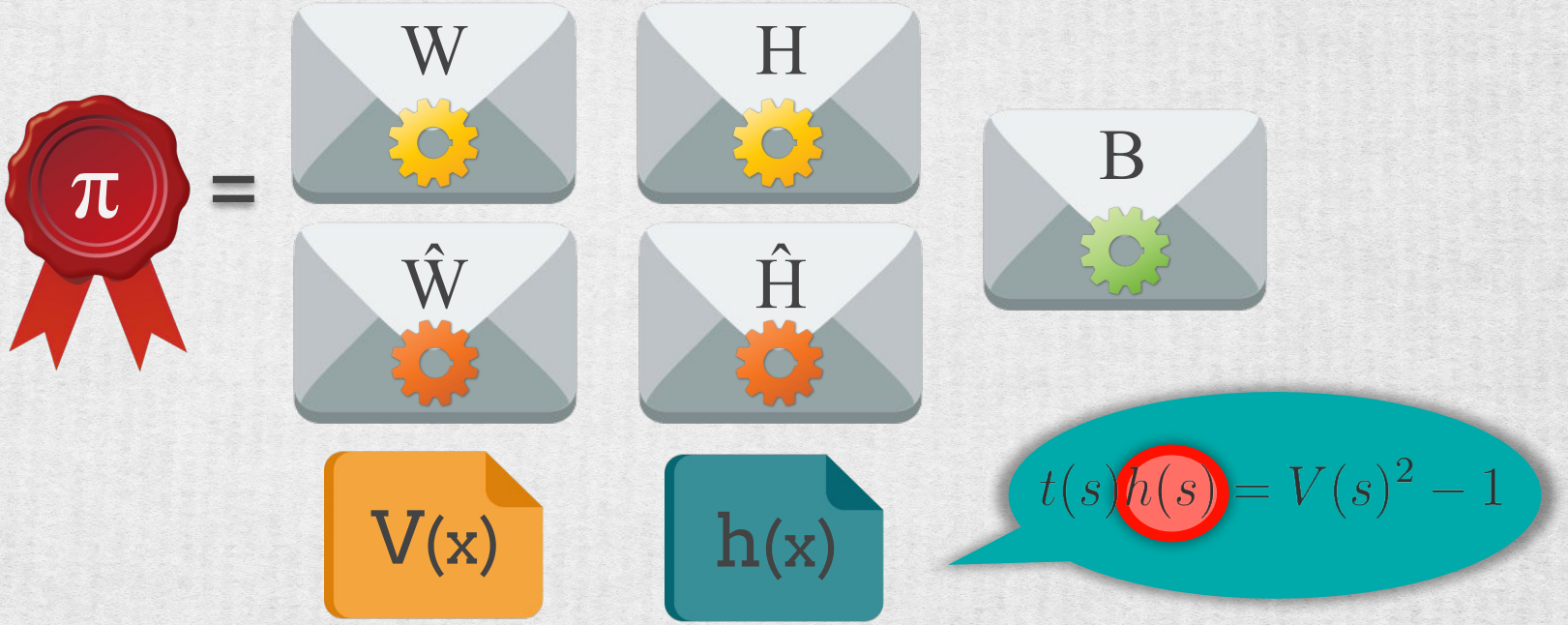
Cheating Strategy





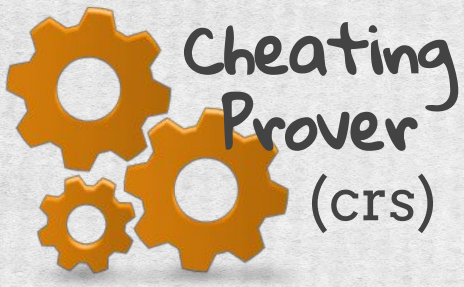
Division does not hold





- $t(x)h(x) \neq V^2(x) - 1$, but

Invalid linear combination



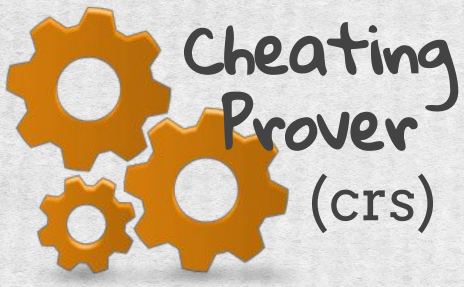
$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

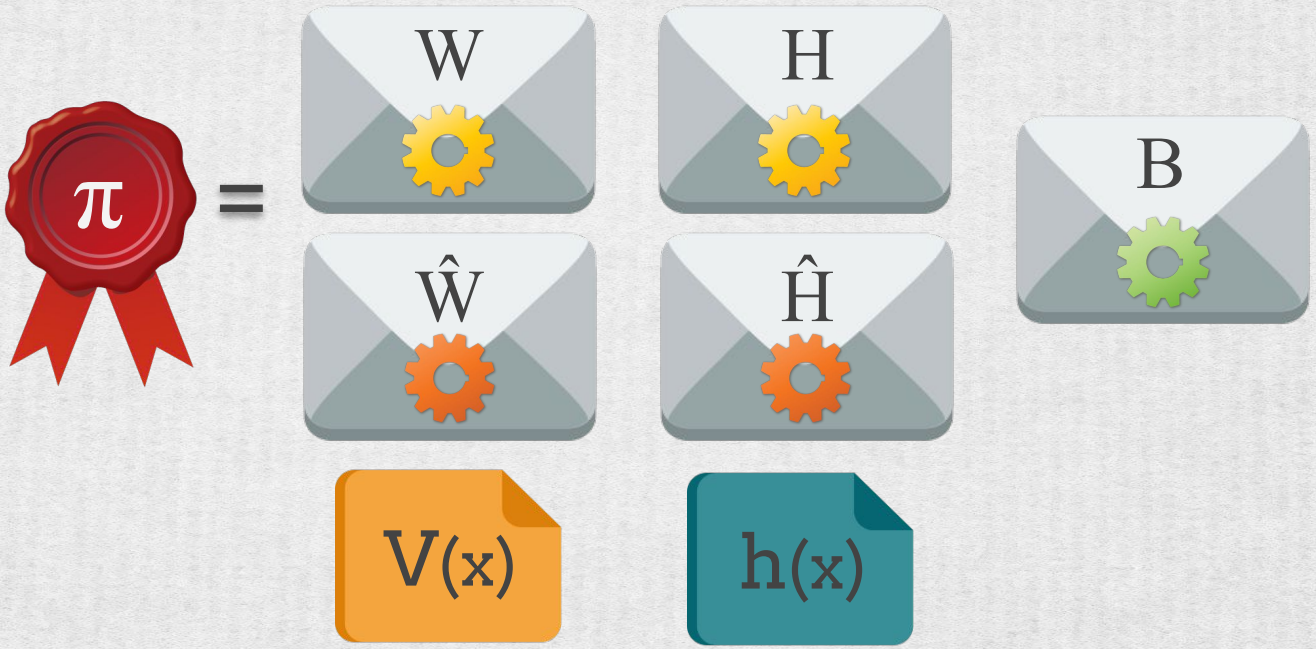
V(x)

h(x)

- $\mathrm{t(x)h(x)} \neq V^2(x) - 1$, but $\mathsf{Enc}(t(s))H = W^2 - 1$
- $\mathrm{V(x)} \notin \mathsf{Span}(v_1, \ldots v_m)$, but

Prover unable to compute higher degree polynomials

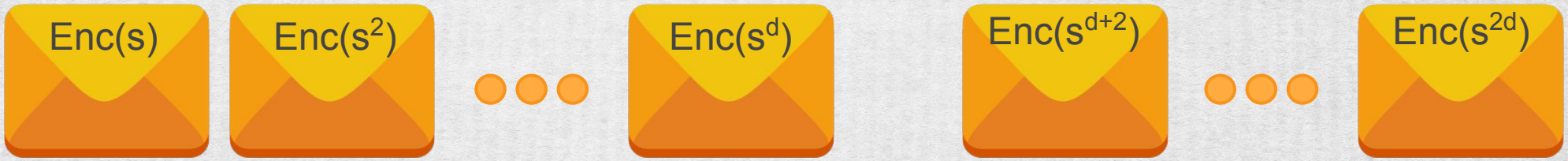




- $\mathrm{t(x)h(x)} \neq V^2(x) - 1$, but $\mathsf{Enc}(t(s))H = W^2 - 1$
- $\mathrm{V(x)} \notin \mathsf{Span}(v_1, \ldots v_m)$, but $B = \mathsf{Enc}(\beta V(s))$

