SNARKs Tutorial:

Introduction & Circuit-Based Construction

Anca Nitulescu

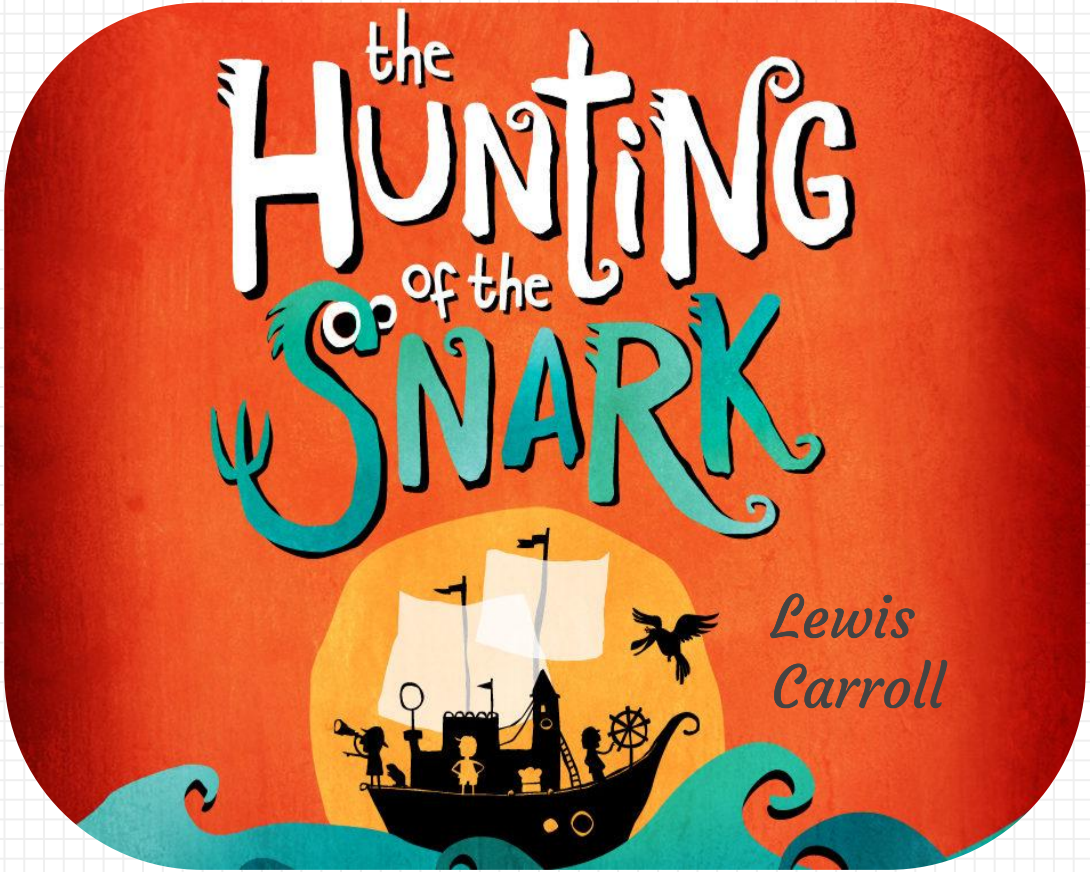
AARHUS UNIVERSITY

"For the Snark's a
peculiar creature, that won't

Be caught in a commonplace way.

Do all that you know,
and try all that you don't:

Not a chance must be wasted
to-day!"



Succinct Non-interactive Arguments of Knowledge

Short Pairing-based Non-interactive Zero-Knowledge Arguments

Jens Groth*

Progression-Free Sets and Sublinear Pairing-Based Non-Interactive Zero-Knowledge Arguments

Helger Lipmaa

Quadratic Span Programs and Succinct NIZKs without PCPs

Rosario Gennaro* Craig Gentry† Bryan Parno‡ Mariana Raykova§

The Hunting of the SNARK*

Nir Bitansky† Ran Canetti‡ Alessandro Chiesa§ Shafi Goldwasser§ Huijia Lin¶ Aviad Rubinstein† Eran Tromer†

July 24, 2014

Abstract

The existence of succinct non-interactive arguments for NP (i.e., non-interactive computationally-sound proofs where the verifier's work is essentially independent of the complexity of the NP nondeterministic verifier) has been an intriguing question for the past two decades. Other than CS proofs in the random oracle model [Micali, FOCS '94], the only existing candidate construction is based on an elaborate assumption that is tailored to a specific protocol [Di Crescenzo and Lipmaa, CiE '08].

Outline

SNARK
Background

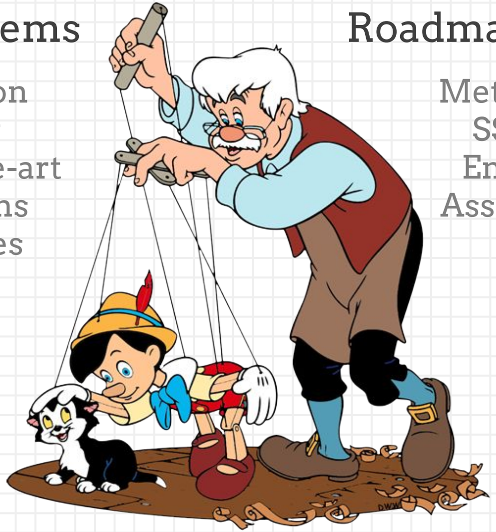
Frameworks
for SNARKs

Construction
Security

The END

Proof Systems

Motivation
History
State-of-the-art
Definitions
Properties

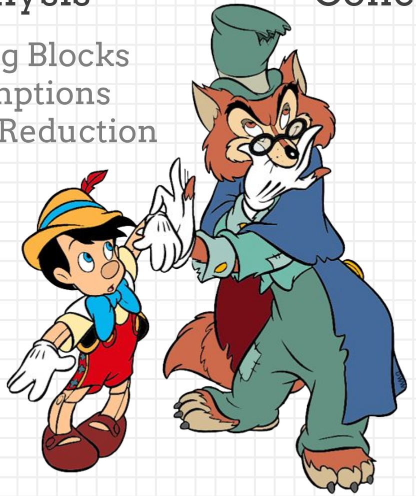


Roadmap and Tools

Methodology
SSP, QAP
Encodings
Assumptions

Analysis

Building Blocks
Assumptions
Security Reduction



Conclusions

SNARK

SNARK
Background

Framework
for SNARKs

Construction
Security

The
END

Verifier

Prover

Delegated Computation

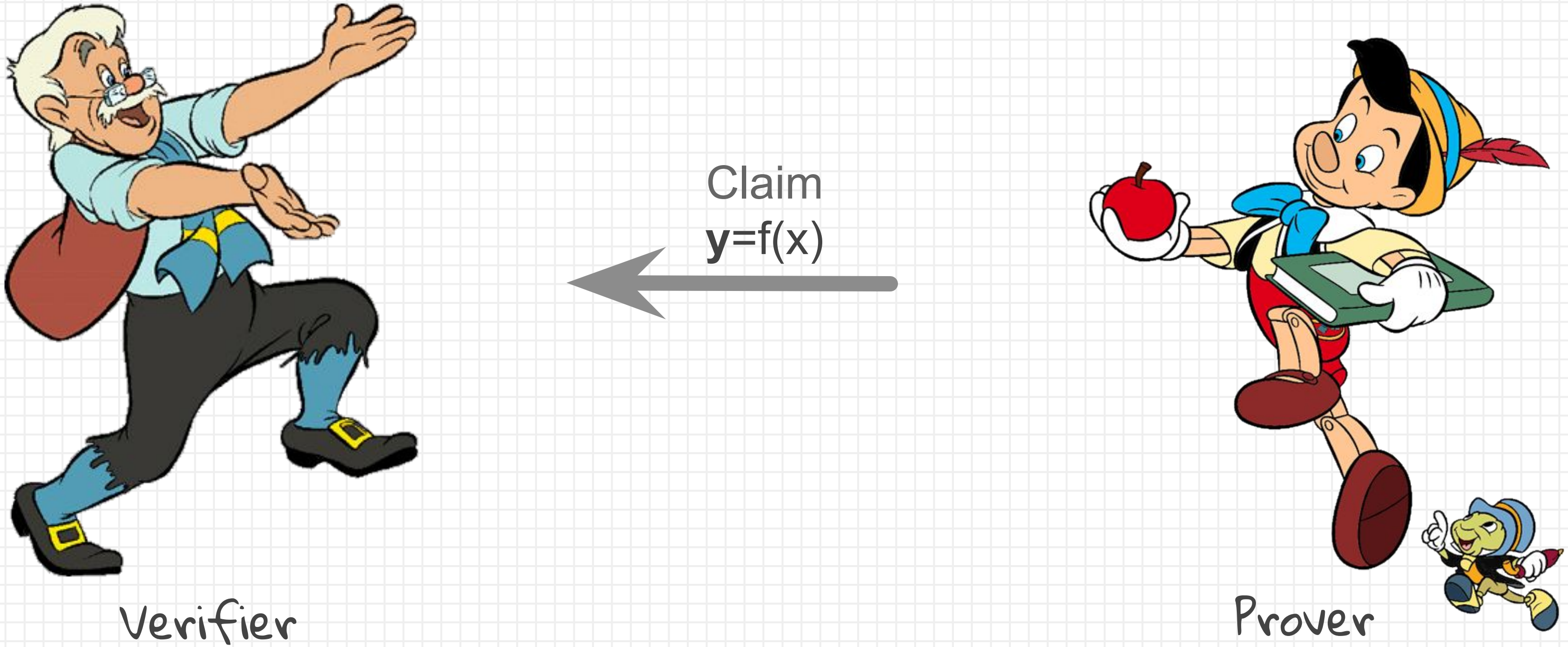
Verifier

Task

computes
f(x)=**y**

Prover

Prover claims a statement



Verifier does not trust

$f(x)=\mathbf{y}$

$\mathbf{y} \neq f(x)$

Verifier

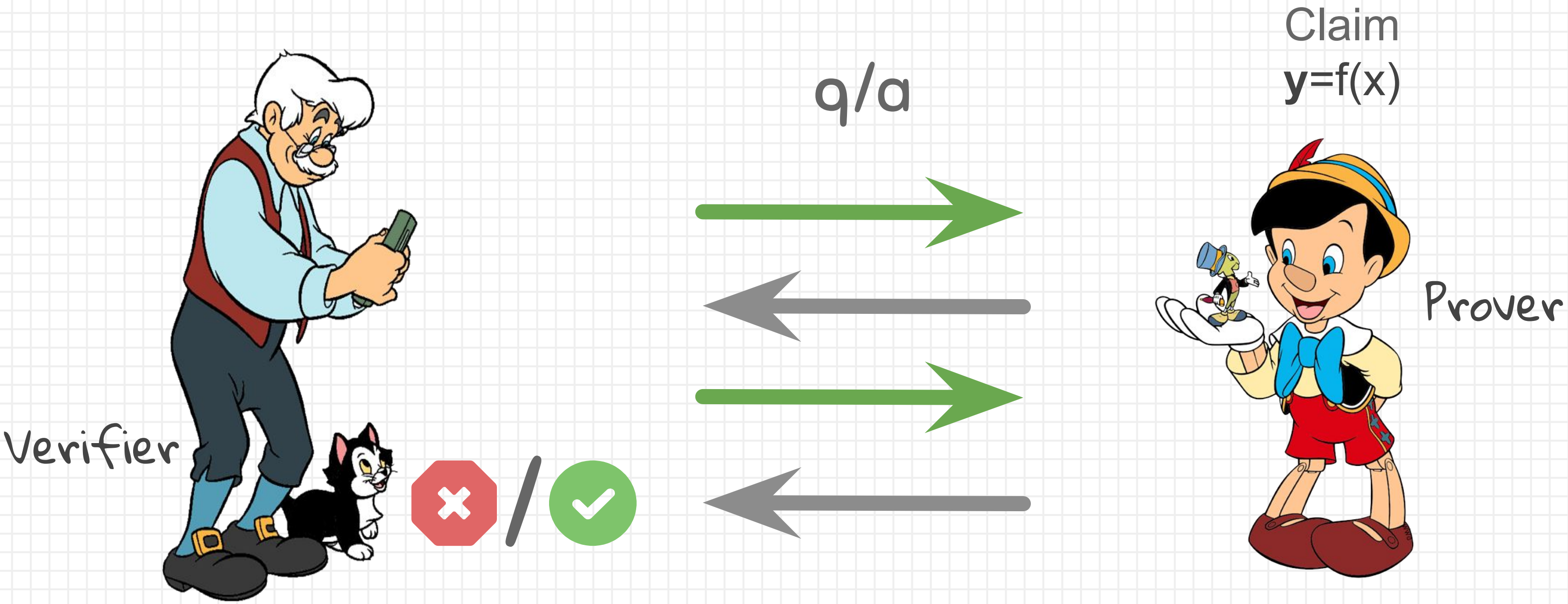
Corrupted Prover

Proof Systems: Since the 1980s

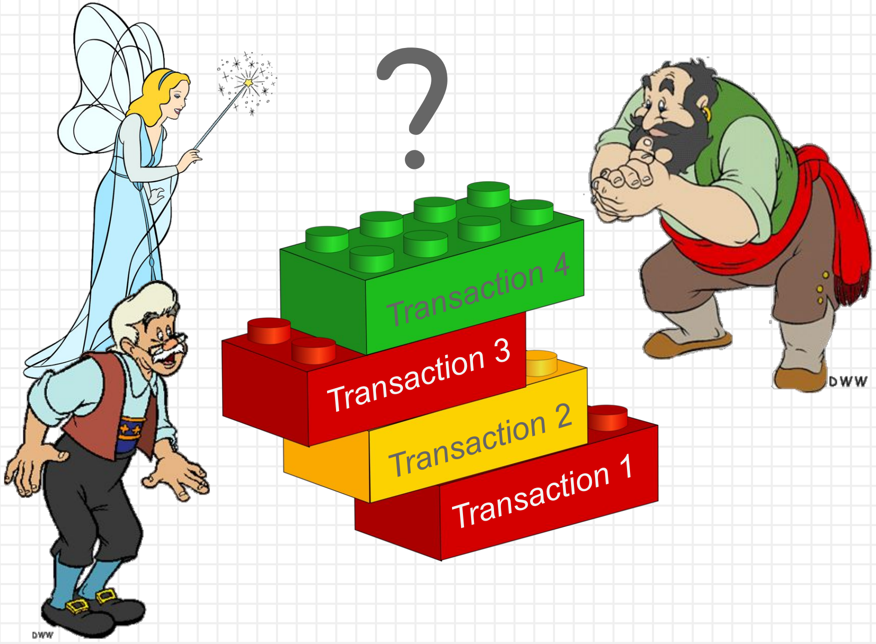
Joe Kilian

[Kil92]
A note on efficient zk-proofs and arguments

Interactive Proof Protocol [Kil92]



Decentralised Setting: Multiple Verifiers?