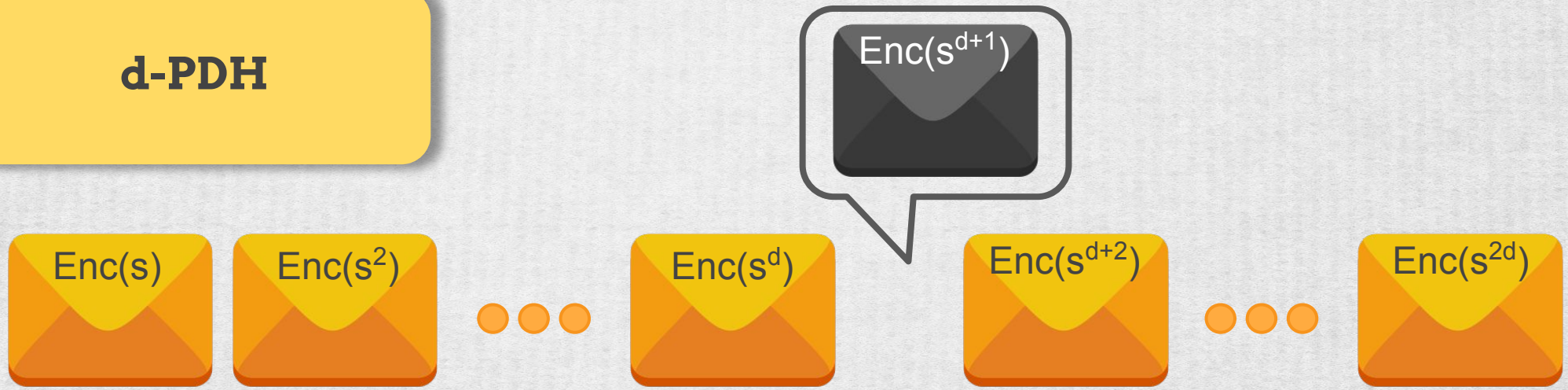
Assumption PDH: Power Diffie-Hellman

d-PDH



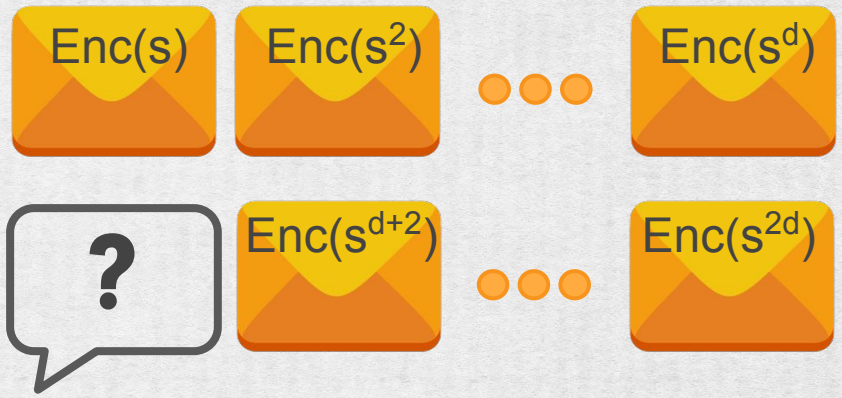
Assumption PDH: Power Diffie-Hellman

d-PDH



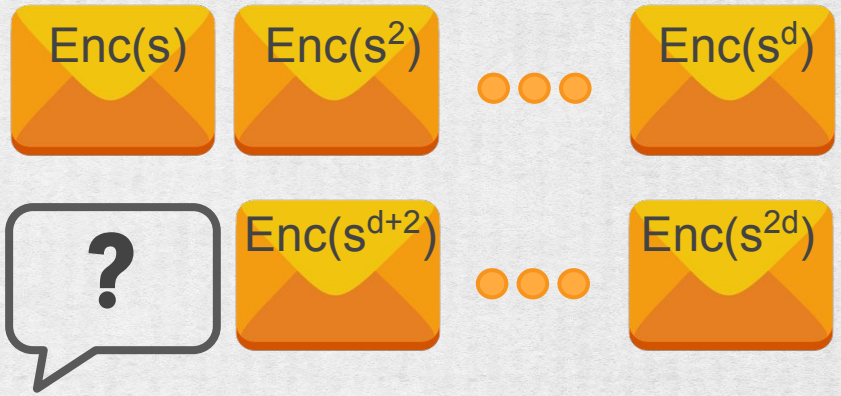
Security Reduction: Cheating Prover to d-PDH

d-PDH



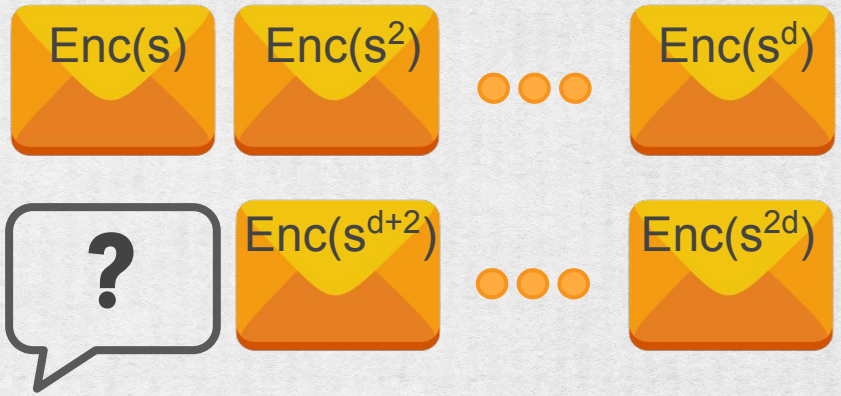
Security Reduction: Cheating Prover to d-PDH

d-PDH



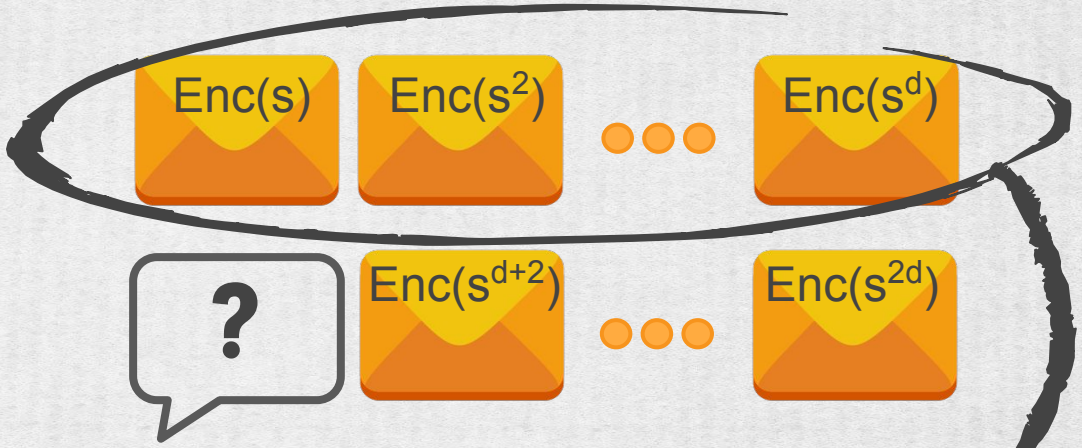
Security Reduction: Cheating Prover to d-PDH

d-PDH

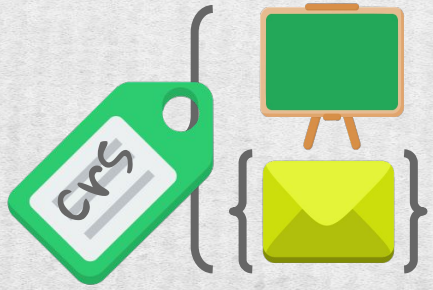


Security Reduction: Cheating Prover to d-PDH

d-PDH

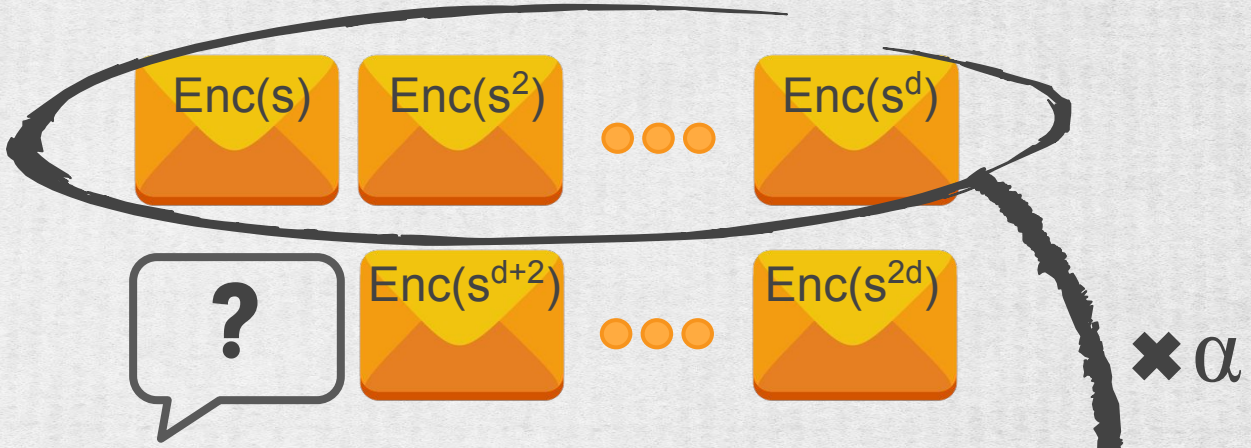


Cheating Prover

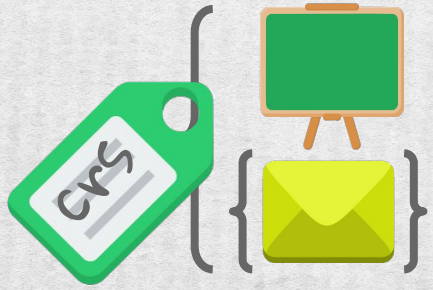


Security Reduction: Cheating Prover to d-PDH

d-PDH



Cheating Prover



Security Reduction: Cheating Prover to d-PDH

d-PDH

Enc(s) Enc(s^2) ••• Enc(s^d)

Enc(s^{d+2}) ••• Enc(s^{2d})

$\{v_i(x)\}_{i=1,m}$

$t(x)$

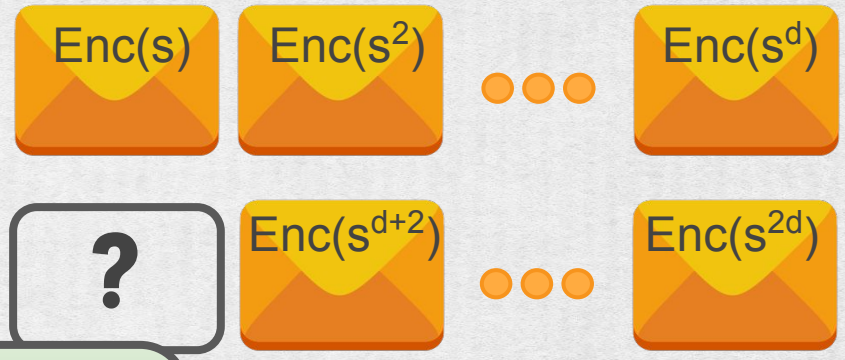
Cheating Prover

s s^2 ••• s^d

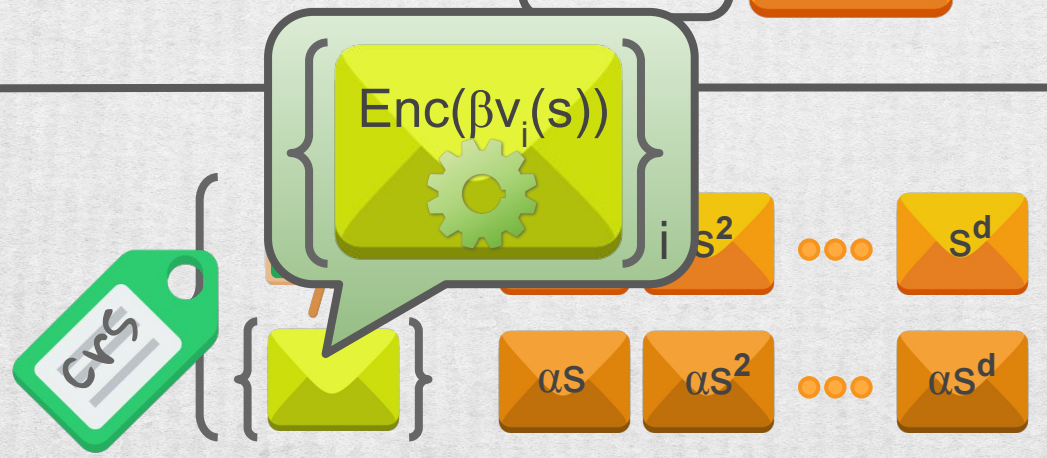
αs $αs^2$ ••• $αs^d$

Security Reduction: Cheating Prover to d-PDH

d-PDH

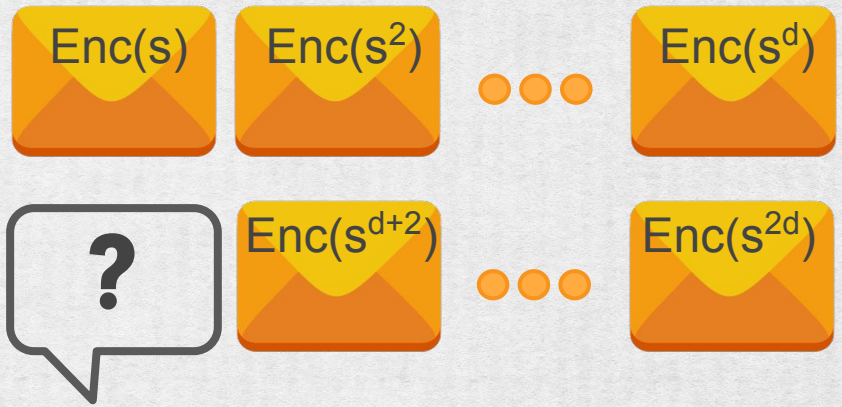


Cheating Prover

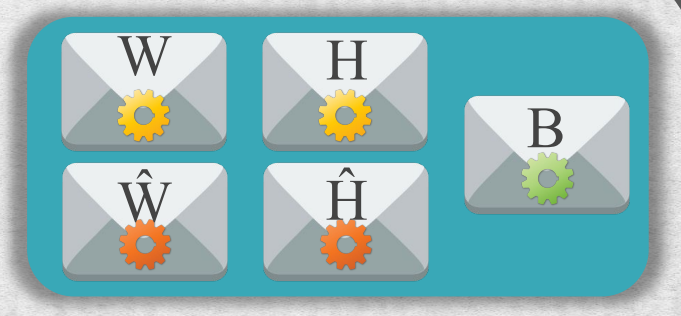


Security Reduction: Cheating Prover to d-PDH

d-PDH

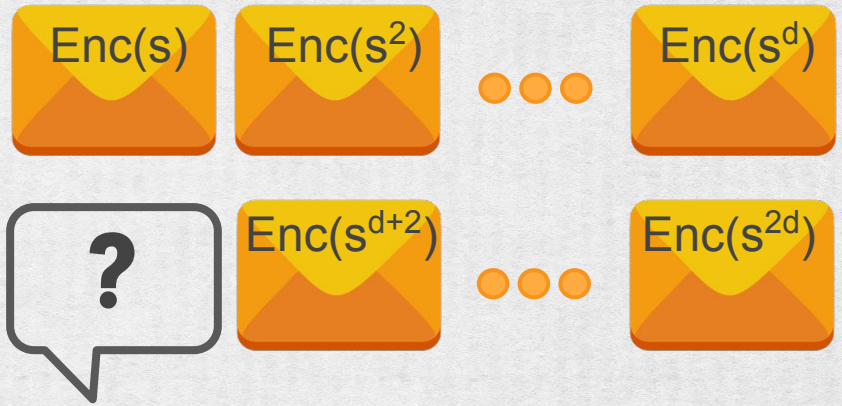


Cheating Prover

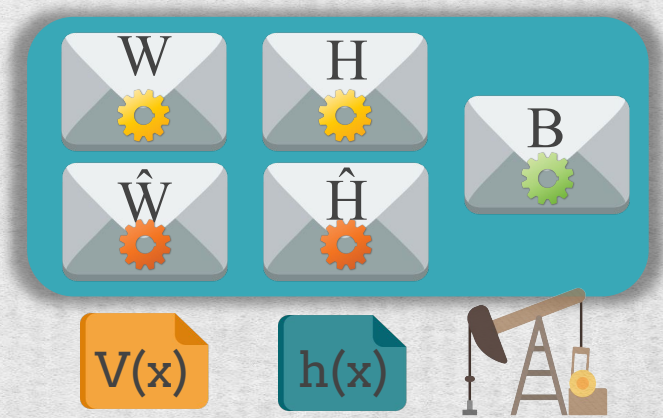


Security Reduction: Cheating Prover to d-PDH

d-PDH

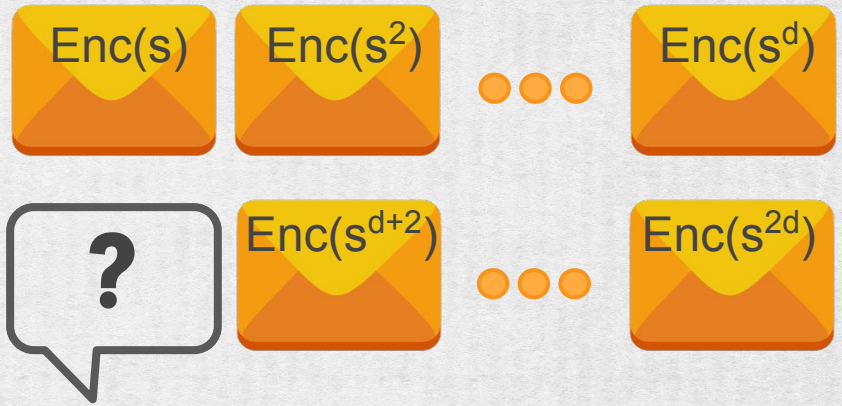


Cheating Prover



Security Reduction: Cheating Prover to d-PDH

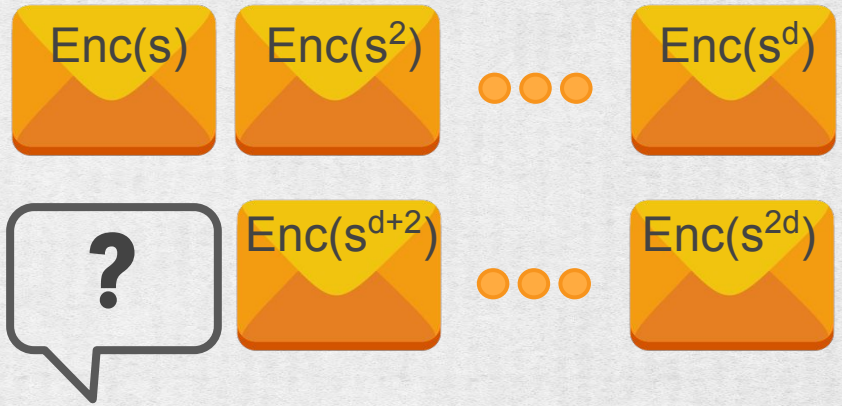
d-PDH



- $t(x)h(x) \neq V^2(x) - 1$, but $\mathrm{Enc}(t(s))H = W^2 - 1$

Security Reduction: Cheating Prover to d-PDH

d-PDH

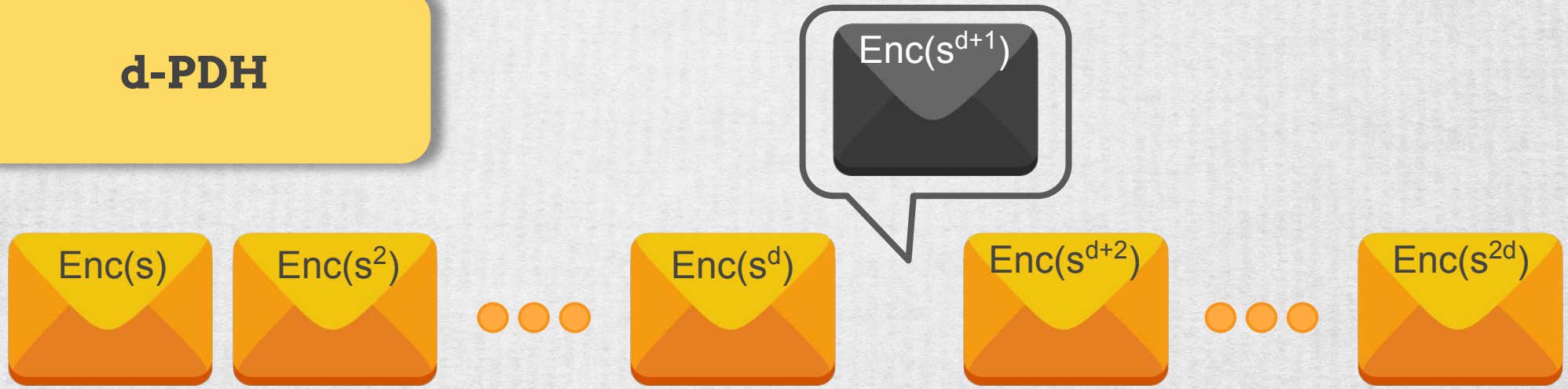


- $t(x)h(x) \neq V^2(x) - 1$, but $\mathrm{Enc}(t(s))H = W^2 - 1$

$p(x) = t(x)h(x) - V^2(x) + 1 \neq 0$, but $p(s) = 0$

Security Reduction: Cheating Prover to d-PDH

d-PDH



- $\mathrm{t(x)h(x)} \neq V^2(x) - 1$, but $\mathsf{Enc}(t(s))H = W^2 - 1$

$$p(x) = t(x)h(x) - V^2(x) + 1 \neq 0, \text{ but } p(s) = 0$$

$$p_{d+1}\,\mathsf{Enc}(s^{d+1}) = Enc(p(s)) - \sum_{i=1,\ldots d}^{d+2,\ldots 2d} p_i Enc(s^i)$$

Security: Types of encodings

Public Verifiable Encoding:

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

Designated Verifiable Encoding:

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

Security: Types of encodings

Public Verifiable Encoding:

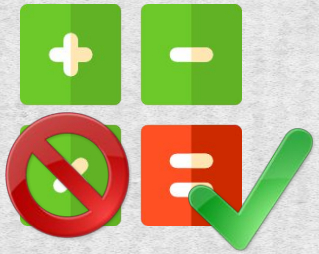