Verifiers

Prover

Proof Systems: Non-Interactive Arguments

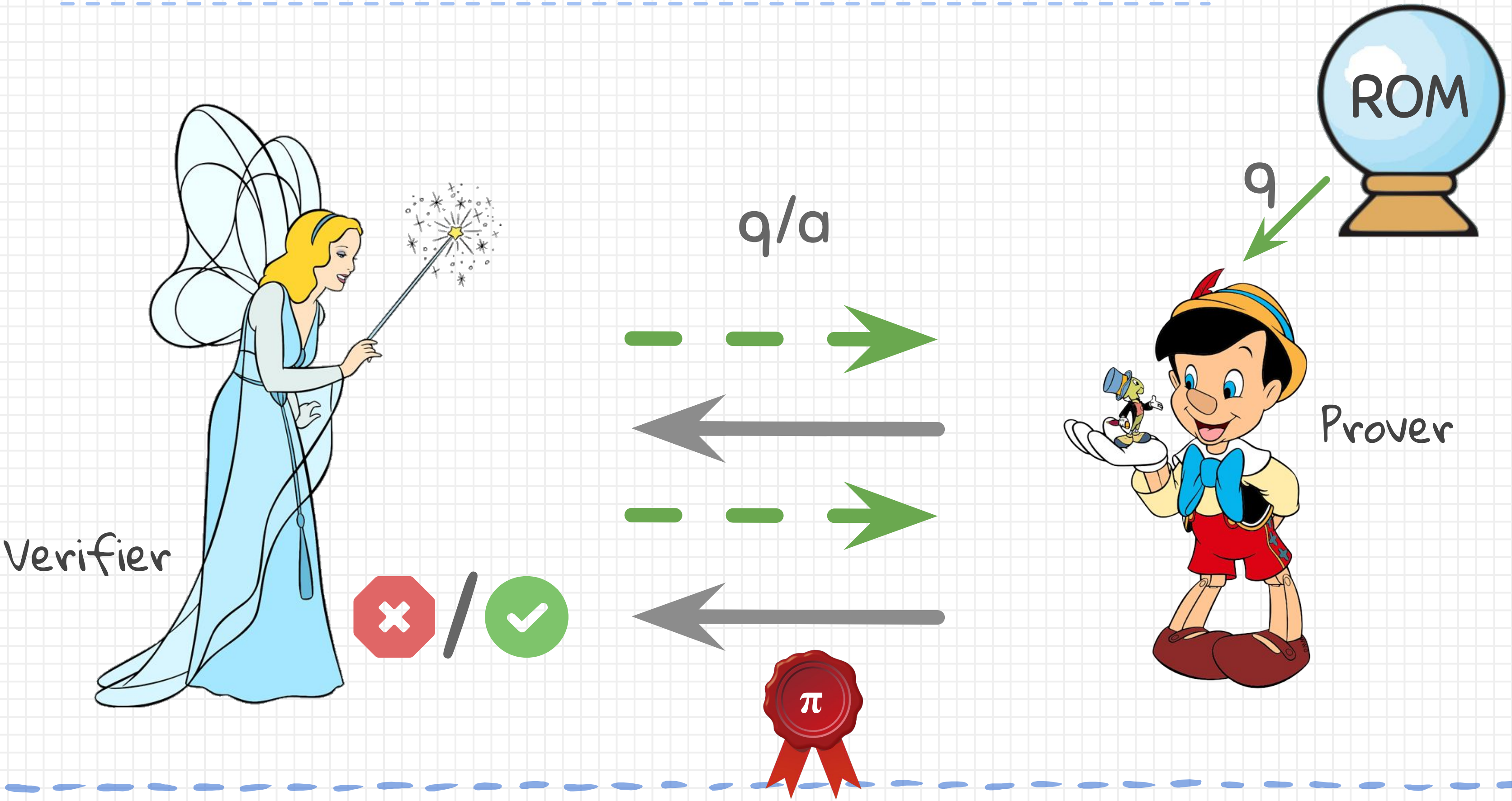
Joe Kilian

[Kil92]
A note on efficient zk-proofs and arguments

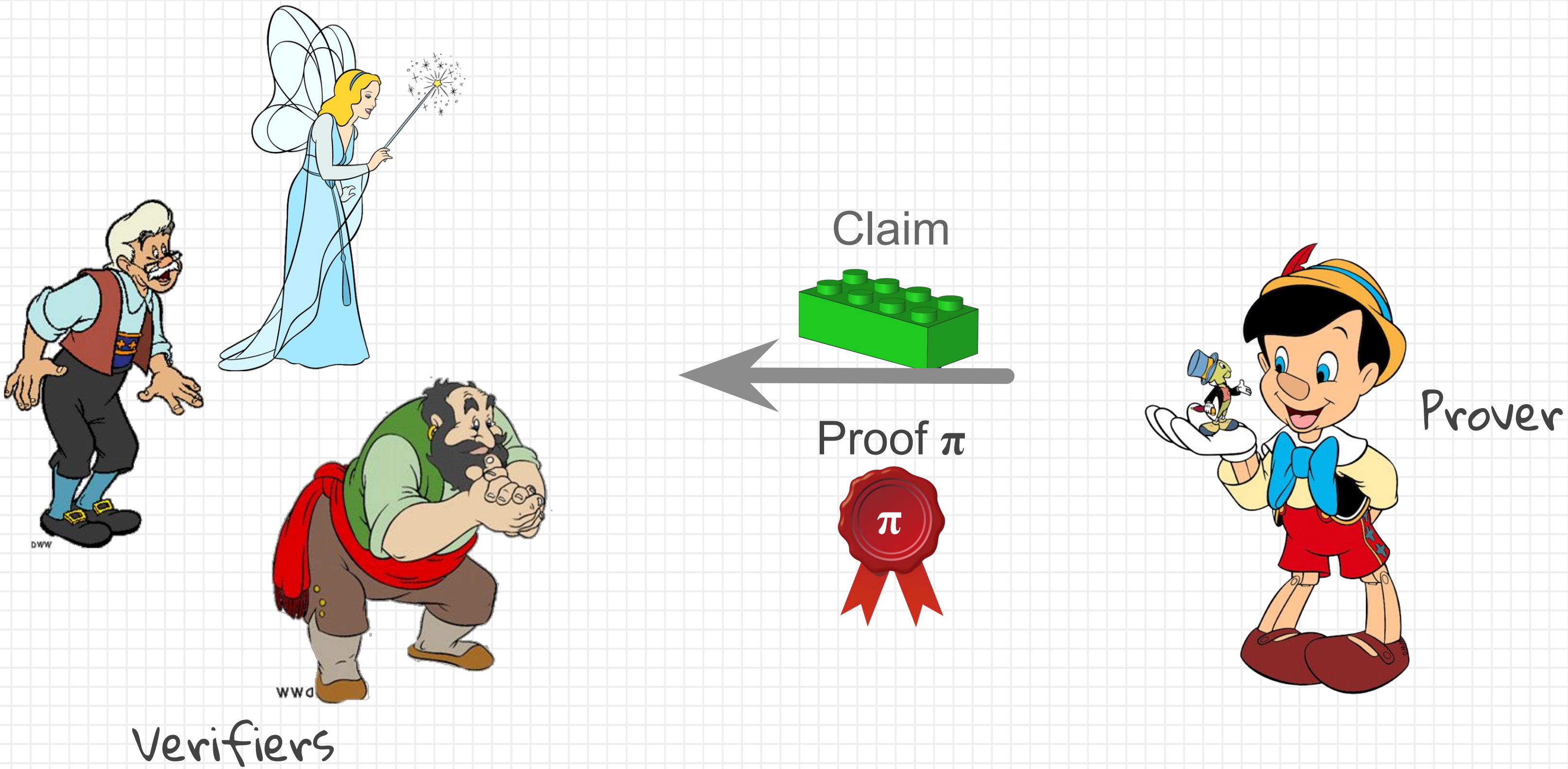
[Mic00]
Computationally sound proofs

Silvio Micali

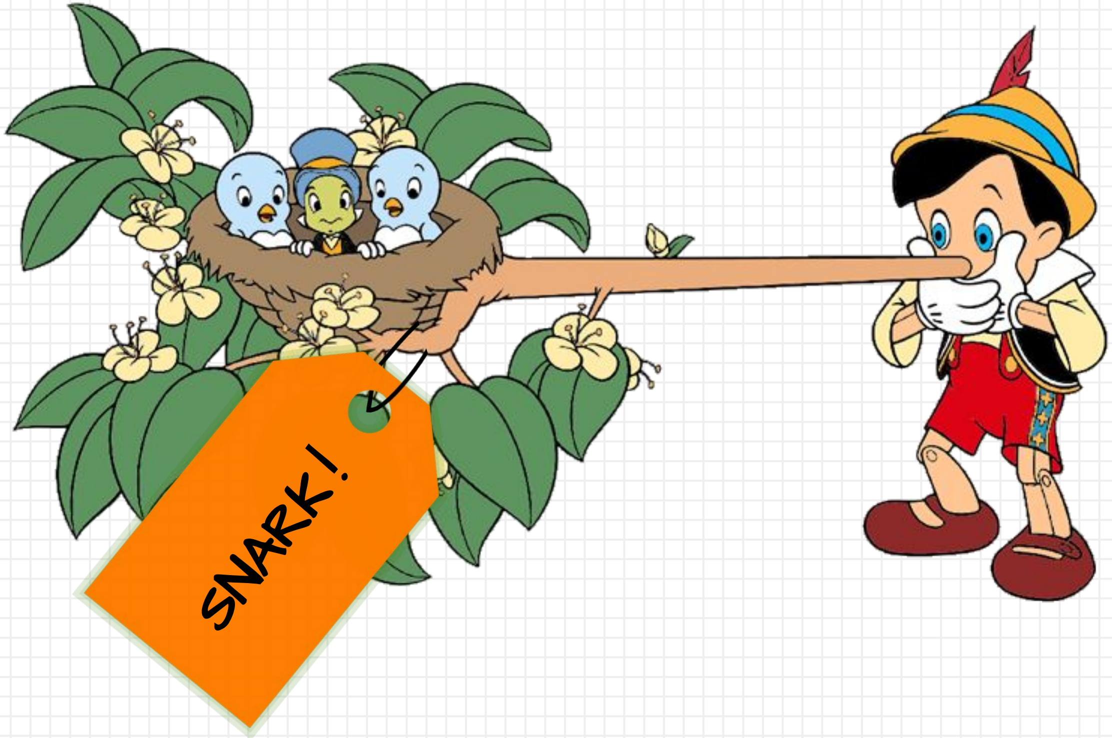
Removing Interaction with Random Oracle



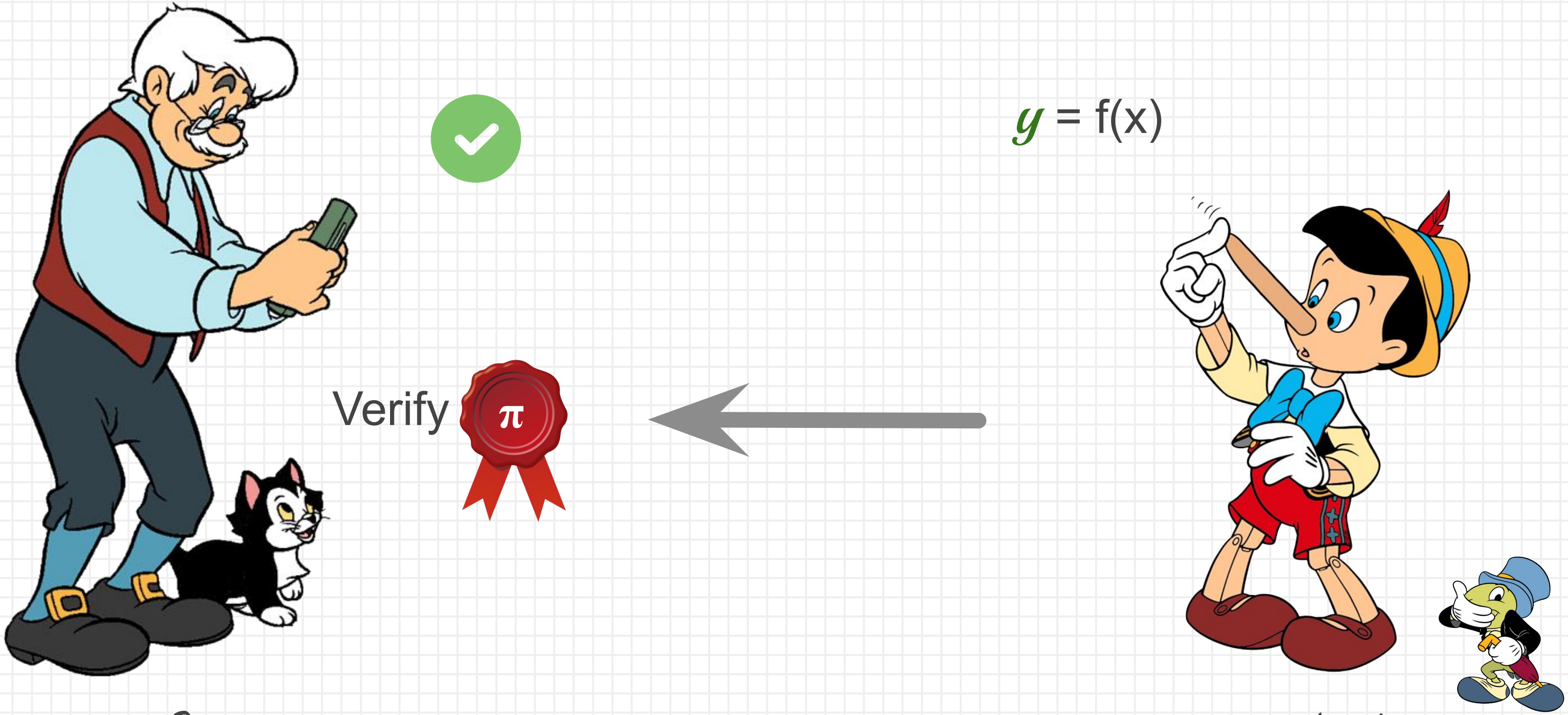
Non-Interactive Proof Protocol [Mic00]