- linear operation using **crs**
- quadratic root detection using **crs**
- image verification using **crs**

Designated Verifiable Encoding:

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

Prover

Verifier



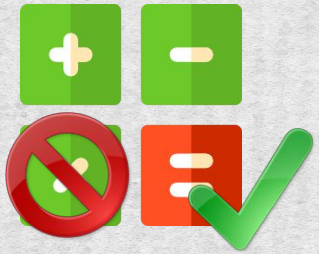
Security: Types of encodings

Public Verifiable Encoding:

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

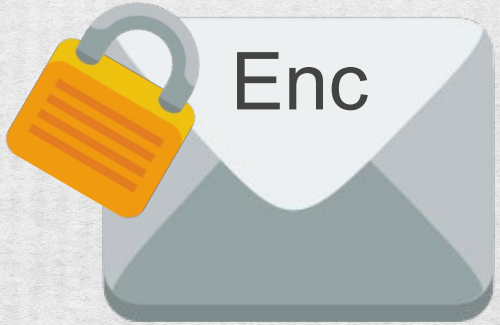
Prover

Verifier



Designated Verifiable Encoding:

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



Prover

Verifier

Security: Publicly Verifiable Encoding

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

$$\langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$
$$e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$
$$e(g^a, g^b) = \tilde{g}^{ab}$$

Prover

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

Security: Publicly Verifiable Encoding

$$Enc(s) = g^s \qquad \langle g \rangle = \mathbb{G}, \langle \tilde{g} \rangle = \tilde{\mathbb{G}}$$
$$e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$
$$e(g^a, g^b) = \tilde{g}^{ab}$$

Prover

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

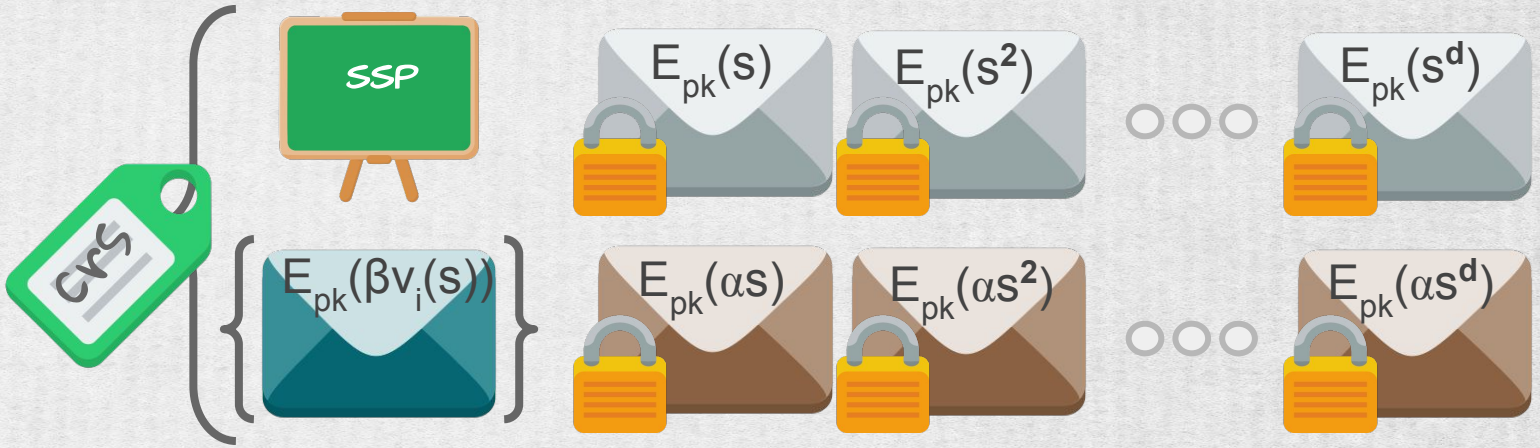
Verifier

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

Security: Designated verifiable Encoding

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

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



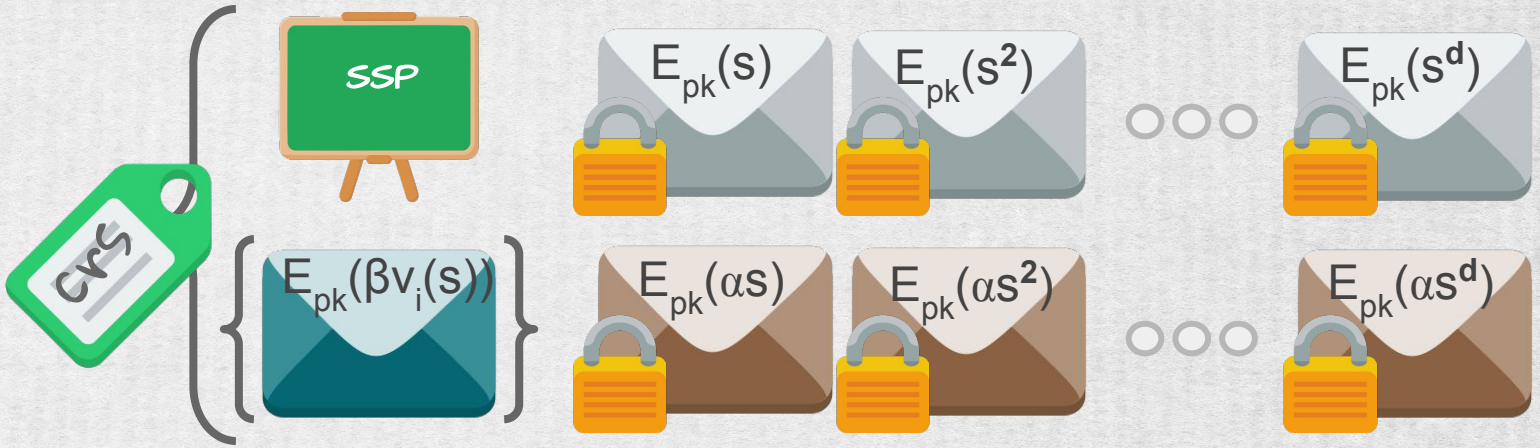
Prover

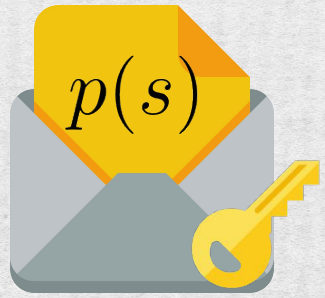
$$Enc(p(s)) = E_{pk}(p(s)) = E_{pk}(\sum p_i s^i) = \sum p_i E_{pk}(s^i)$$