Succinct Non-interactive ARgument of Knowledge



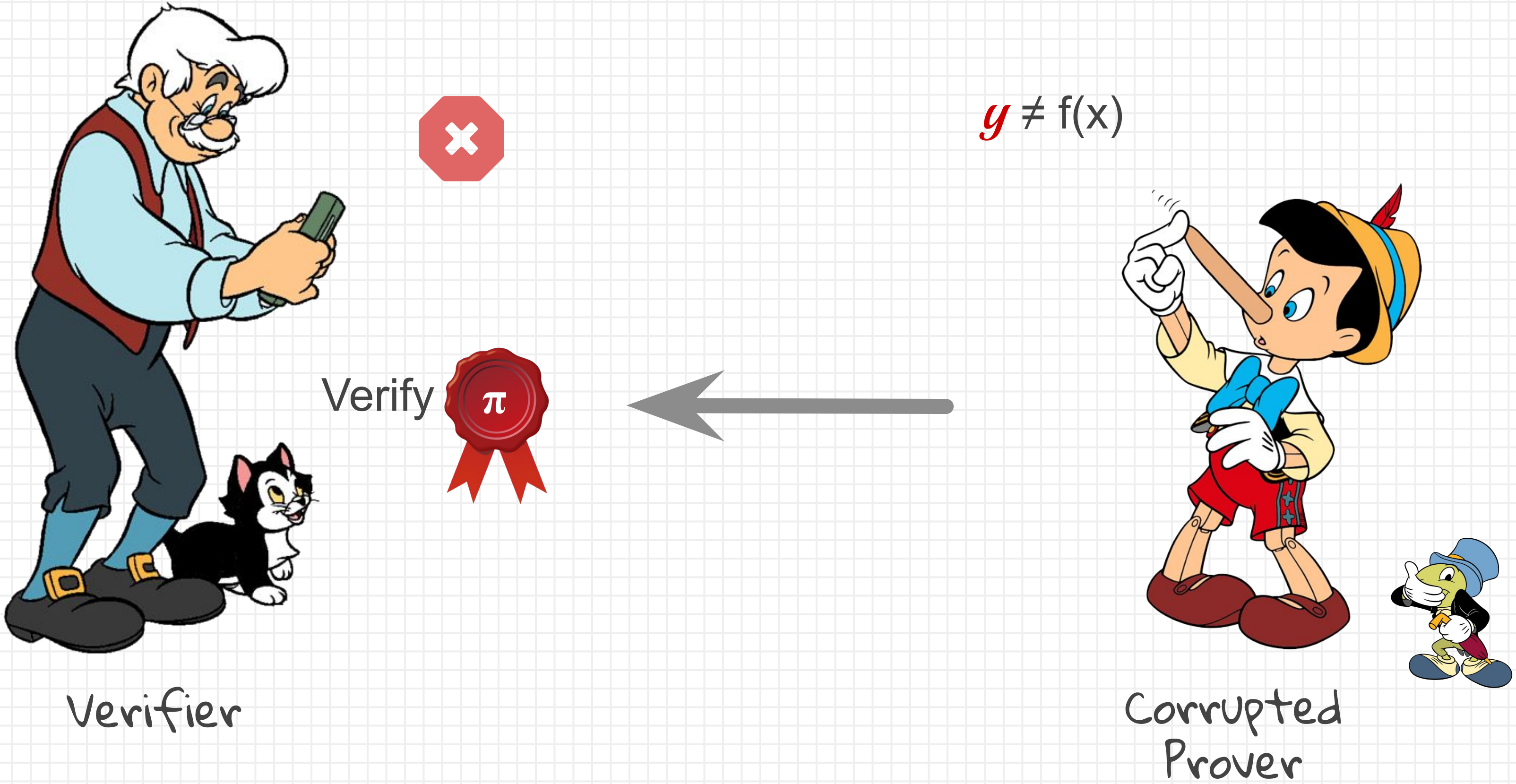
Correctness and Soundness



Verifier

Corrupted Prover

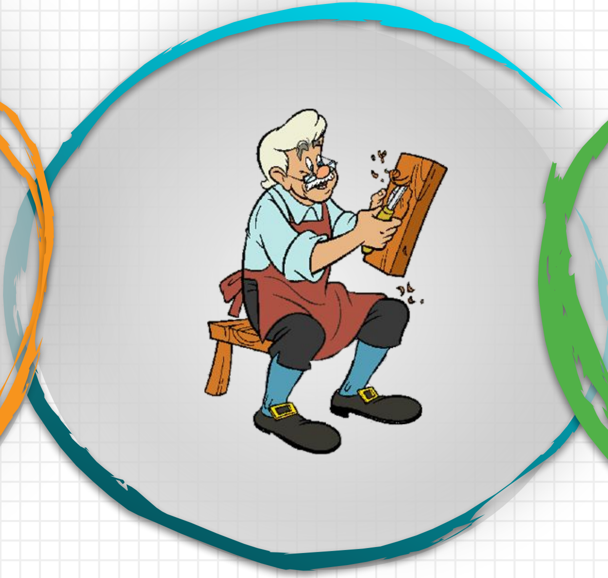
Correctness and Soundness



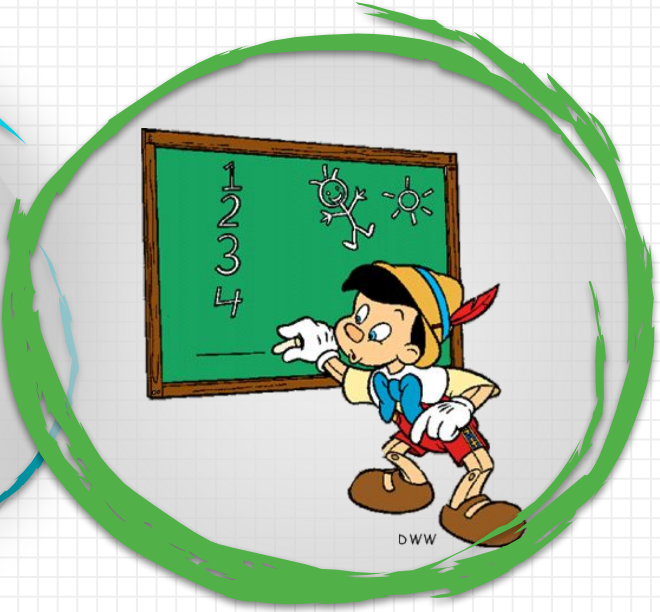
Extra Properties for the Proof

Succinct Proof

Efficient Verification



Knowledge Soundness



Pre-Processing for Efficient Arguments

Joe Kilian

[Kil92]
A note on efficient zk-proofs and arguments

[Mic00]
Computationally sound proofs

Silvio Micali

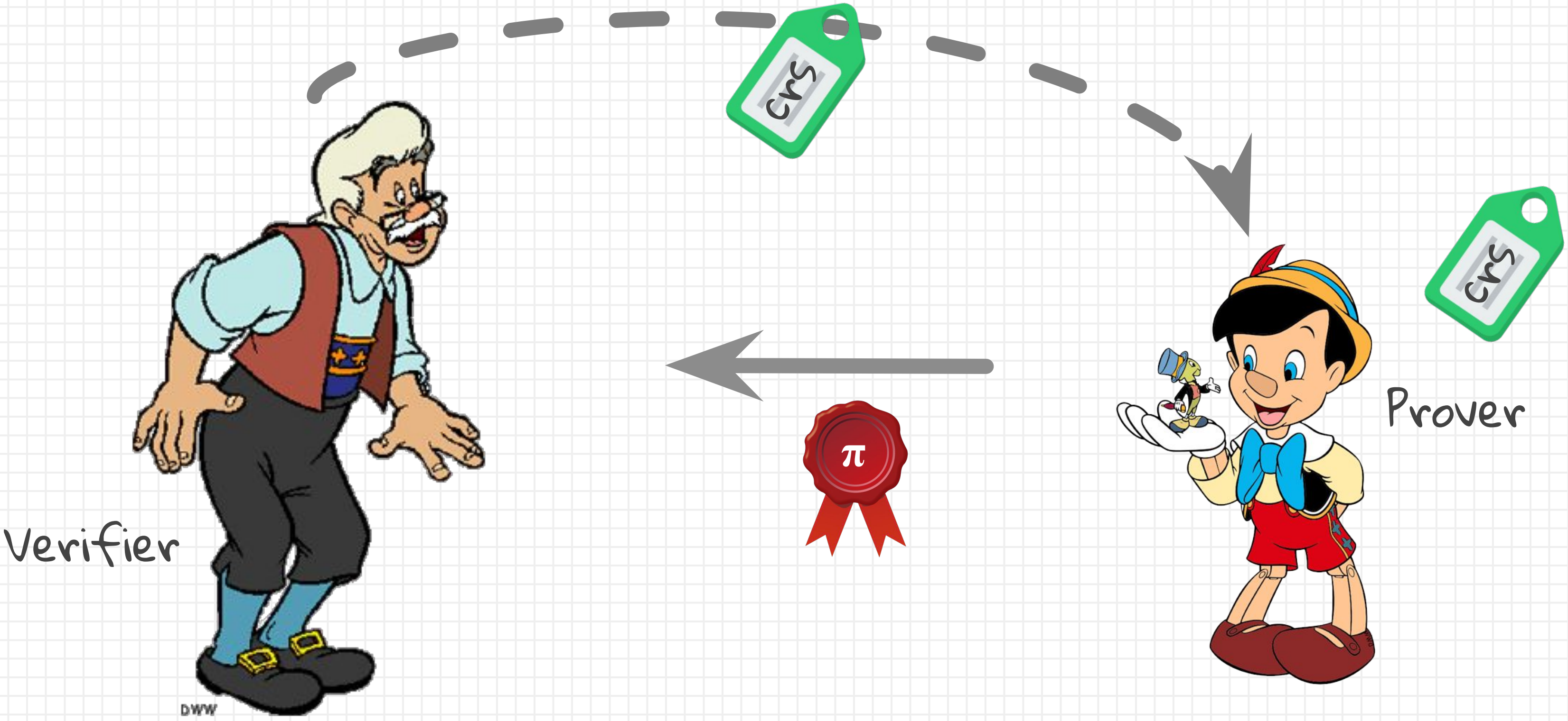
G. Di Crescenzo
Helger Lipmaa

[DCL08]
Succinct NP proofs from an extractability assumption

J. Groth

[Gro10]
Short Pairing-based Non-interactive Zero-Knowledge Arguments

One round Interaction



Strong Assumptions

Joe Kilian

[Kil92]
A note on efficient zk-proofs and arguments

Silvio Micali

[Mic00]
Computationally sound proofs

G. Di Crescenzo
Helger Lipmaa

[DCL08]
Succinct NP proofs from an extractability assumption

J. Groth

[Gro10]
Short Pairing-based Non-interactive Zero-Knowledge Arguments

Craig Gentry
Daniel Wichs

[GW11]
Separating succinct non-interactive arguments from all falsifiable assumptions

State-of-the-art

R. Gennaro, C. Gentry, B. Parno, M. Raykova

[GGPR13]
QSP and succinct NIZKs without PCPs

PKE

B. Parno, J.Howell, C. Gentry, M. Raykova

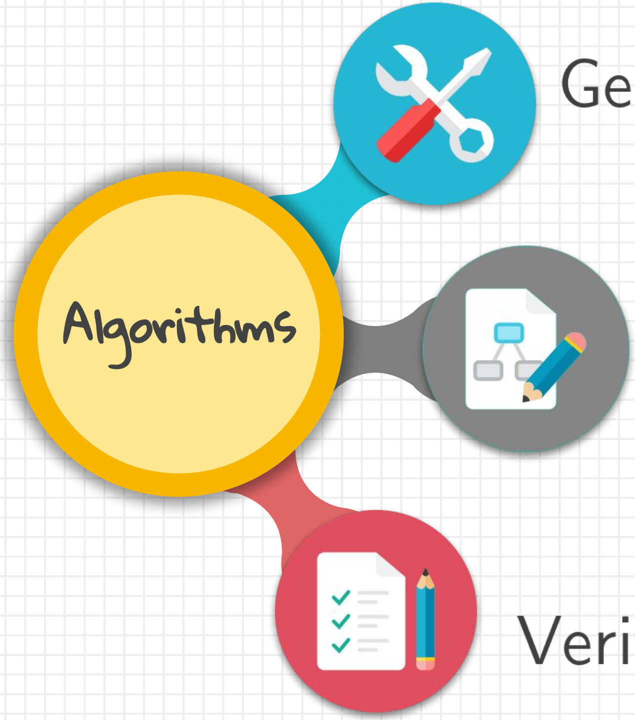
[PHGR13]
Pinocchio: Nearly practical verifiable computation

ECRH

[BCI+13]
SNARGs via linear interactive proofs

N.Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, O Paneth

SNARK with Preprocessing

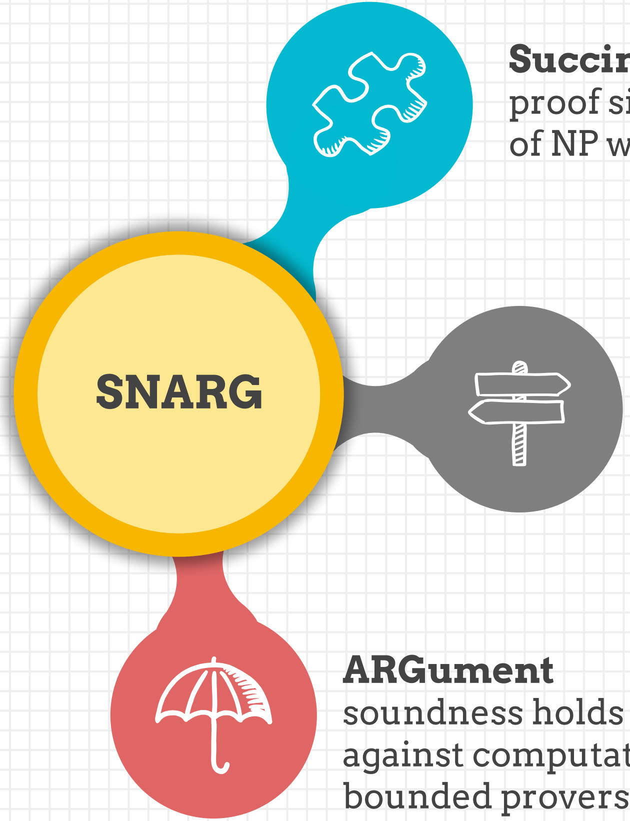


$\mathsf{Gen}(1^\lambda, \mathcal{R}) \to (\mathsf{crs}, \mathsf{vk})$

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

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

SNARG: Succinct Non-Interactive ARGument

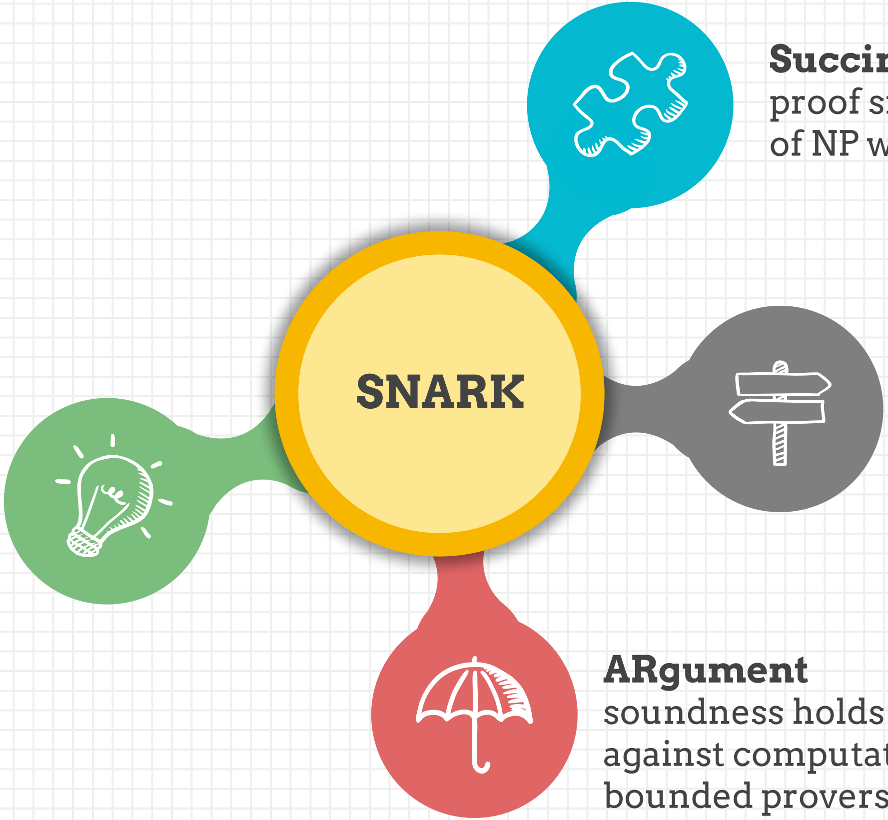


Succinctness
proof size independent of NP witness size

Non-Interactivity
no exchange between prover and verifier

ARGument
soundness holds only against computationally bounded provers

SNARK: Succinct Non-Interactive ARgument of Knowledge



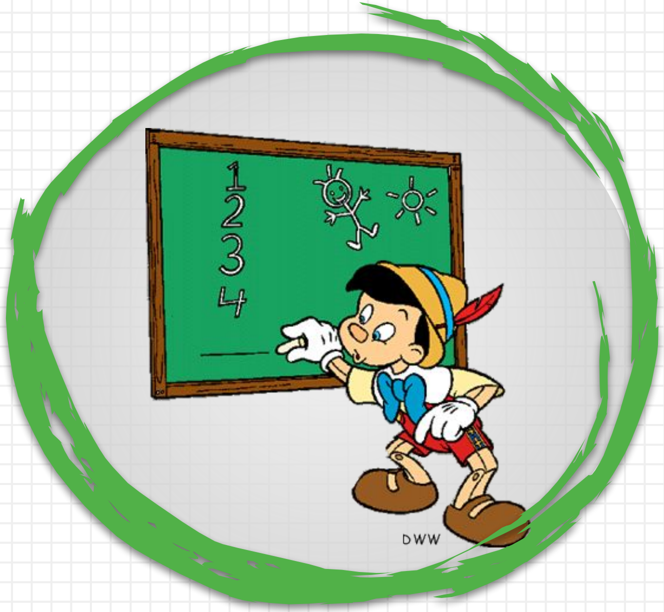
Succinctness
proof size independent of NP witness size

Non-Interactivity
no exchange between prover and verifier

Knowledge Soundness
a witness can be efficiently extracted from the prover

ARgument
soundness holds only against computationally bounded provers

Knowledge Soundness



Knowledge Soundness

$\mathcal{A}$

crs

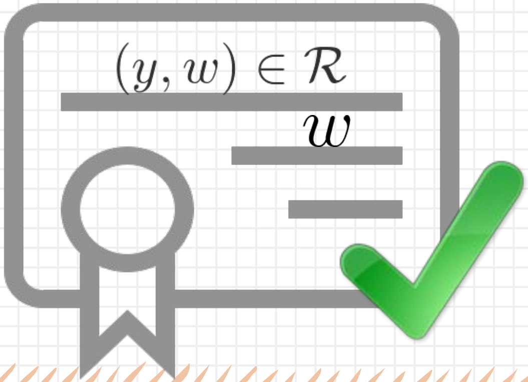
Adversary

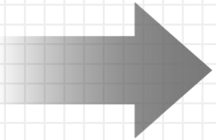
$\mathcal{E}$

crs

extractor

π





Zero-Knowledge SNARK

zk-SNARK

Zero-Knowledge
does not leak anything about the witness

Succinctness
proof size independent of NP witness size

Non-Interactivity
no exchange between prover and verifier

Knowledge Soundness
a witness can be efficiently extracted from the prover

Argument
soundness holds only against computationally bounded provers

Zero-Knowledge: Verifier learns nothing about witness

Zero-Knowledge

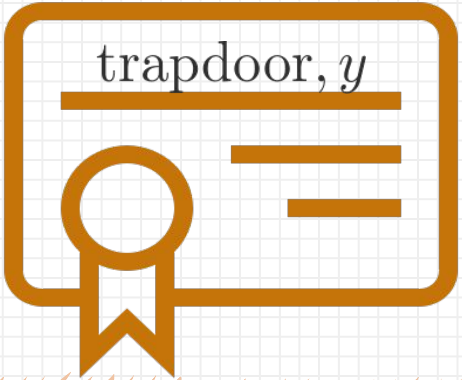
$(\mathsf{crs}, \mathsf{vk})$
trapdoor

Simulator

$(\mathsf{crs}, \mathsf{vk})$

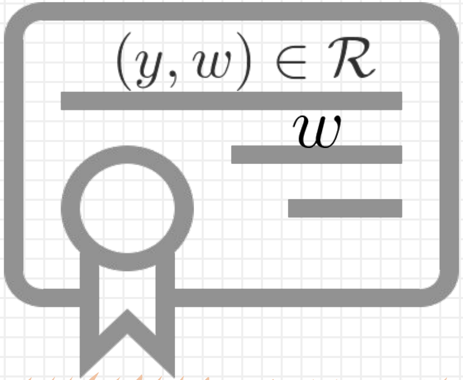
Prover

π'

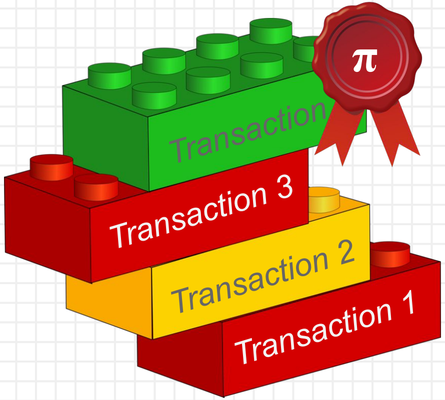


$\simeq$

π

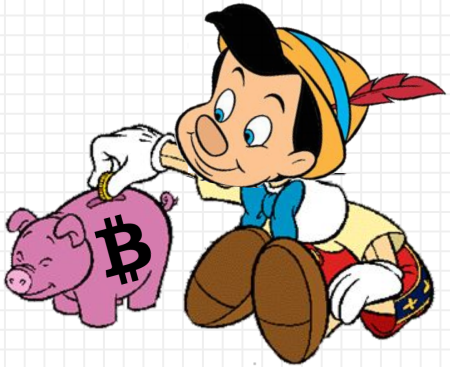


SNARKs in Confidential Transactions

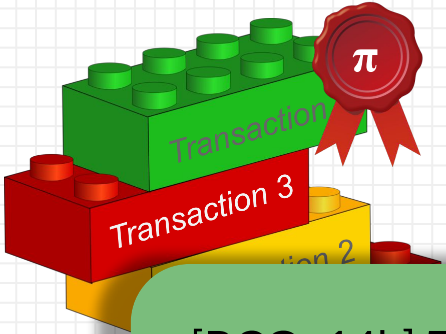


Key Properties for usage in Distributed Protocols

- **zero knowledge**
- **proof of knowledge**
- non-interactivity
- publicly verifiable
- succinctness

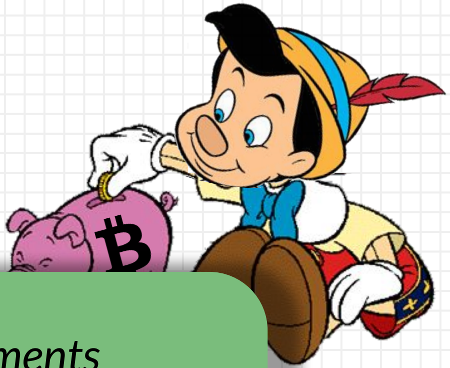


SNARKs in Confidential Transactions



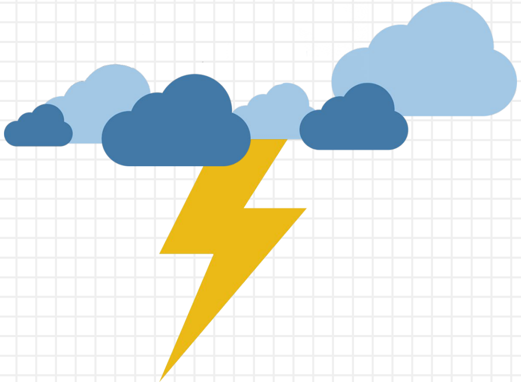
Key Properties for usage in Distributed Protocols

- zero knowledge
- proof of knowledge



[BCG+14b] *Zerocash: Decentralized anonymous payments from Bitcoin.*
E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer and M. Virza

Drawback Preprocessing: Trusted Setup



Pre-Processing:
(**crs**: common reference string)

- ✘ Secret coins
- ✘ Expensive
- ✘ Subversion

Efficient Protocols
from
Knowledge Assumptions

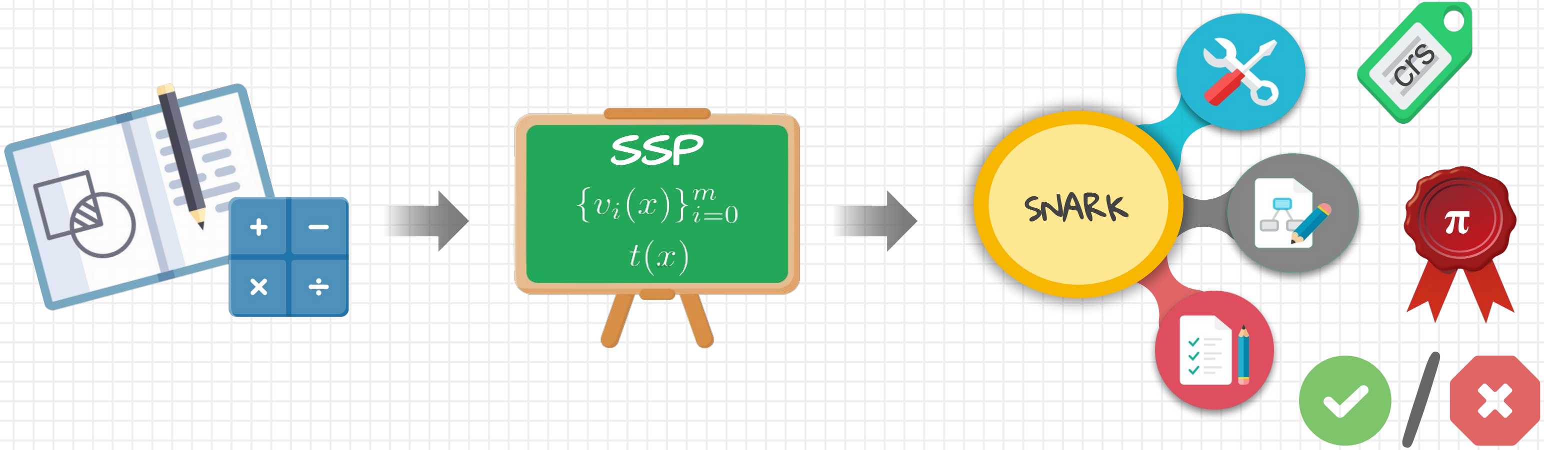
Frameworks

SNARK
Background

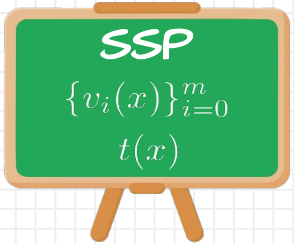
Frameworks
for SNARKs

Construction
Security

The END



SNARK: Methodology



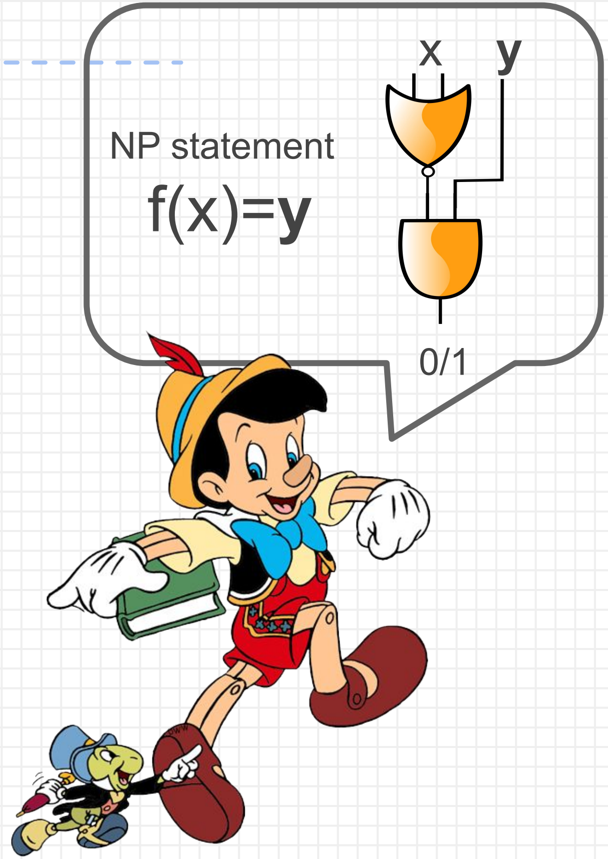
Target Statement R(y,w)=1 → **Computational Model (Representation)** → **SNARK under Knowledge Assumptions**

Computation y=F(x)	**PCP:** Probabilistically Checkable Proofs	**ECRH:** Extractable Collision-Resistant Hash
Boolean Circuit SAT	**QSP / SSP:** Quadratic / Square Span Programs	**PKE:** Power Knowledge of Exponent
Arithmetic Circuit SAT	**QAP / SAP:** Q / S Arithmetic Programs	**PKE:** Power Knowledge of Exponent

Computation: Circuit SAT

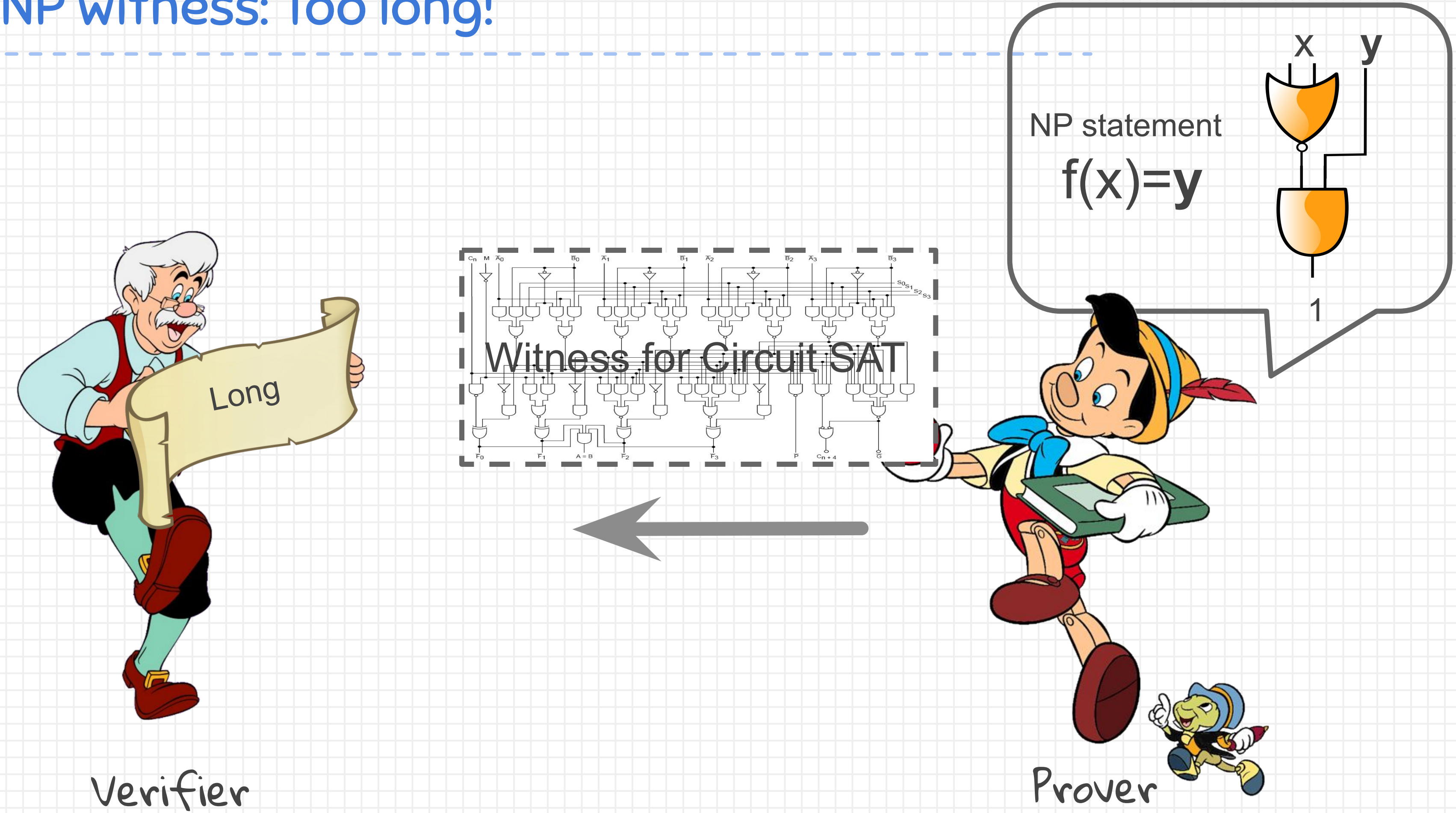
Verifier

Claim $f(x)=y$



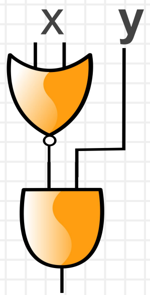
Prover

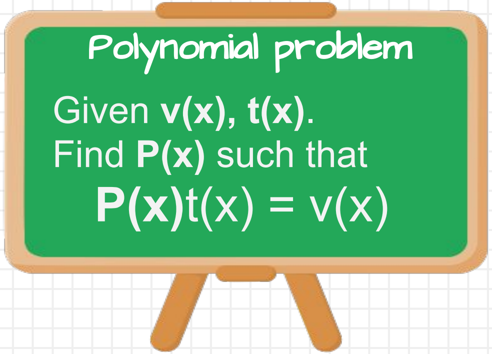
NP witness: Too long!



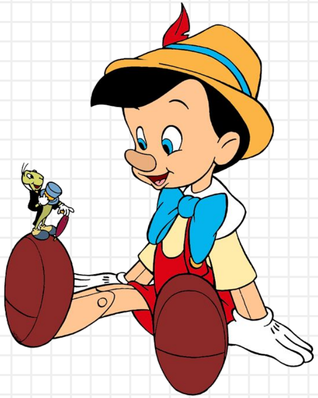
Solve equivalent problem instead

Circuit SAT solution





Verifier



Prover

Solve equivalent problem instead

Polynomial problem

Given $\mathbf{v(x)}$, $\mathbf{t(x)}$.
Find $\mathbf{P(x)}$ such that

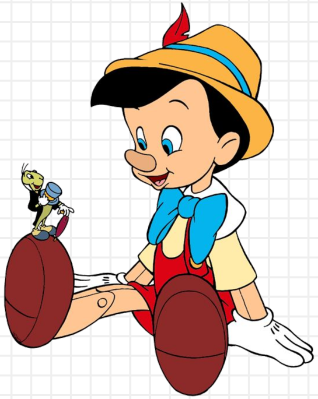
$$\mathbf{P(x)}t(x) = v(x)$$

$$P(x) = \sum p_i x^i$$

Verifier

Coefficients of solution P(x)