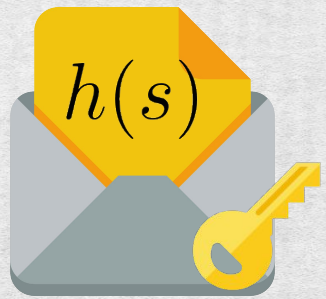
Security: Designated Verifiable Encoding

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

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







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

Lattice-based Encodings: Regev Encryption Scheme

error

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

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

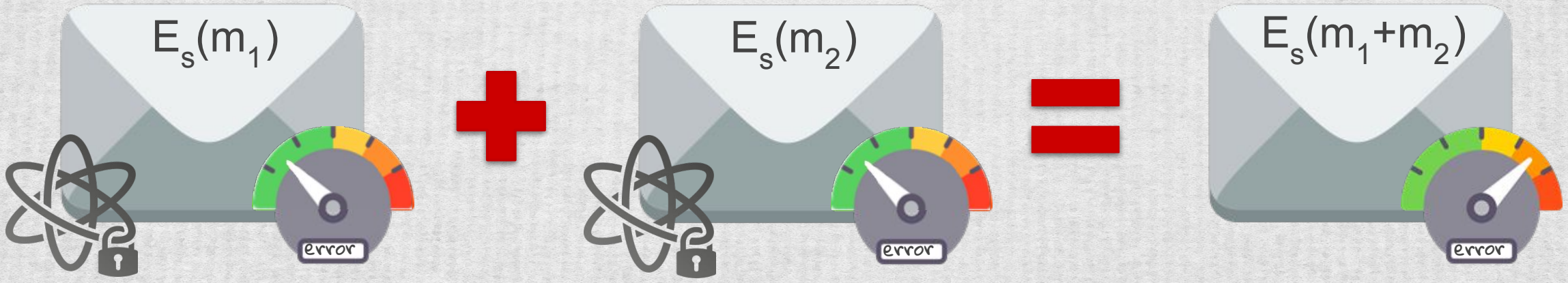
Lattice-based Encodings: Regev Encryption Scheme

error

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

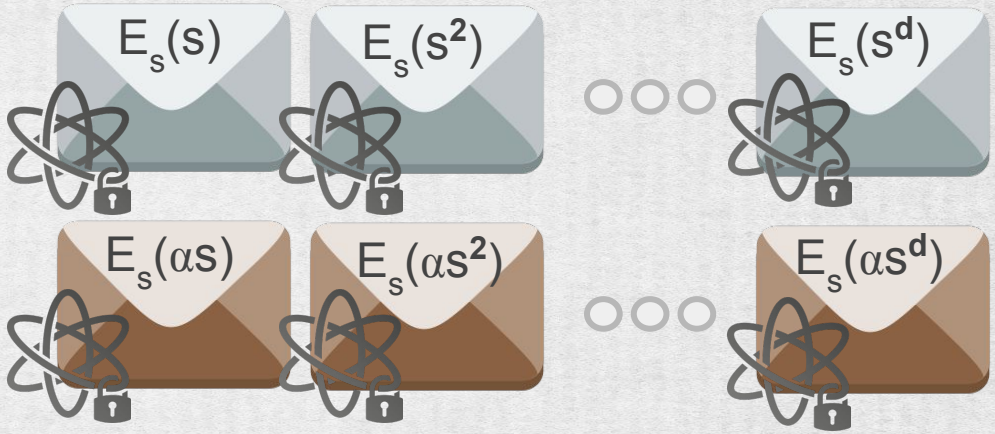
Decryption: $D_{\vec{s}}((\vec{c_0}, c_1)) = \vec{c_0} \cdot \vec{s} + c_1 \pmod{p}$

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



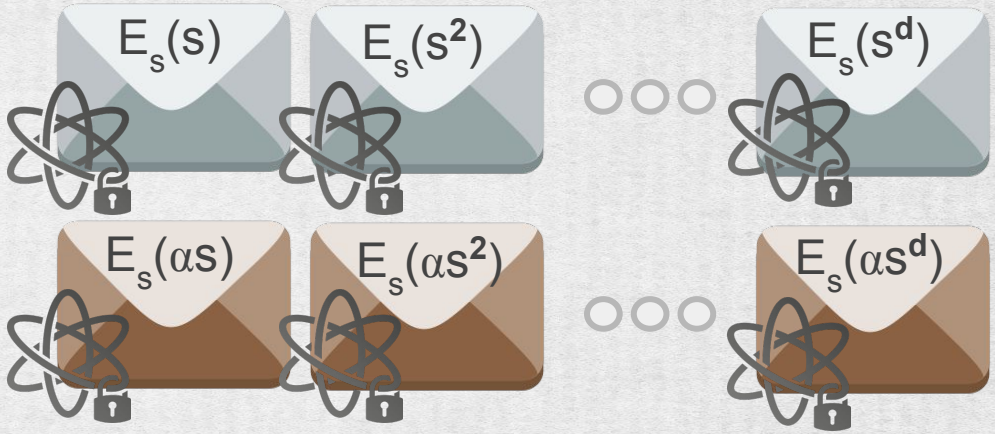
New Lattice Assumptions

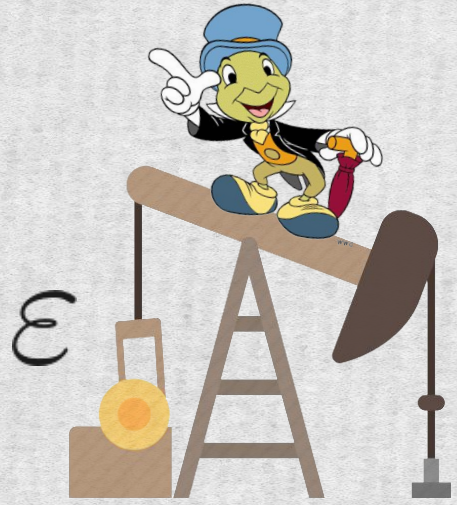
d-PKE



New Lattice Assumptions

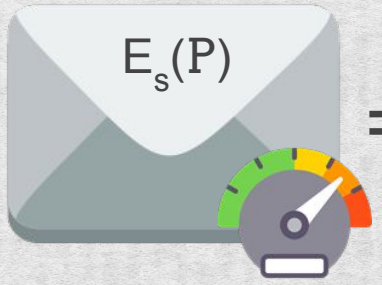
d-PKE





ε

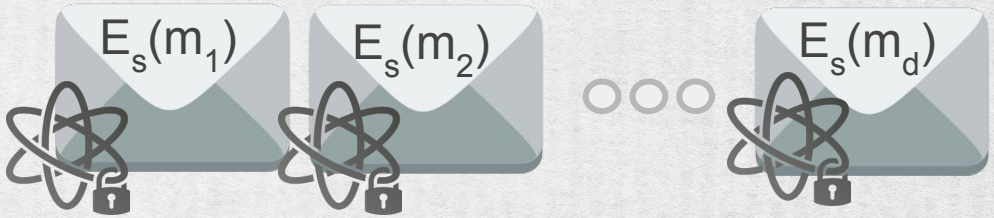
p_1 p_2 ••• p_d

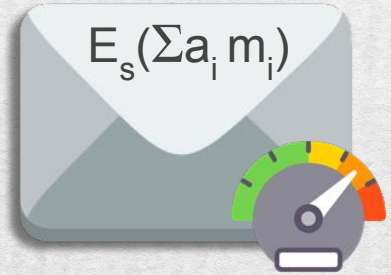


$= \mathrm{Enc}(\Sigma\, p_i s^i)$

Stronger Assumption: Linear-only Encoding

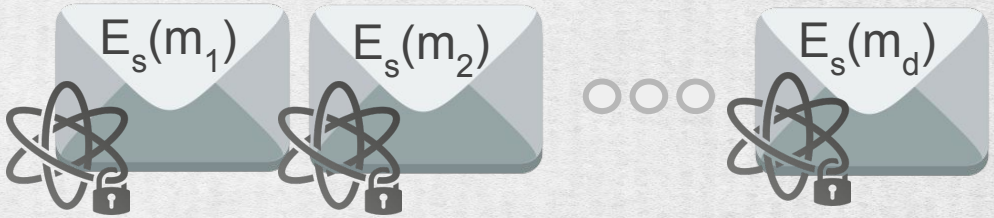
d-LinOnly



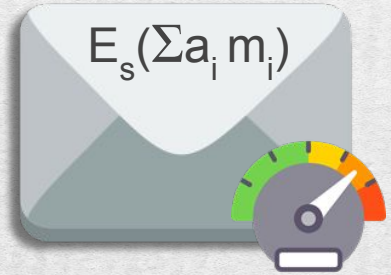
ct = 

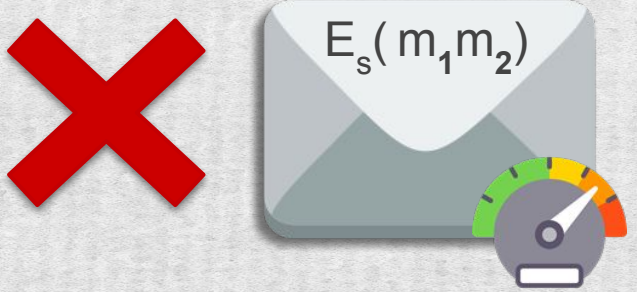
Stronger Assumption: Linear-only Encoding