$\mathbf{p_0, p_1, p_2, \ldots\ p_d}$

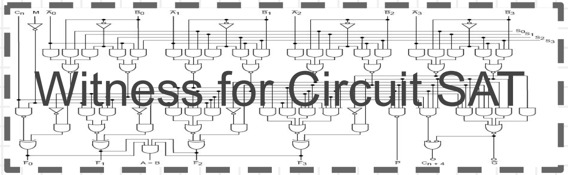


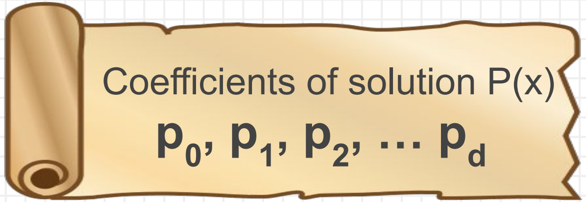
Prover

Solution as big as witness for Circuit SAT

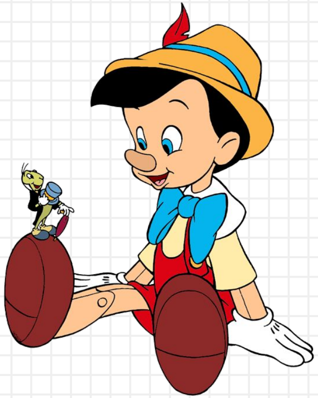
Not Succinct

$$P(x) = \sum p_i x^i$$



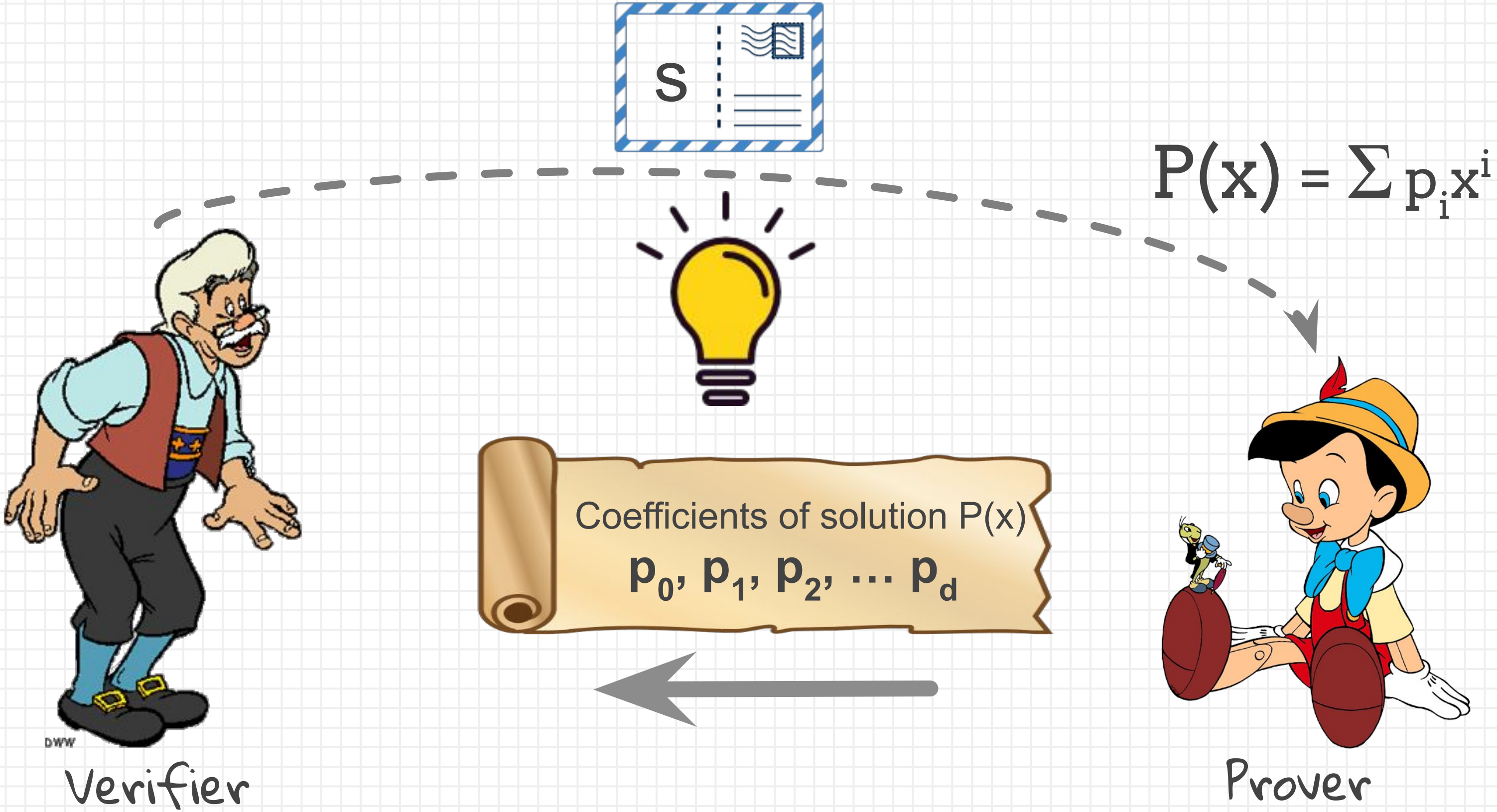


Verifier

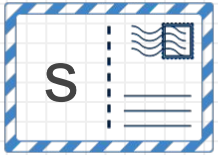


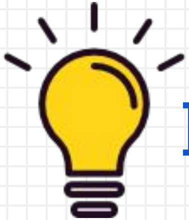
Prover

Evaluate polynomial in one point s



Evaluate polynomial in one point s

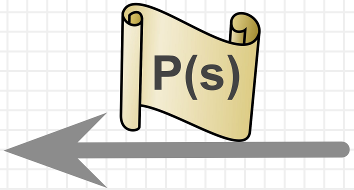


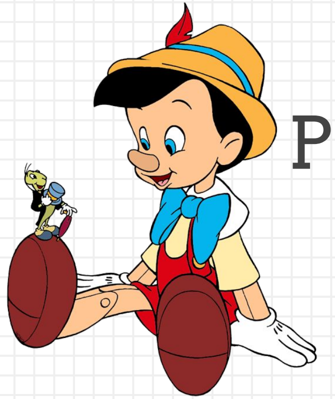


$P(s) = \sum p_i s^i$

$\mathbf{P(x)}t(x) = v(x) \rightarrow \mathbf{P(s)}t(s) = v(s)$

Verifier

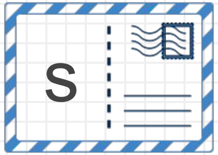




$P(x)$

Prover

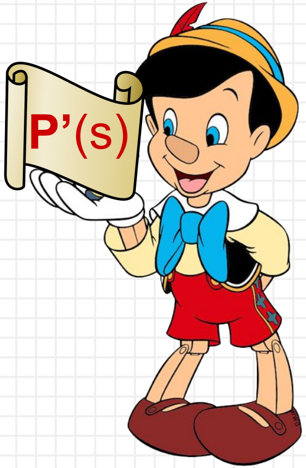
The evaluation point should be **hidden**



$$P'(x) \neq P(x)$$

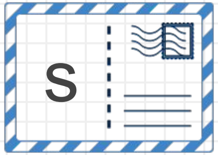
$$P'(s) = P(s)$$

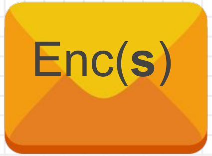
Verifier



Prover

The evaluation point should be **hidden**

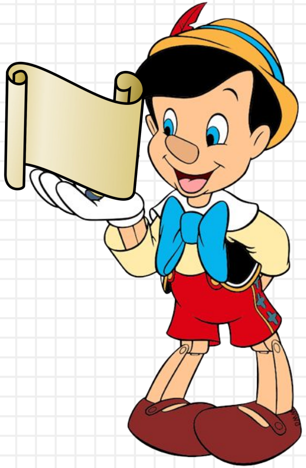




Solves

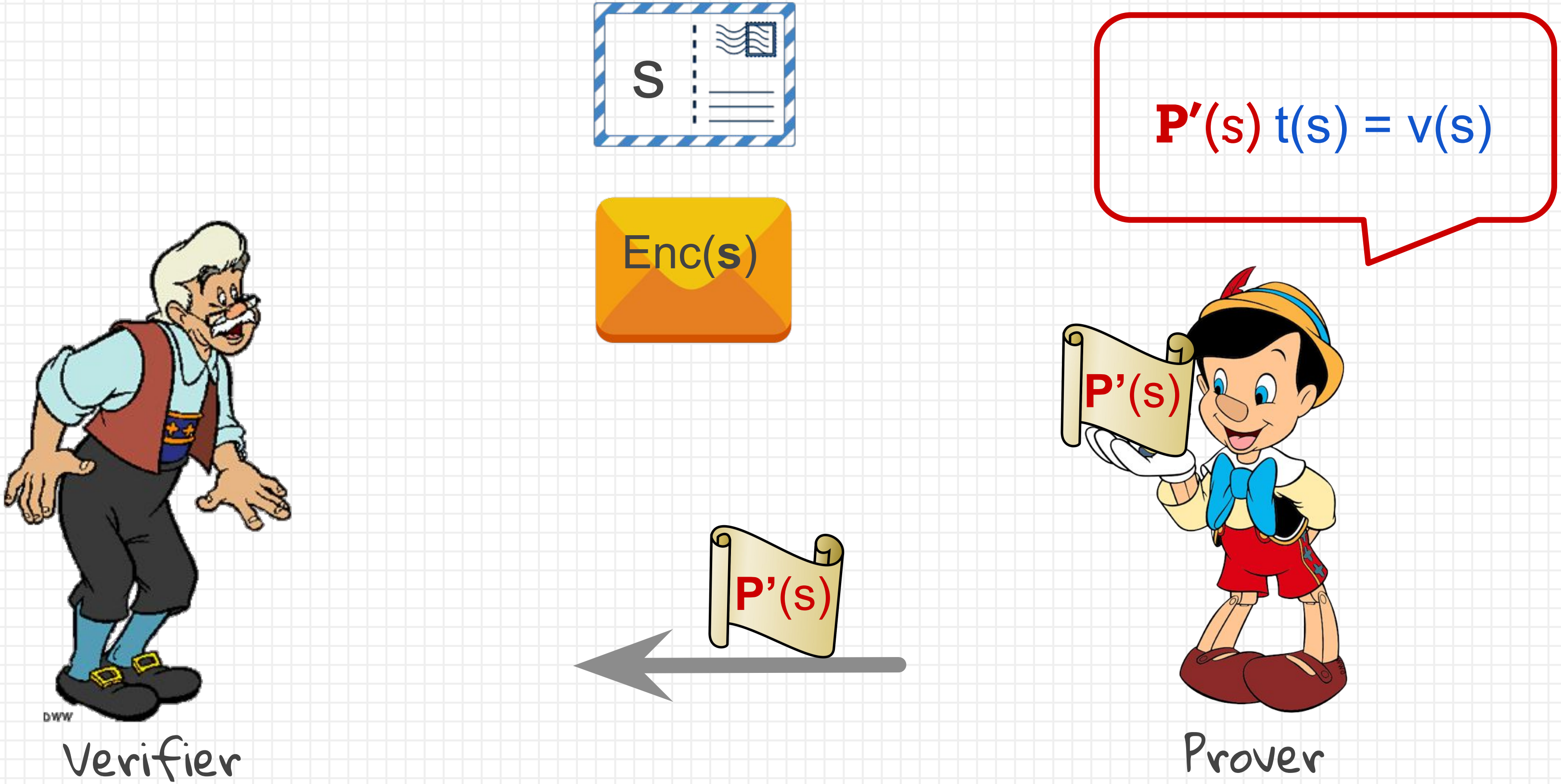
$t(s) = v(s)$

Verifier

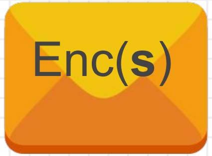


Prover

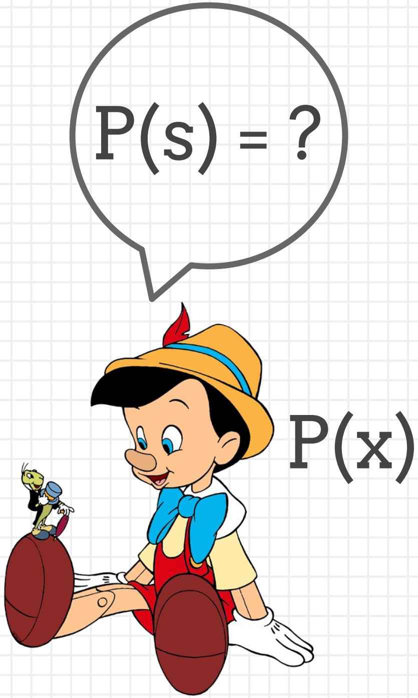
The evaluation point should be **hidden**



Encoding of evaluation point s – not enough!

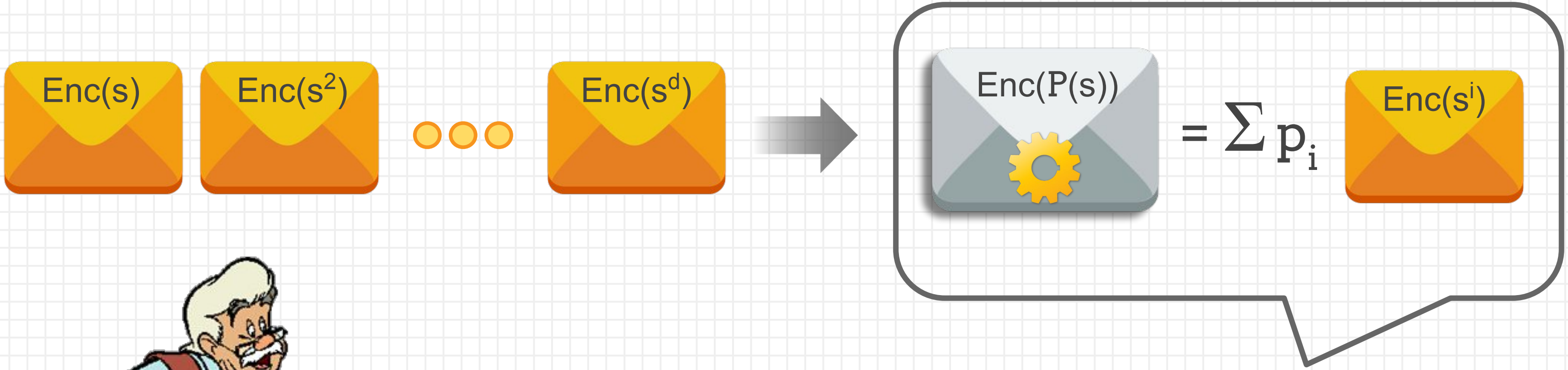


Verifier



Prover

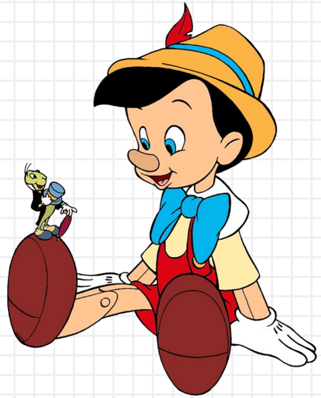
Encoding Properties



Verifier

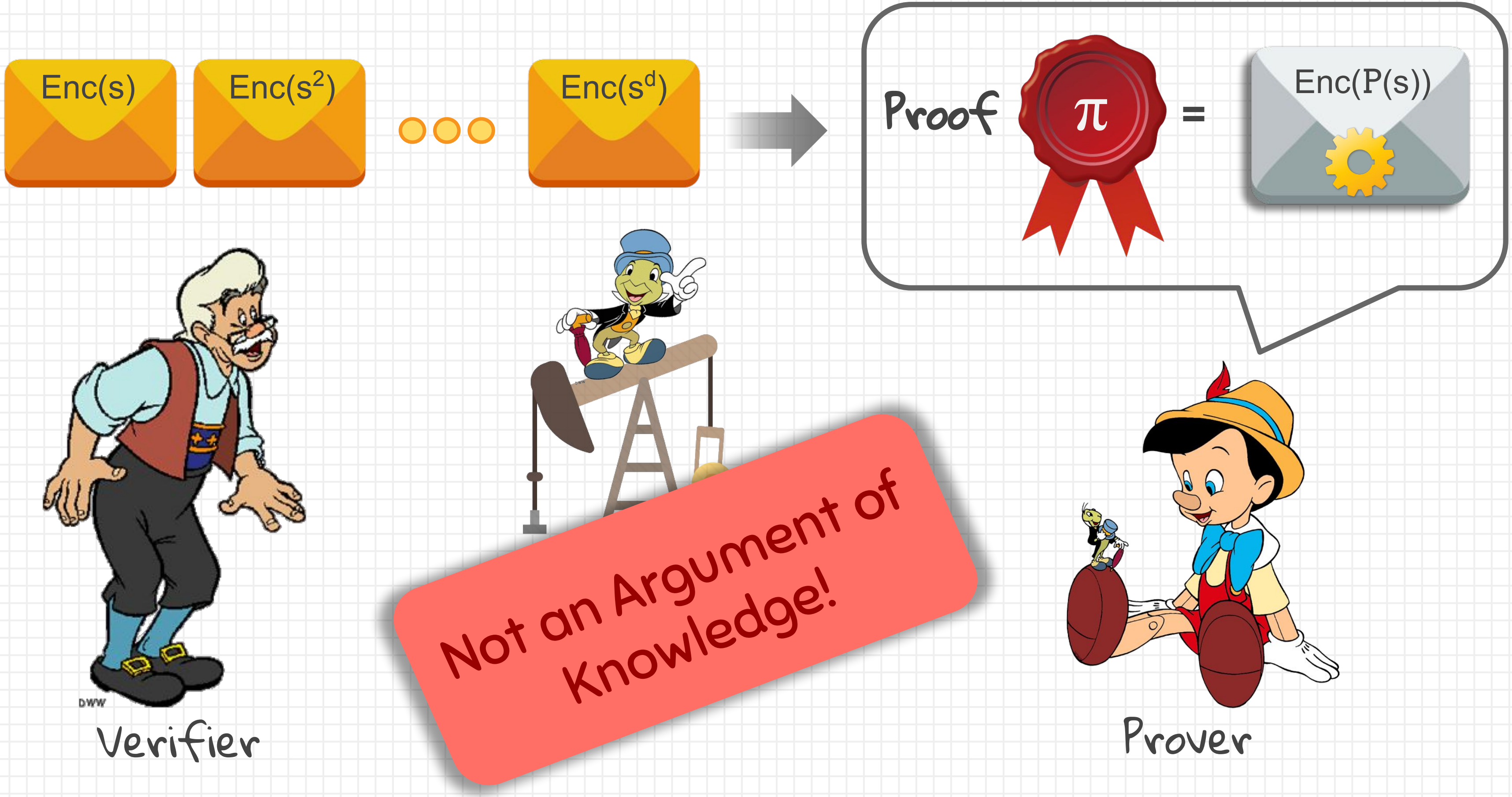
Encoding:

- ***linearly*** homomorphic

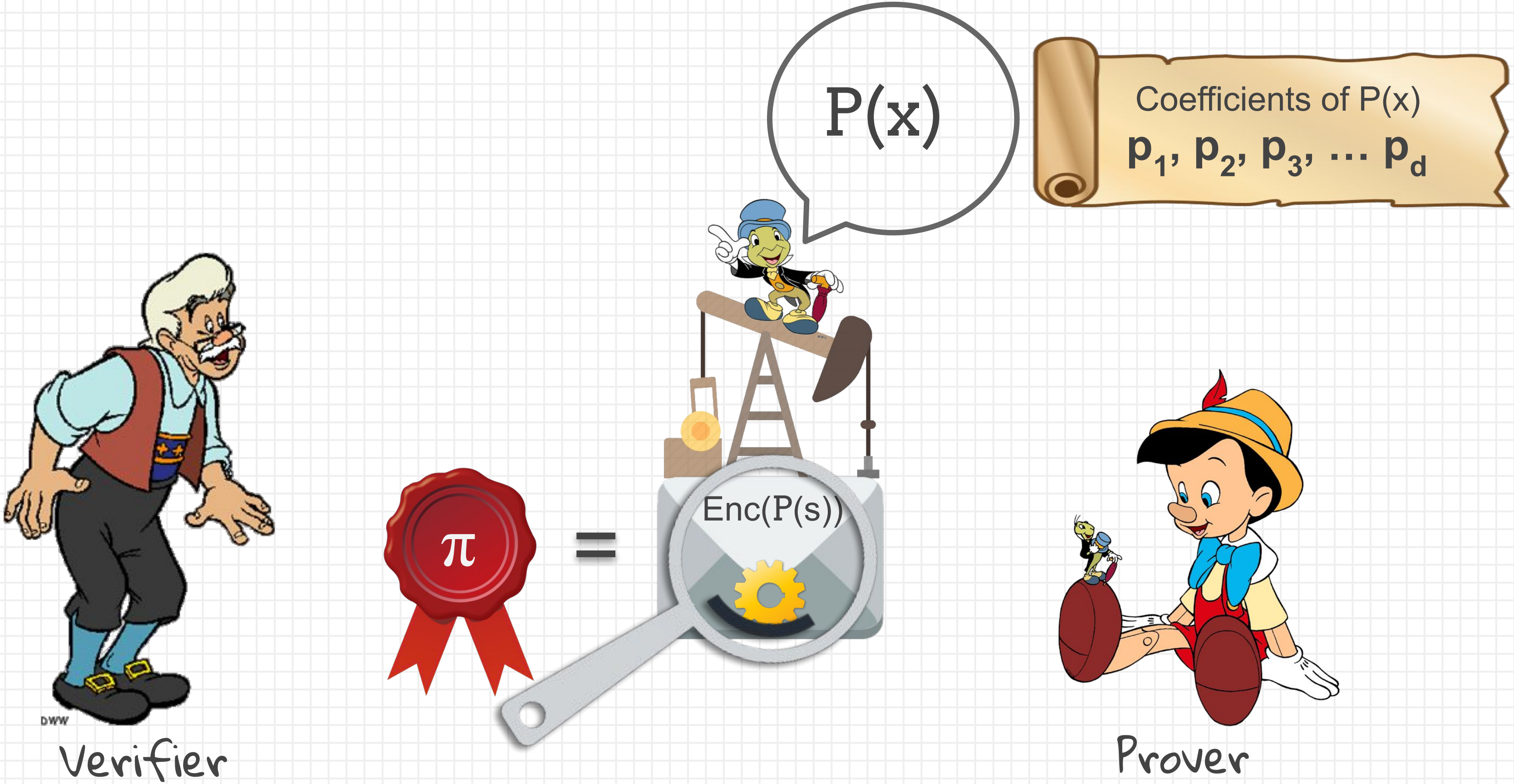


Prover

Not Knowledge Sound



Extraction from the proof



Non-falsifiable Assumption: Power Knowledge of Exponent

d-PKE

$Enc(P)$ $Enc(\alpha P)$

$Enc(s)$ $Enc(s^2)$ … $Enc(s^d)$

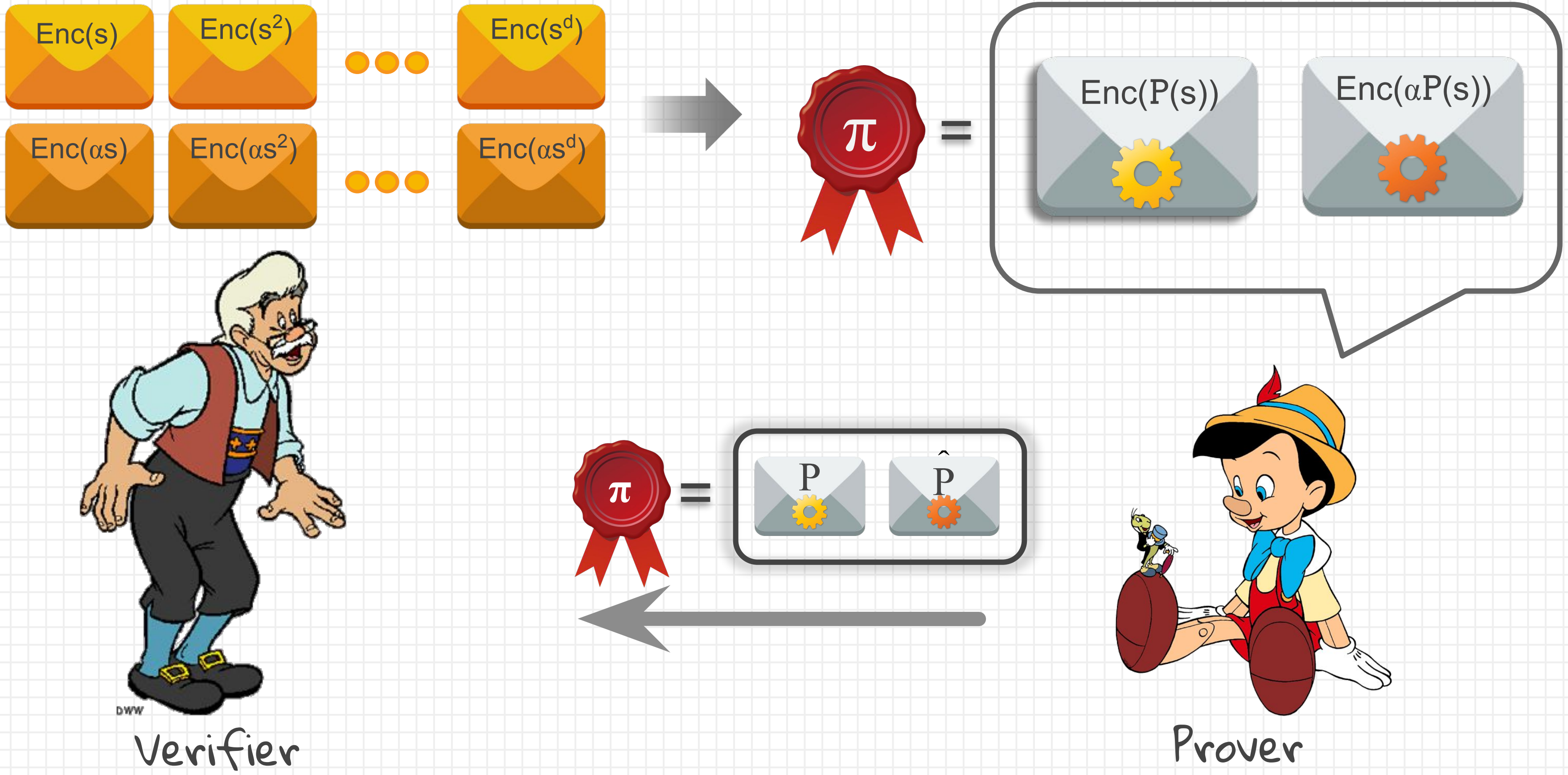
$Enc(\alpha s)$ $Enc(\alpha s^2)$ … $Enc(\alpha s^d)$

$\mathcal{E}$

p_1 p_2 … p_d

$Enc(P) = Enc(\sum p_i s^i)$

Preprocessing: Double the Proof

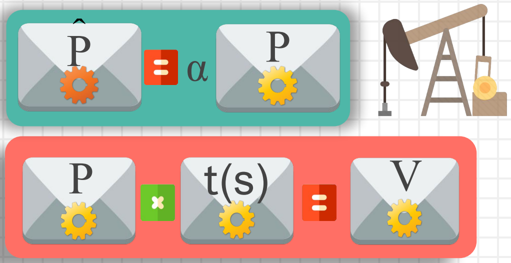


Verification

Polynomial problem

Given **v(x), t(x)**.
Find **P(x)** such that

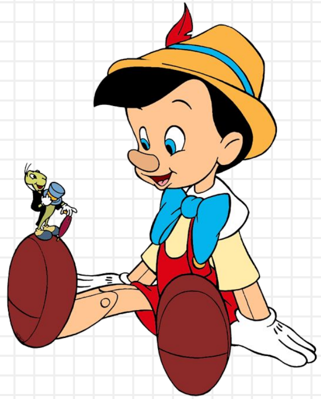
$$\mathbf{P(x)}t(x) = v(x)$$



Encoding:

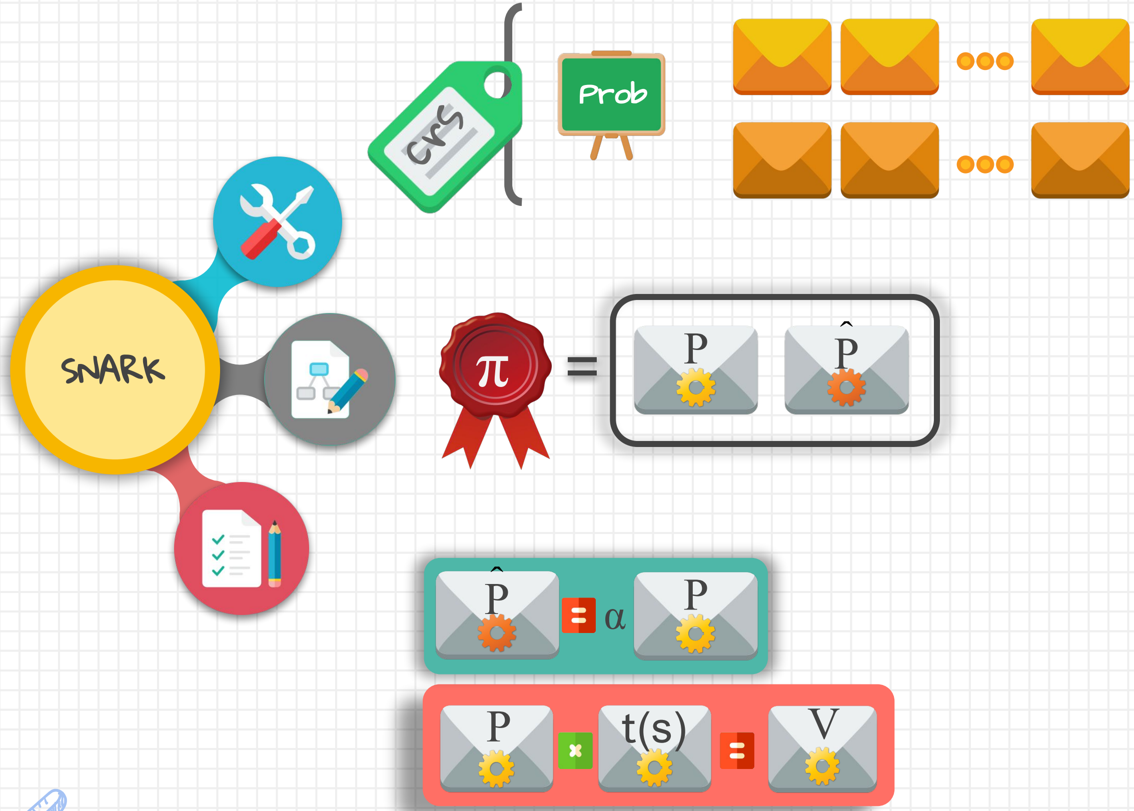
- *linearly* homomorphic
- quadratic root detection
- image verification

Verifier



Prover

Review of the Protocol (Algorithms)



Security of our SNARK

SNARK
Background

Framework
for SNARKs

Construction
Security

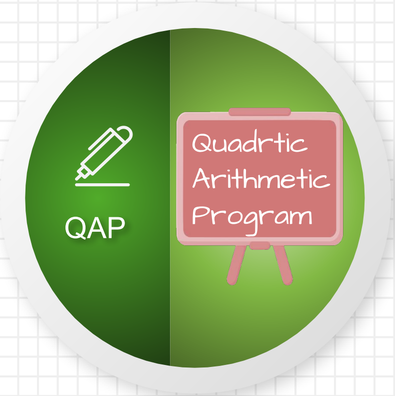
The
END

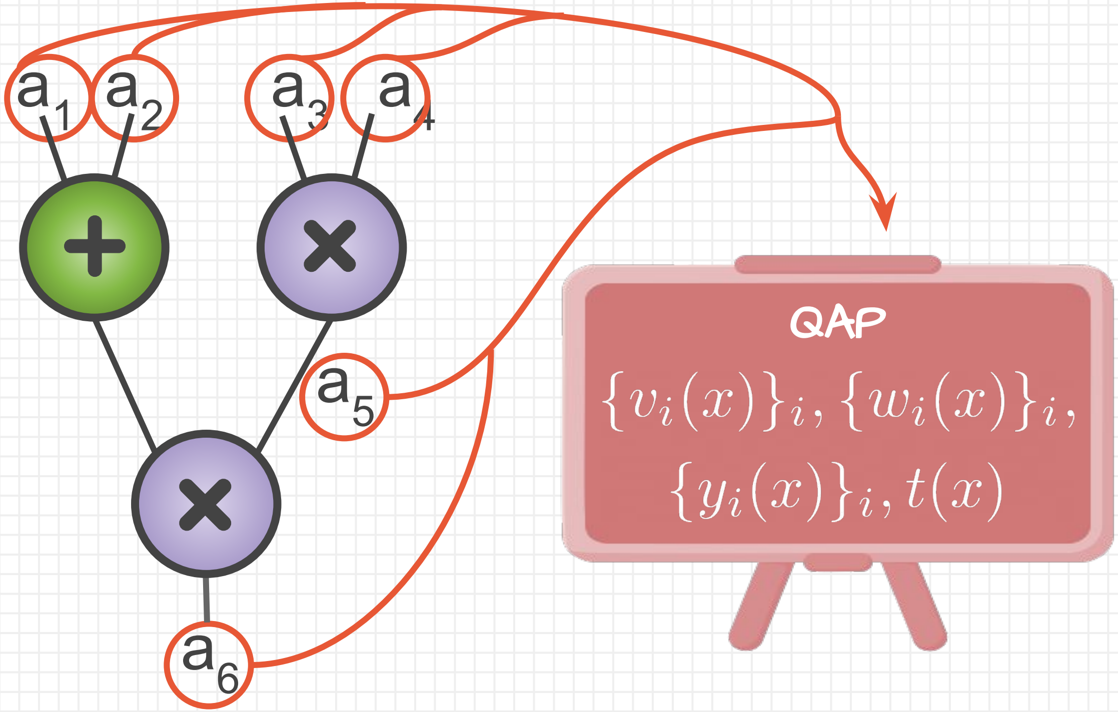
NP Representations

Computational Models

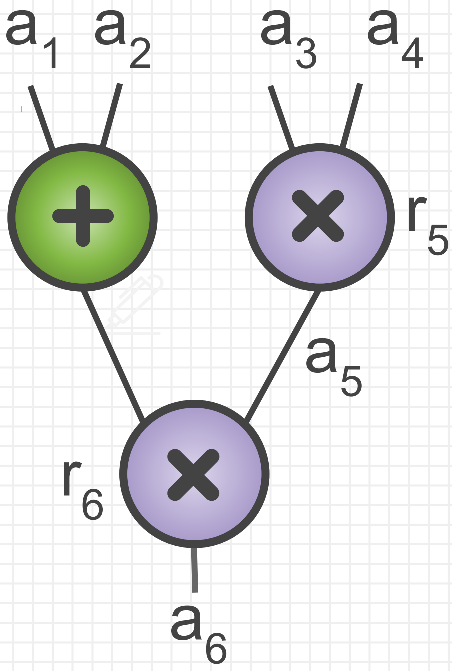
For SNARK

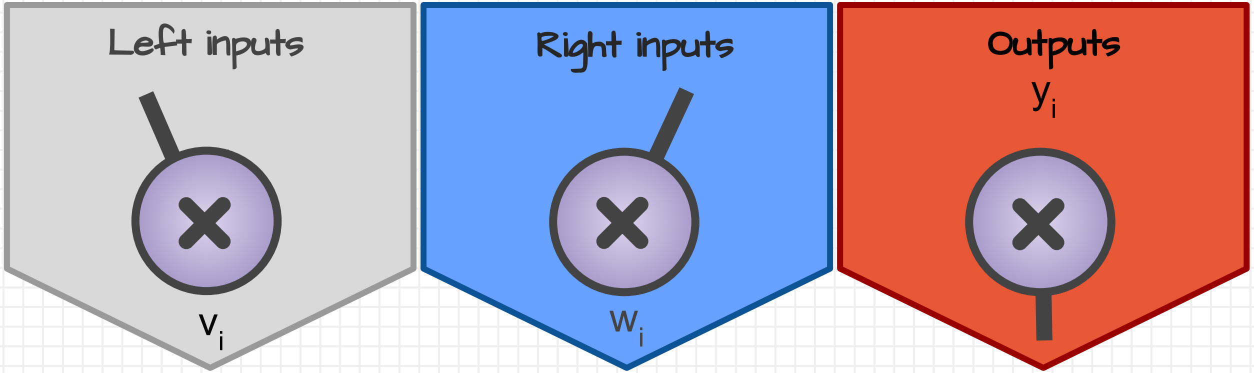
Quadratic Arithmetic Programs





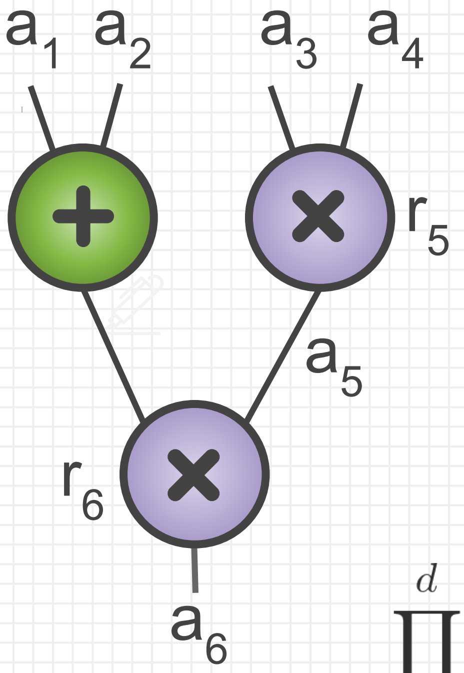
Build a table to interpolate polynomials

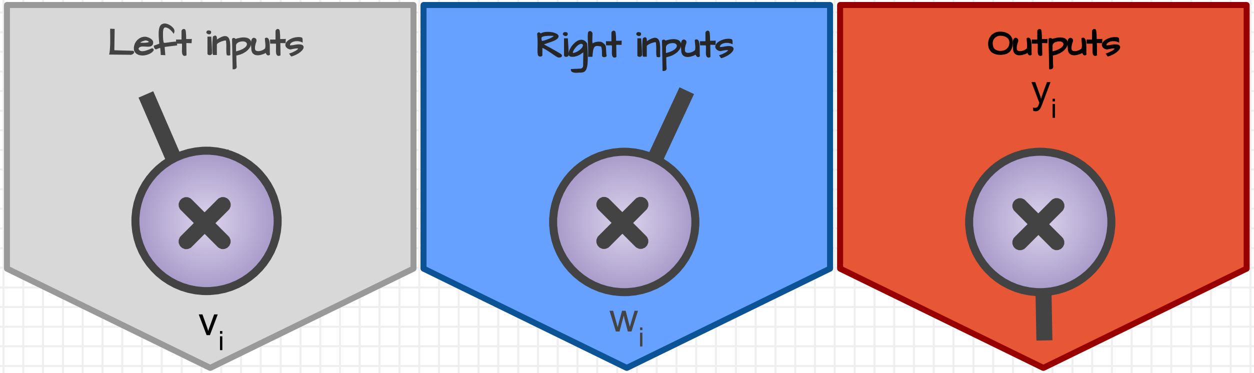




$v_3(r_5) = 1$	$w_4(r_5) = 1$	$y_5(r_5) = 1$
$v_1(r_6) = v_2(r_6) = 1$	$w_5(r_6) = 1$	$y_6(r_6) = 1$
$v_i(r_5) = 0, \ i \neq 3$	$w_i(r_5) = 0, \ i \neq 4$	$y_i(r_5) = 0, \ i \neq 5$
$v_i(r_6) = 0, \ i \neq 1,2$	$w_i(r_6) = 0, \ i \neq 5$	$y_i(r_6) = 0, \ i \neq 6$

Division property: Common Roots r_5, r_6

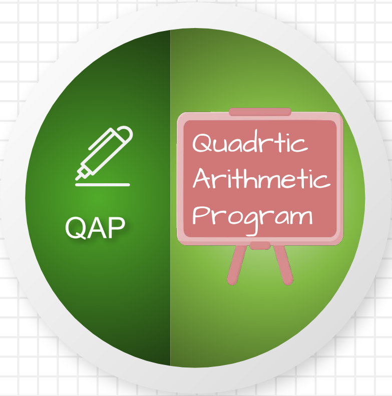




$v_3(r_5) = 1$	$w_4(r_5) = 1$	$y_5(r_5) = 1$
$v_1(r_6) = v_2(r_6) = 1$	$w_5(r_6) = 1$	$y_6(r_6) = 1$

$$\prod_{j=1}^{d}(x - r_j) \,\Big|\, \left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) - \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

Quadratic Arithmetic Program



QAP

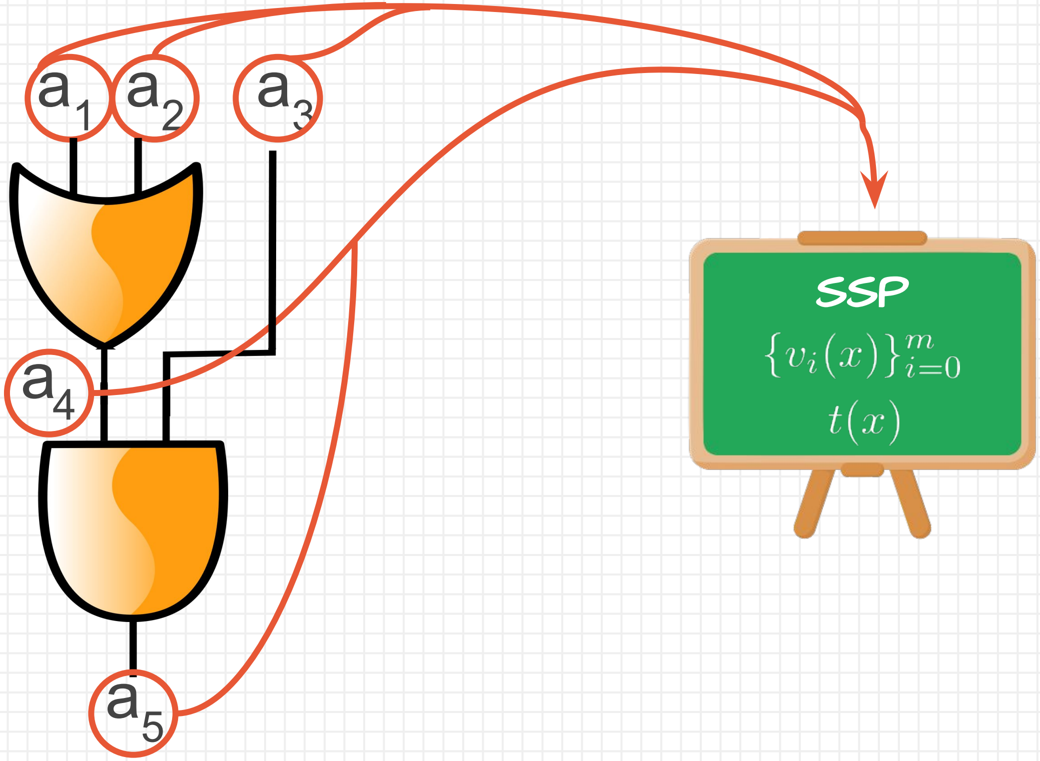
Given $\{v_i(x)\}_i, \{w_i(x)\}_i, \{y_i(x)\}_i, t(x)$

Find $V(x), W(x), Y(x), h(x)$ s.t.

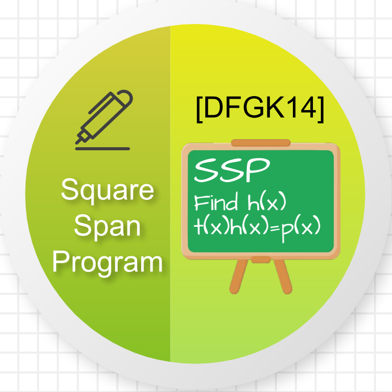
$$V(x) = \sum_{i=0}^{m} a_i v_i(x) \quad \ldots$$

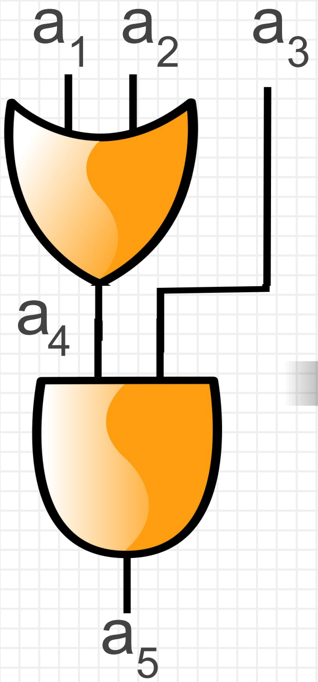
and $t(x)h(x) = V(x)W(x) - Y(x)$

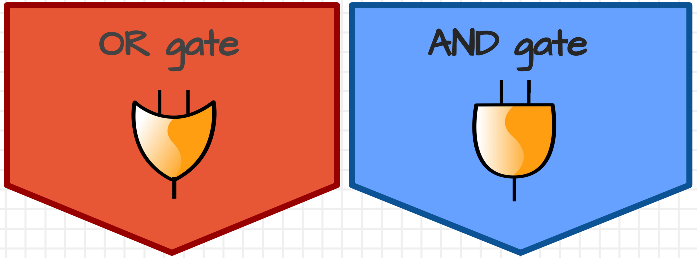
Square Span Programs



Step 1: Linearization of logic gates





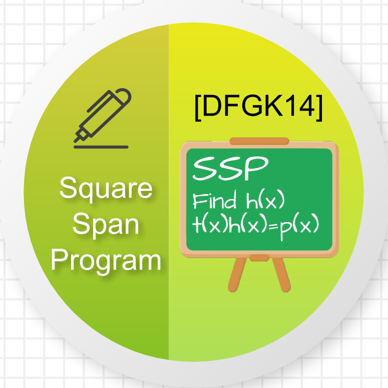


a_1	a_2	a_4	a_3	a_4	a_5
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	1	1	1	1
$-2a_1-2a_2+4a_4 \in \{0,2\}$			$2a_3+2a_4-4a_5 \in \{0,2\}$		

$$L(a_i) = v_{0j} + \sum_{i=1}^{m} a_i v_{ij} \in \{0,2\}$$

Step 2. Square constraint

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

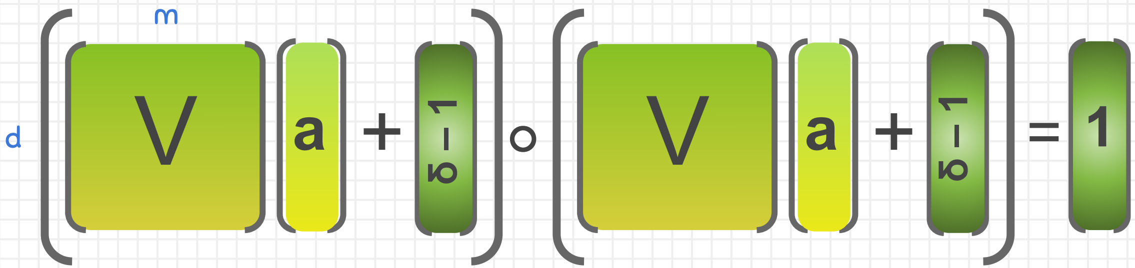


$$\alpha a + \beta b + \gamma c + \delta \in \{0,2\}$$

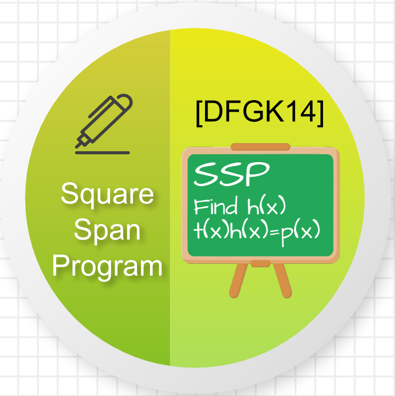
$$\alpha a + \beta b + \gamma c + \delta - 1 \in \{-1,1\}$$

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

Step 3. Polynomial Interpolation



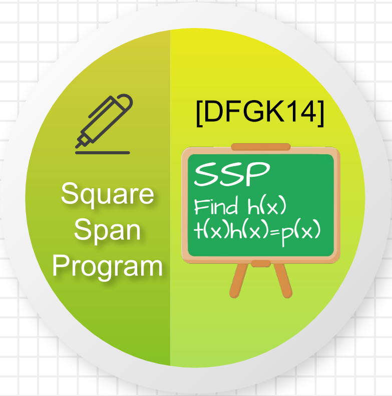
$\forall\{r_j\} \in \mathbb{F}^d$



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

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

Step 4. Polynomial Problem SSP



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

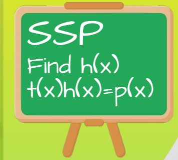
$$\Big\Downarrow \; \forall \{r_j\} \in \mathbb{F}^d$$

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

Polynomial Problem SSP

Square Span Program

[DFGK14]



SSP

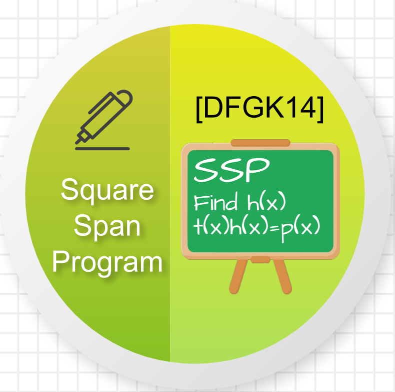
For $\{v_i(x)\}_{i=1,m},\ t(x) \in \mathbb{F}[x]$

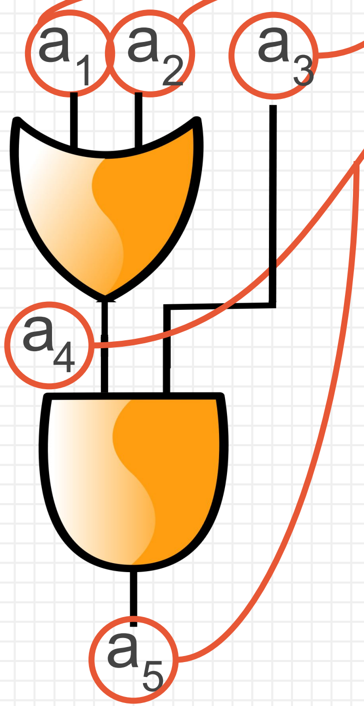
$$t(x) \mid V(x)^2 - 1$$

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

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

Polynomial Problem SSP





SSP

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

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

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

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

Encodings

Properties

Assumptions

Encodings Instantiations: Linearity

DLog Group $\mathbb{G}$

$$\langle g \rangle = \mathbb{G}, \;\; Enc(s) = g^s$$

$$Enc(p(s)) = g^{p(s)}$$

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

DLog Encoding vs Encryption Scheme

DLog Group $\mathbb{G}$

$$\langle g \rangle = \mathbb{G}, \;\; Enc(s) = g^s$$

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

$$Enc(p(s)) = g^{p(s)}$$

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

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

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

$E_{pk}(s)$ $\quad$ $E_{pk}(s^2)$ $\quad \cdots \quad$ $E_{pk}(s^d)$

$$Enc(p(s)) = E_{pk}(p(s))$$

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

Quadratic Root Detection – Pairings

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

$$Enc(s) = g^s \qquad e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$

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

Quadratic root detection **public**

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

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

Publicly Verifiable vs Designated Verifiable

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

$$Enc(s) = g^s \qquad e : \mathbb{G} \times \mathbb{G} \to \tilde{\mathbb{G}}$$

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

Quadratic root detection **public**

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

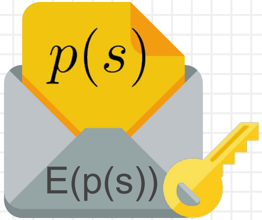
$$e(g^{t(s)}, g^{h(s)}) \stackrel{?}{=} e(g^{p(s)}, g)$$