d-LinOnly



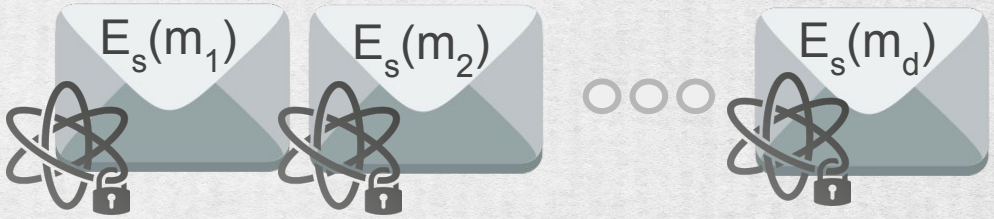
ct =



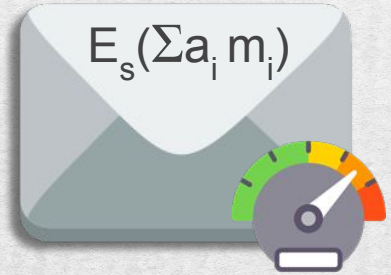


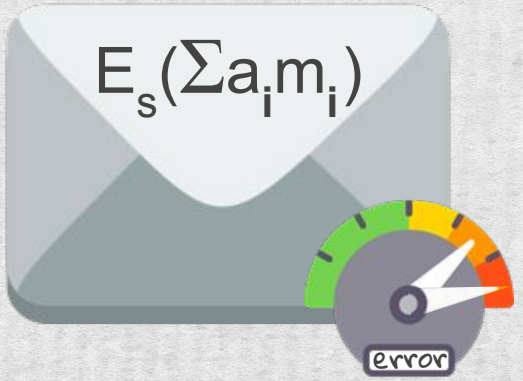
Not Extractable

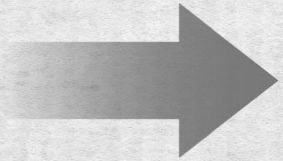
d-LinOnly



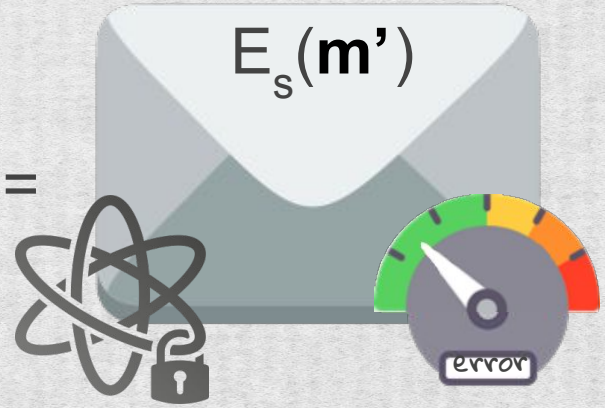
ct =





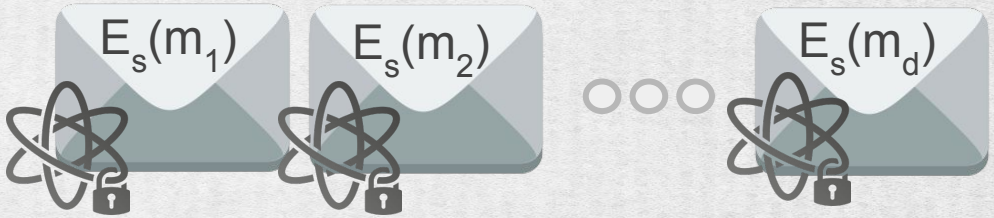


ct =

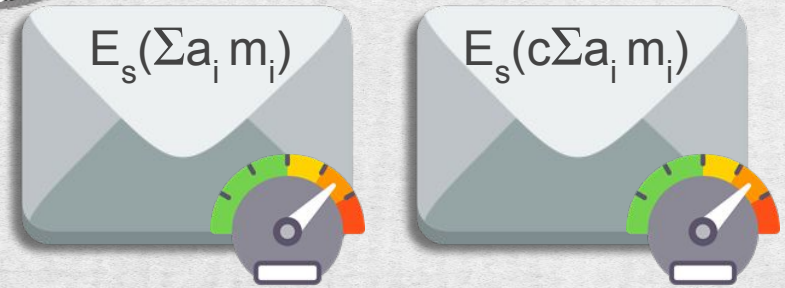


Extractable Linear-only Assumption

Ext LinOnly



ct, ct'

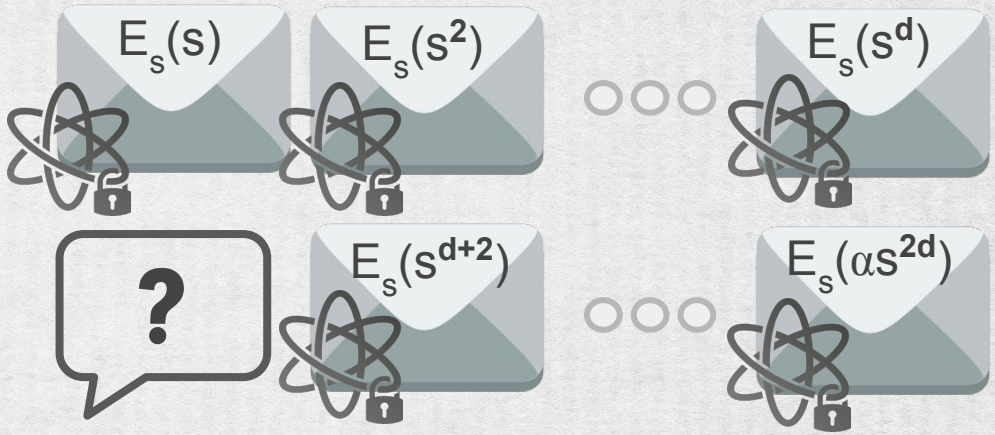


$\mathcal{E}$

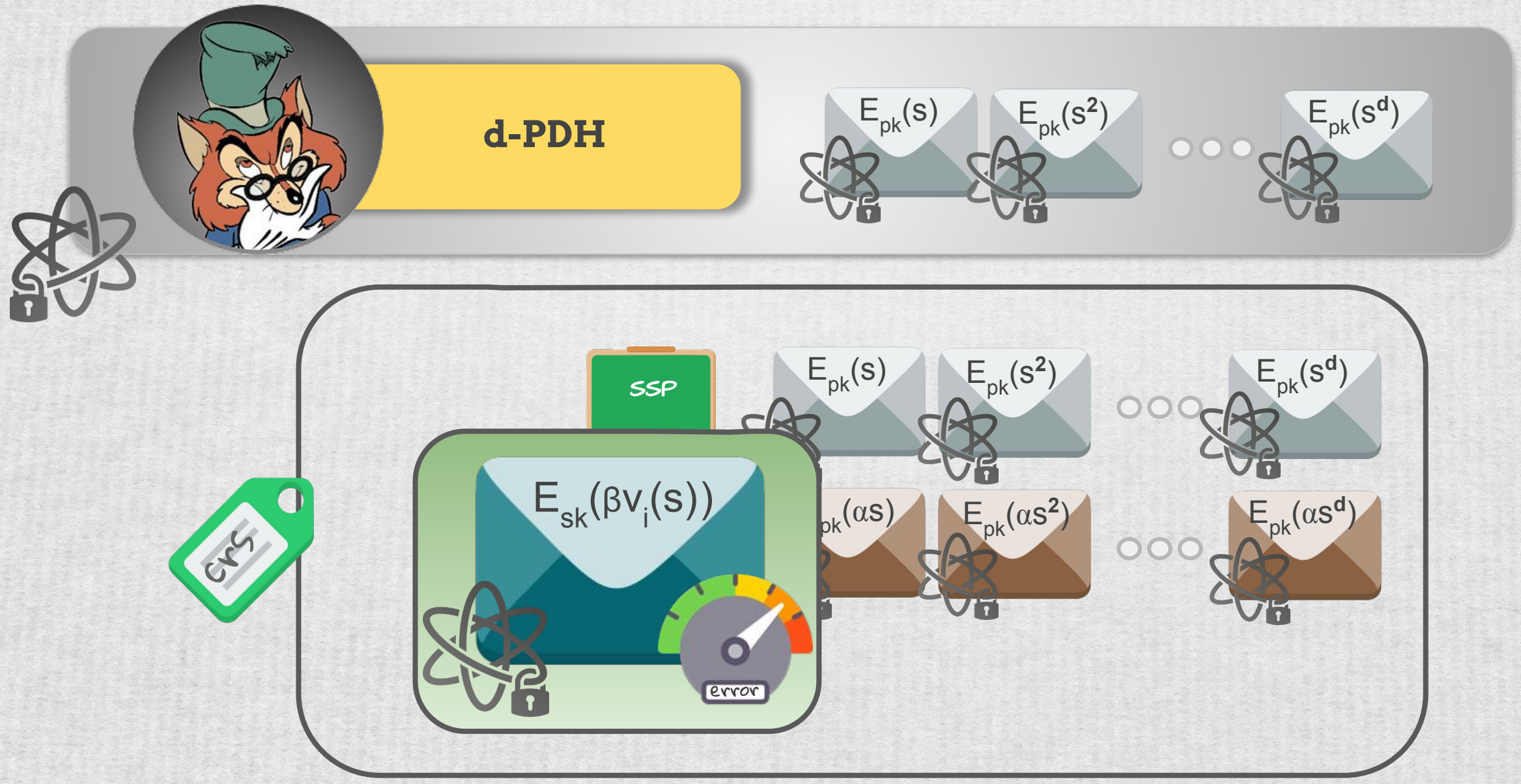
$= \text{Enc}(\Sigma a_i s^i)$

Assumption d-PDH

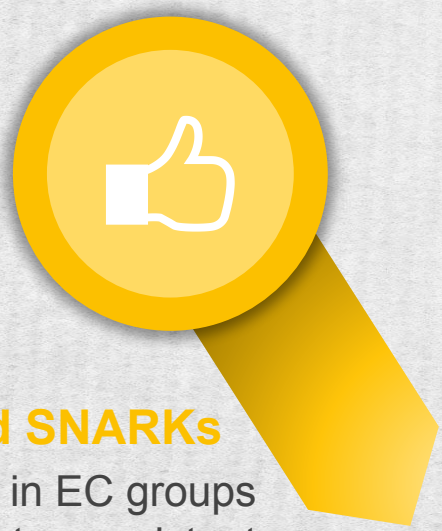
d-PDH



Technical Aspects



SNARKs: Further Directions



Standard SNARKs

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

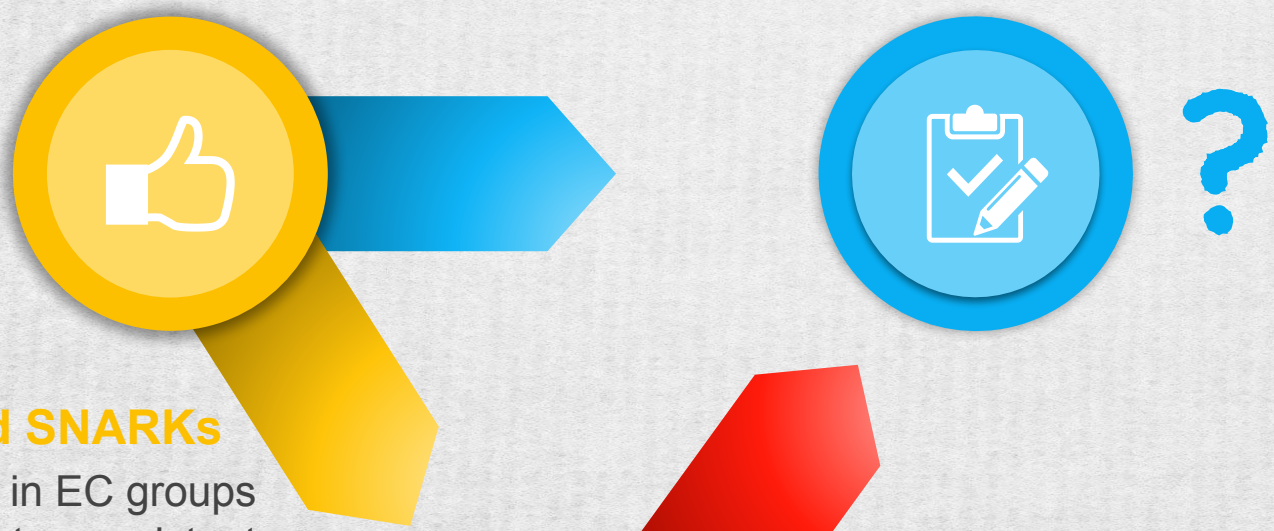
Post-Quantum SNARKs

based on lattice assumptions
designated-verifiable
zero-knowledge

SNARKs: Further Directions

Publicly Verifiable

post-quantum SNARKs



Standard SNARKs

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

Post-Quantum SNARKs

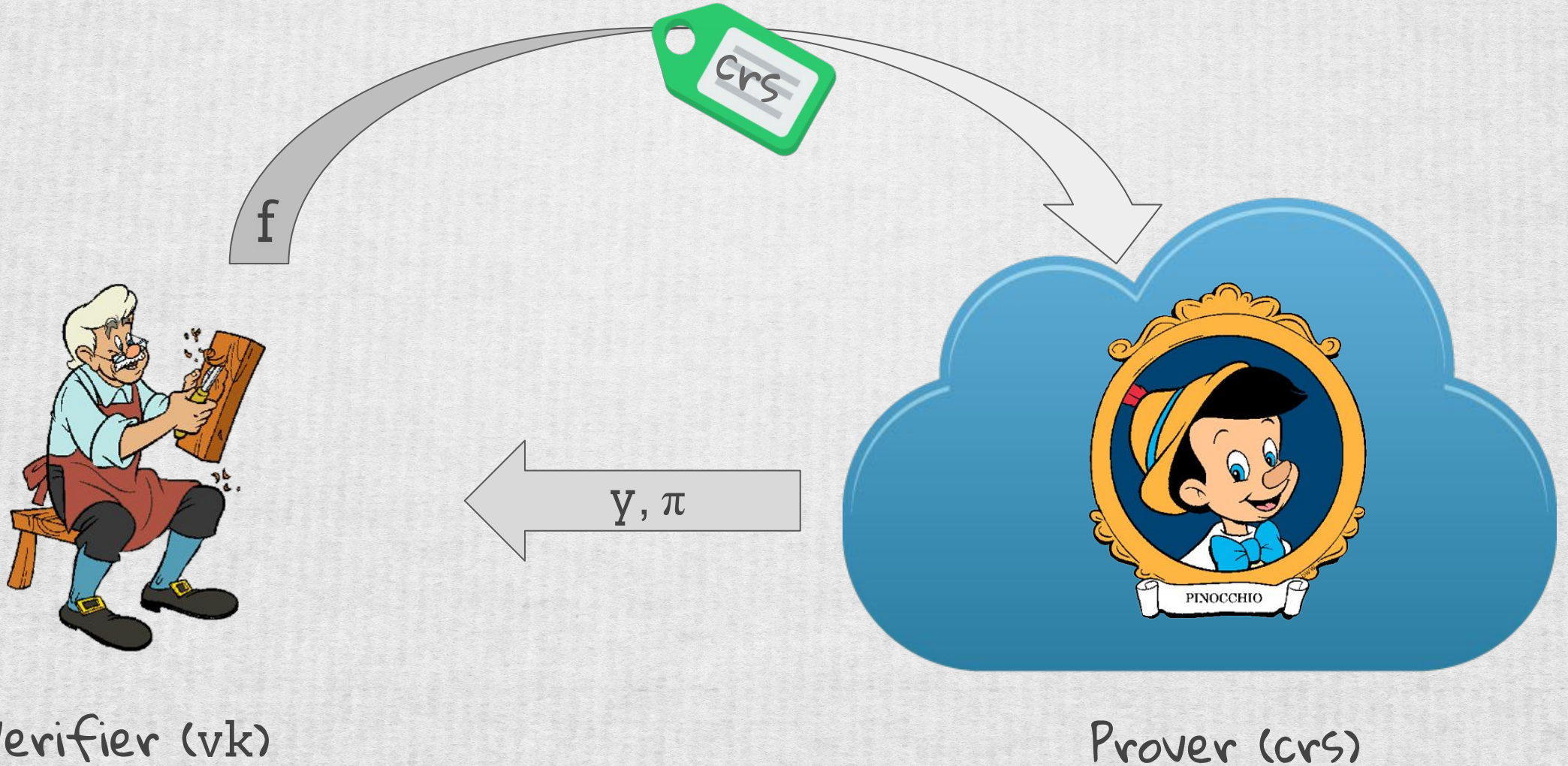
based on lattice assumptions
designated-verifiable
zero-knowledge

More trustful Cloud

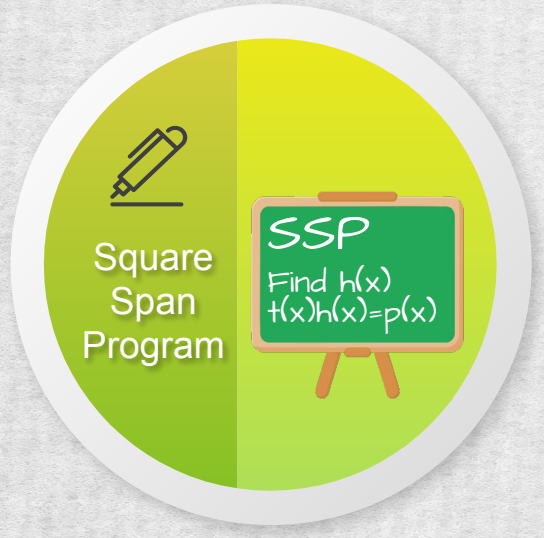
THANK YOU

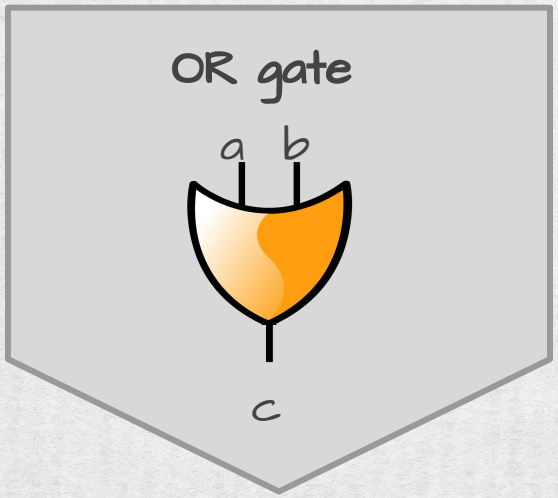
www.di.ens.fr/~nitulesc

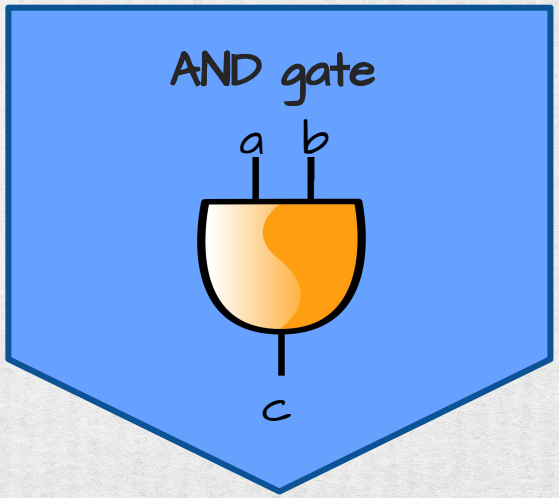
Non-Interactive proofs

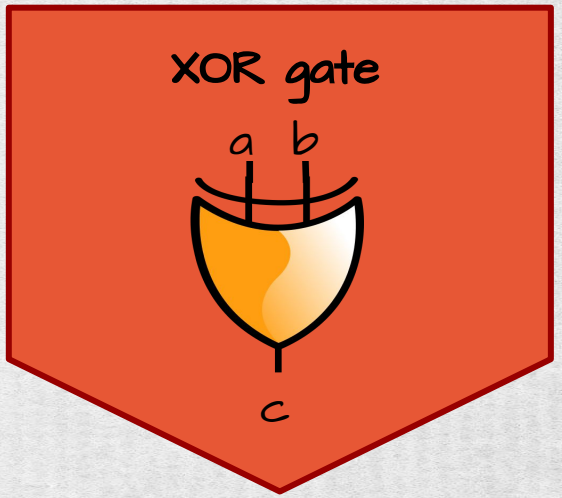


Step 1. Linearization of logic gates





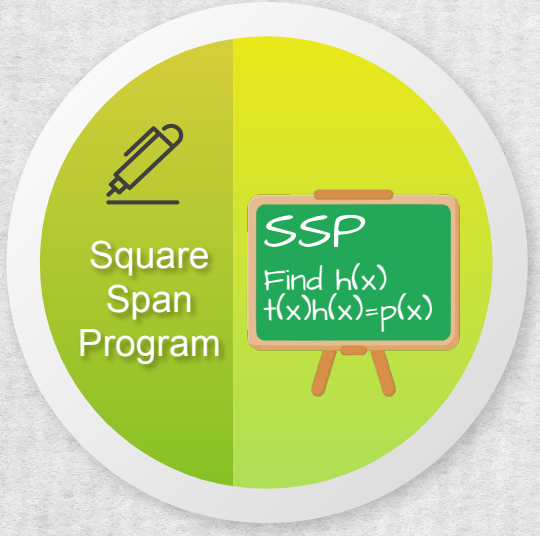




a b c	a b c	a b c
0 0 0	0 0 0	0 0 0
0 1 1	0 1 0	0 1 1
1 0 1	1 0 0	1 0 1
1 1 1	1 1 1	1 1 0
$-a-b+2c \in \{0,1\}$	$a+b-2c \in \{0,1\}$	$a+b+c \in \{0,2\}$

Step 2. Matrix equation for circuit

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



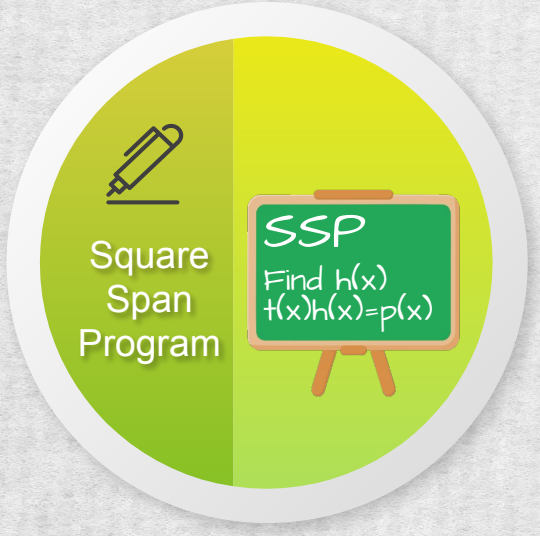
$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$

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

$$\left(V \begin{bmatrix} a \end{bmatrix} + \begin{bmatrix} \delta \end{bmatrix} \right) \circ \left(V \begin{bmatrix} a \end{bmatrix} + \begin{bmatrix} \delta \end{bmatrix} - \begin{bmatrix} 2 \end{bmatrix} \right) = \begin{bmatrix} 0 \end{bmatrix}$$

Step 2. Matrix equation for circuit

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

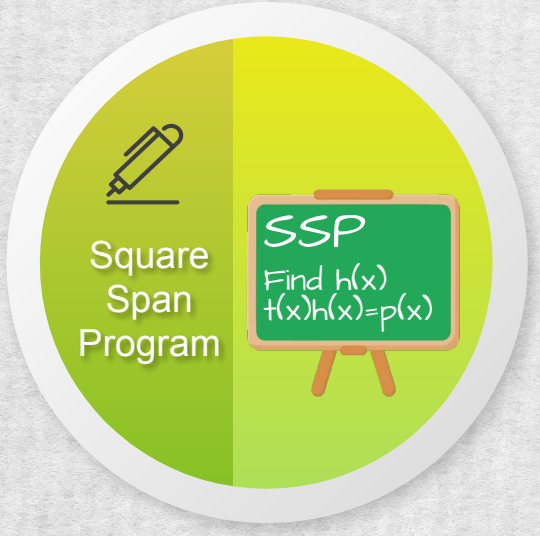


$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$

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

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

Step 3. Polynomial Interpolation



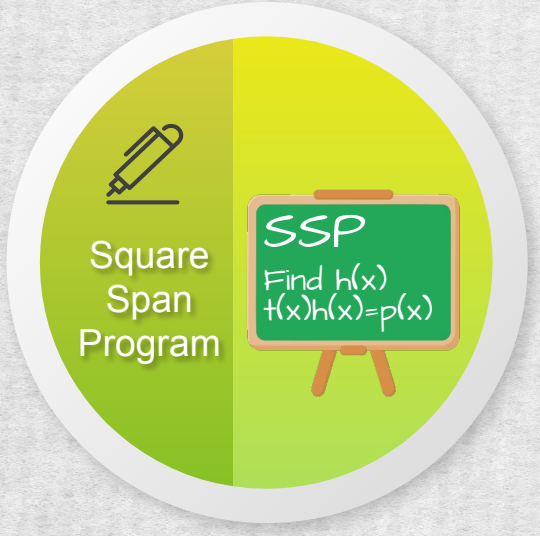
$$\left[V \; a + b-1 \right] \circ \left[V \; a + b-1 \right] = 1$$

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

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

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

Step 4. Polynomial Problem SSP

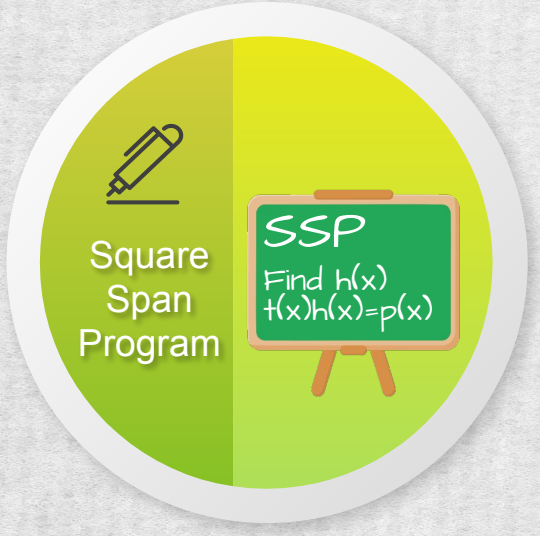


$$\left(v_0(r_j) + \sum_{i=1}^{m} a_i v_i(r_j)\right)^2 - 1 = 0$$

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

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

Step 4. Polynomial Problem SSP



$$t(x) \mid V(x)^2 - 1$$

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

$$t(x) = \prod_{j=1}^{d} (x - r_j)$$

Step 4. Polynomial Problem SSP