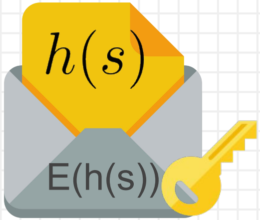
Encryption: $E_{pk}(m) = c$

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

Quadratic root detection needs **sk**

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





Assumption on Discrete Log Encoding

d-PKE

g^{s} g^{s^2} ... g^{s^d}

$g^{s\alpha}$ $g^{\alpha s^2}$ ... $g^{\alpha s^d}$

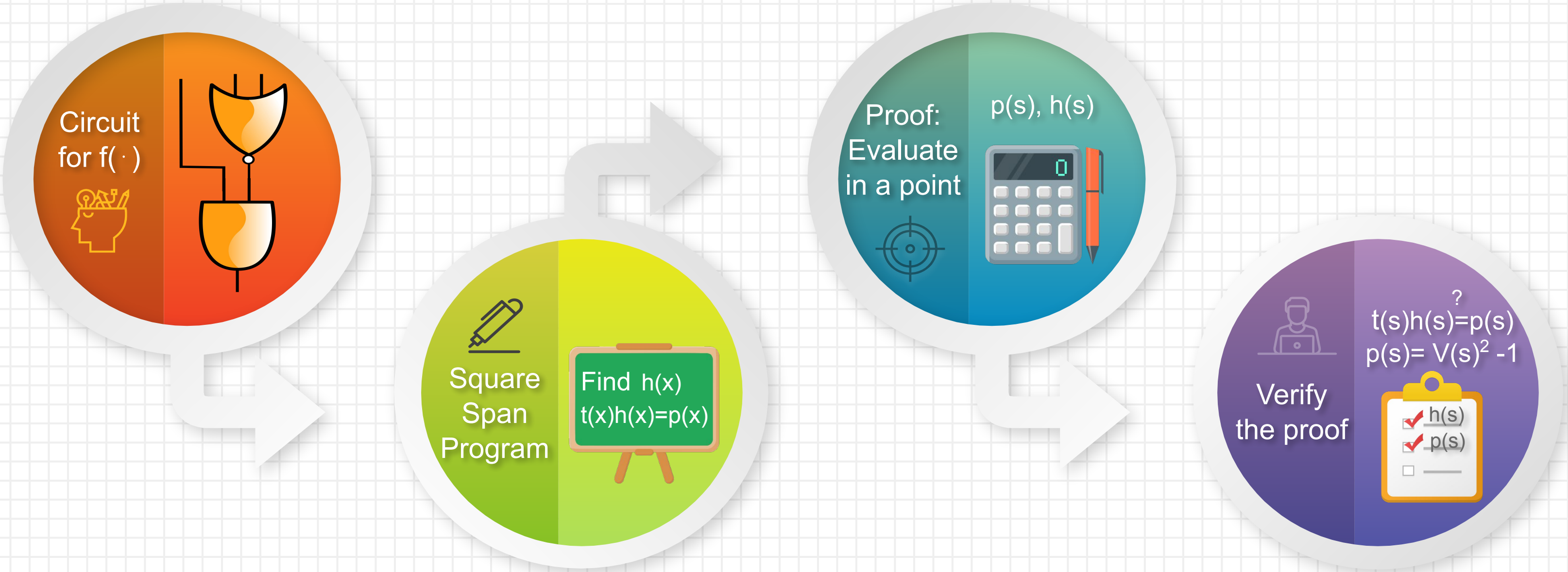
g^{P} $g^{\alpha P}$

$\mathcal{E}$

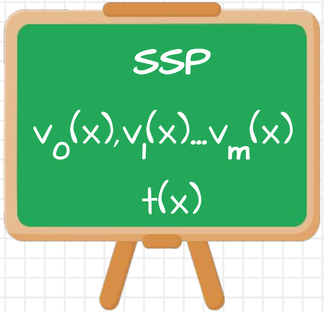
g^{p_1} g^{p_2} ... g^{p_d}

$g^{P} = g^{\Sigma p_i s^i}$

SNARK from SSP



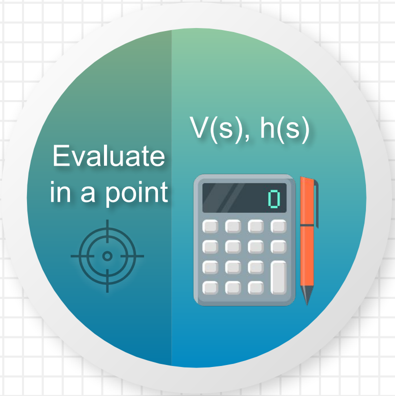
Evaluate solution in s

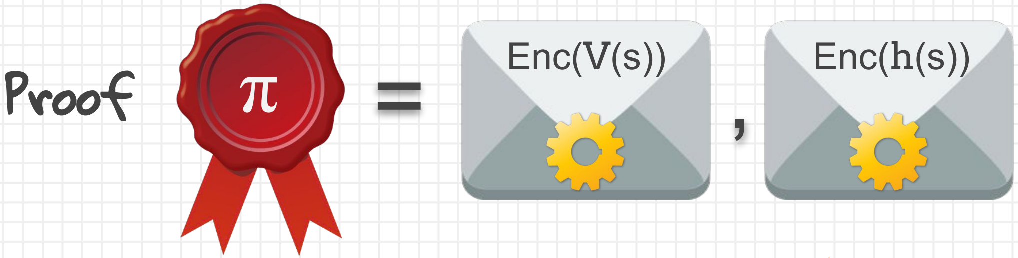


$$V(x), h(x) = ?$$

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

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





Enforce Linear Span

SSP

$v_0(x), v_1(x) \ldots v_m(x)$

$t(x)$

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

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

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

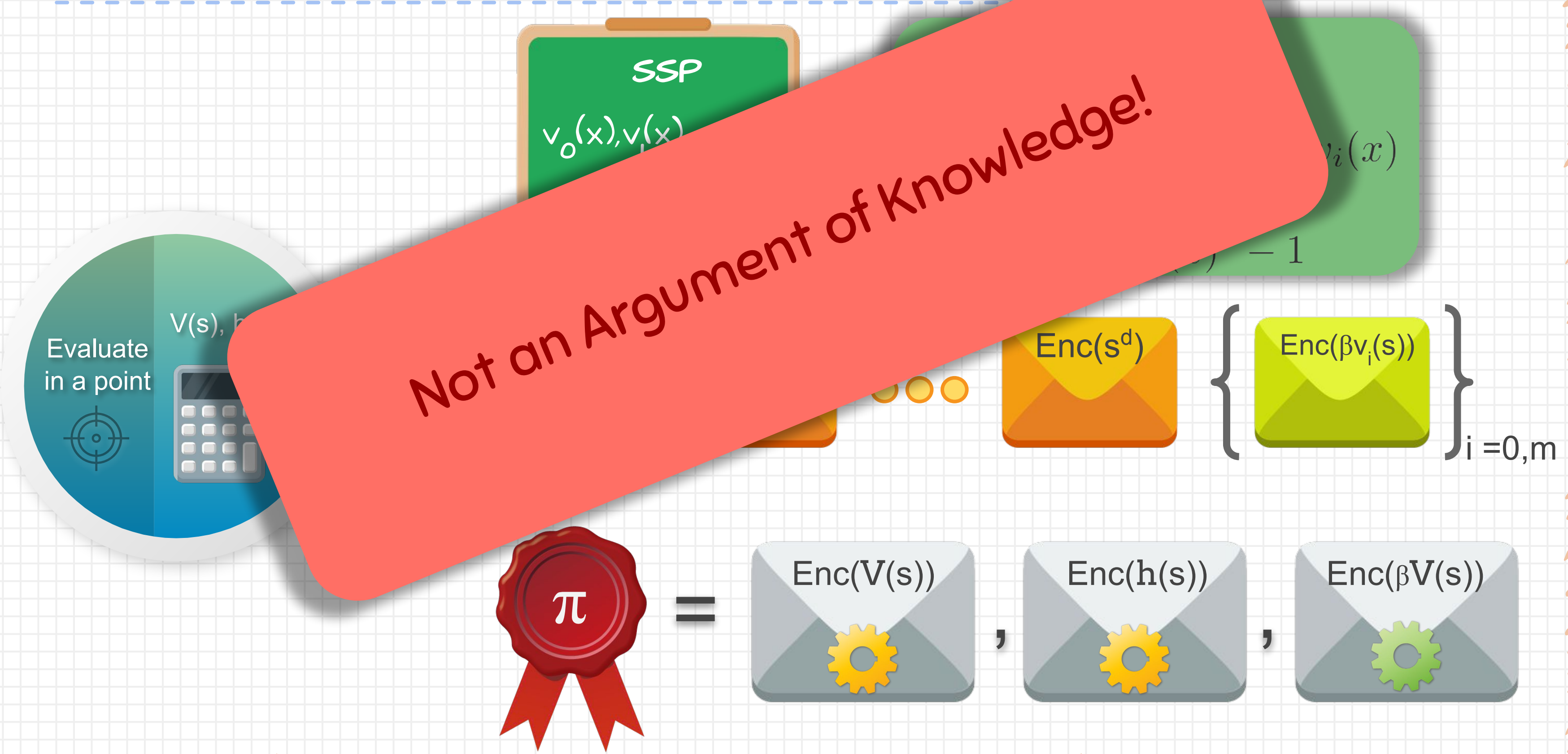
Evaluate in a point

V(s), h(s)

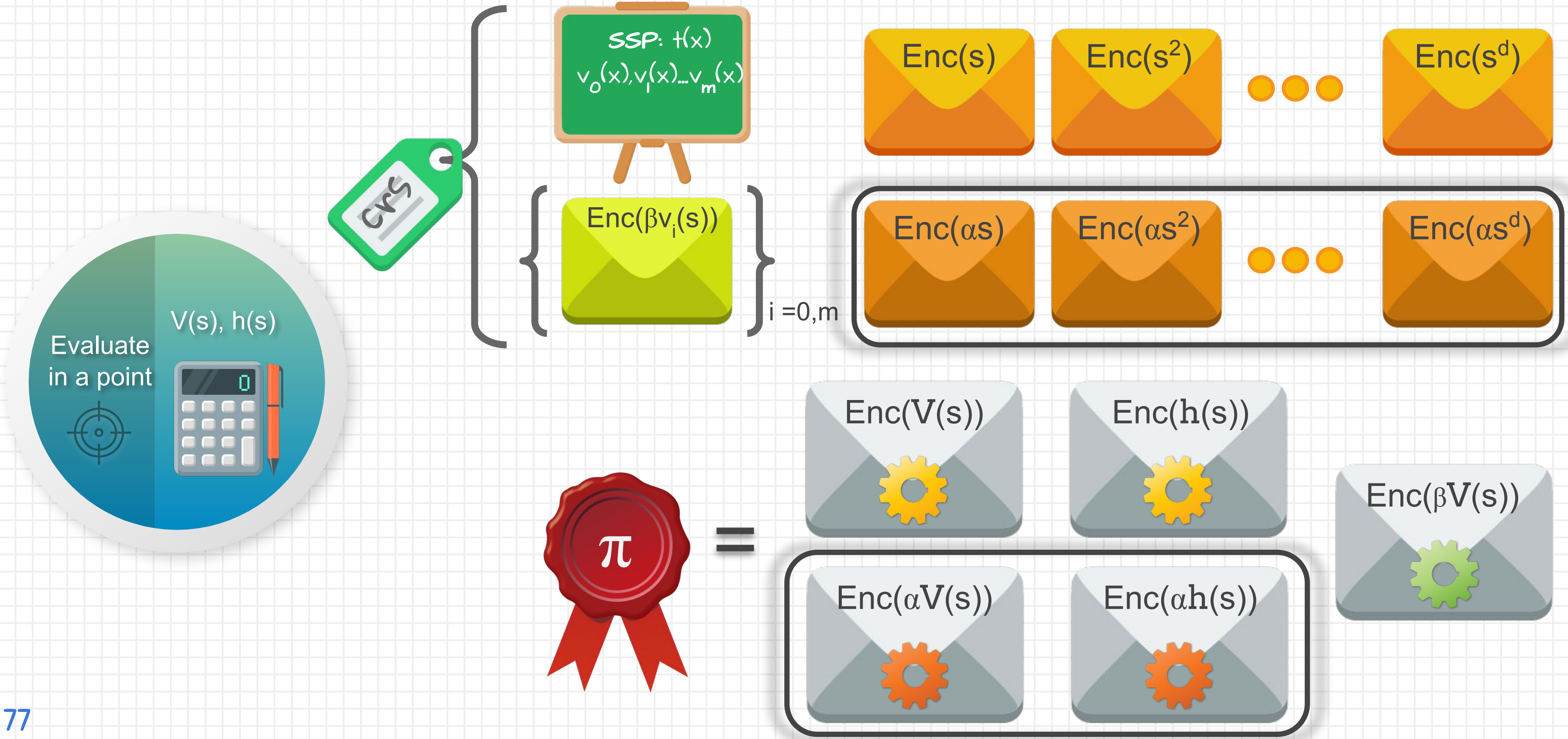
Enc(s) Enc(s^2) ... Enc(s^d) $\{$Enc($\beta v_i(s)$)$\}_{i=0,m}$

π = Enc(V(s)), Enc(h(s)), Enc(βV(s))

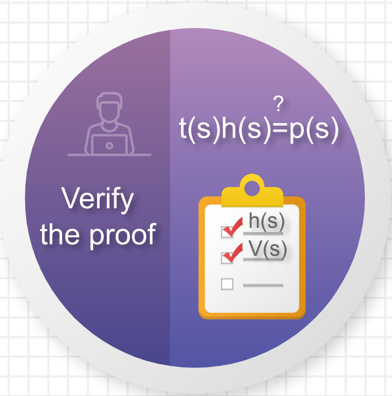
Extractability?

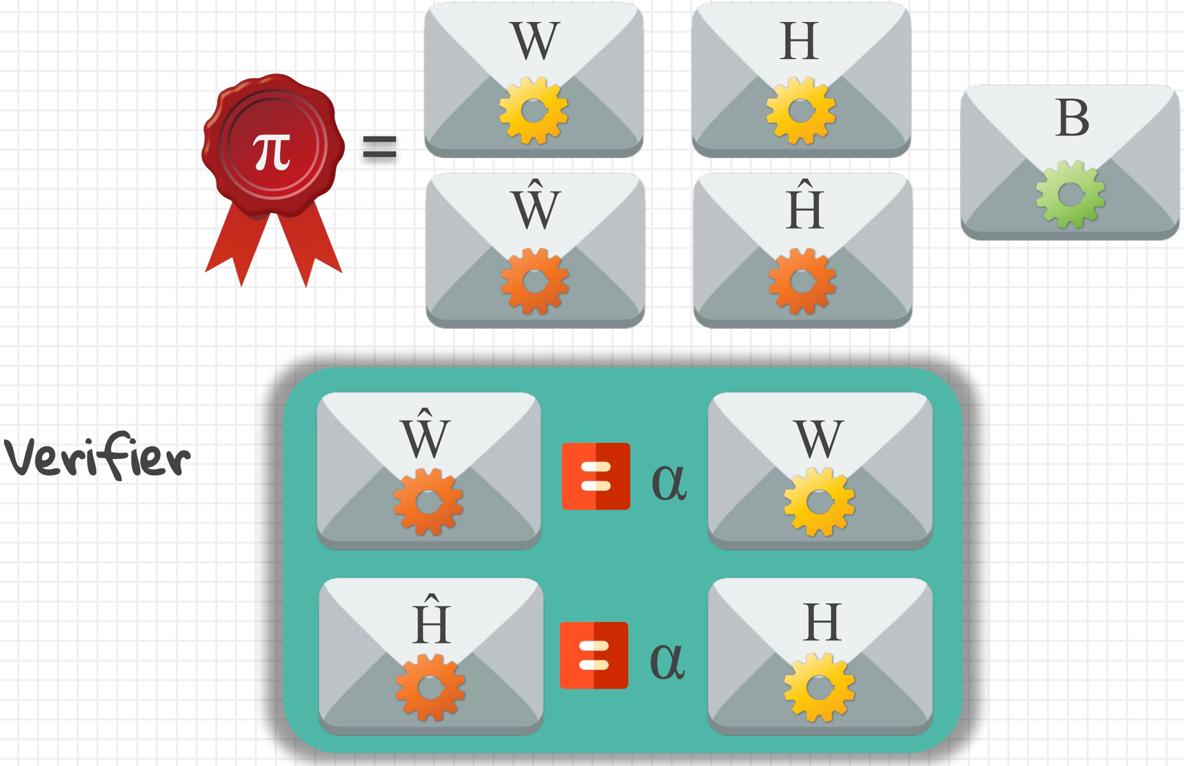


Setup and Proof

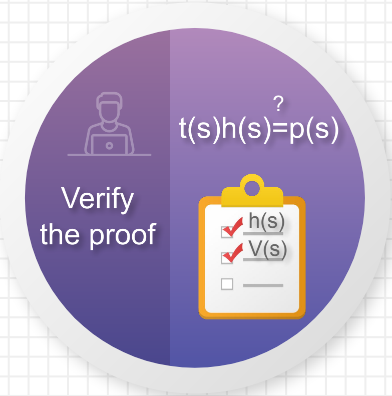


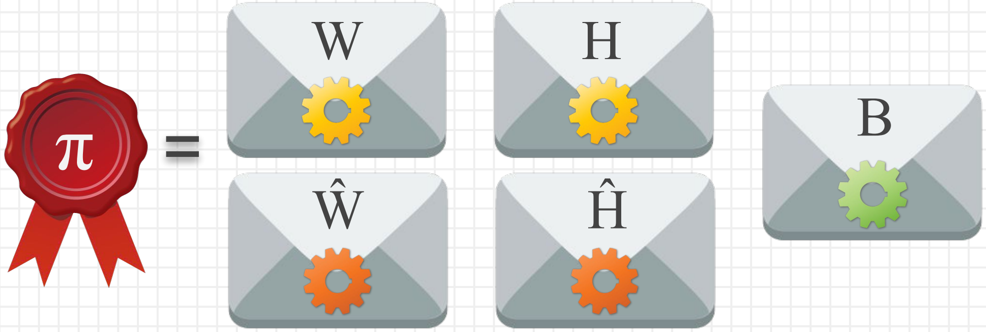
Verifier



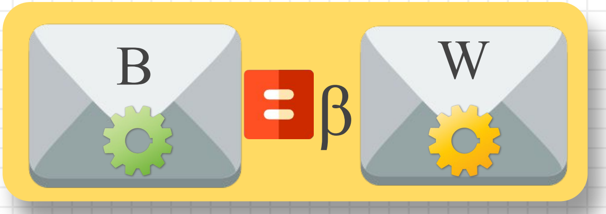


Verifier

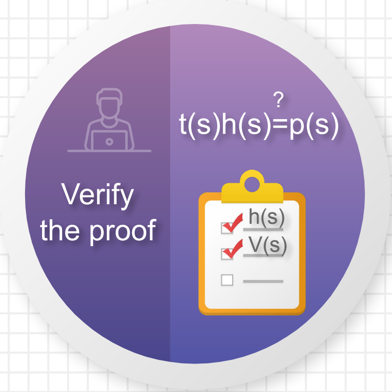


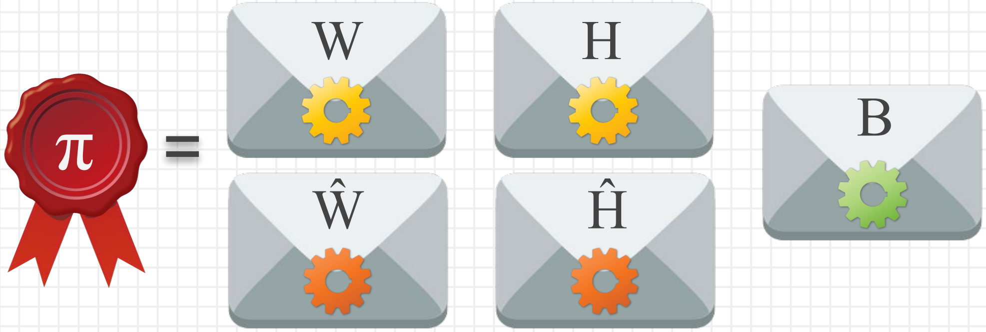


Verifier



Verifier





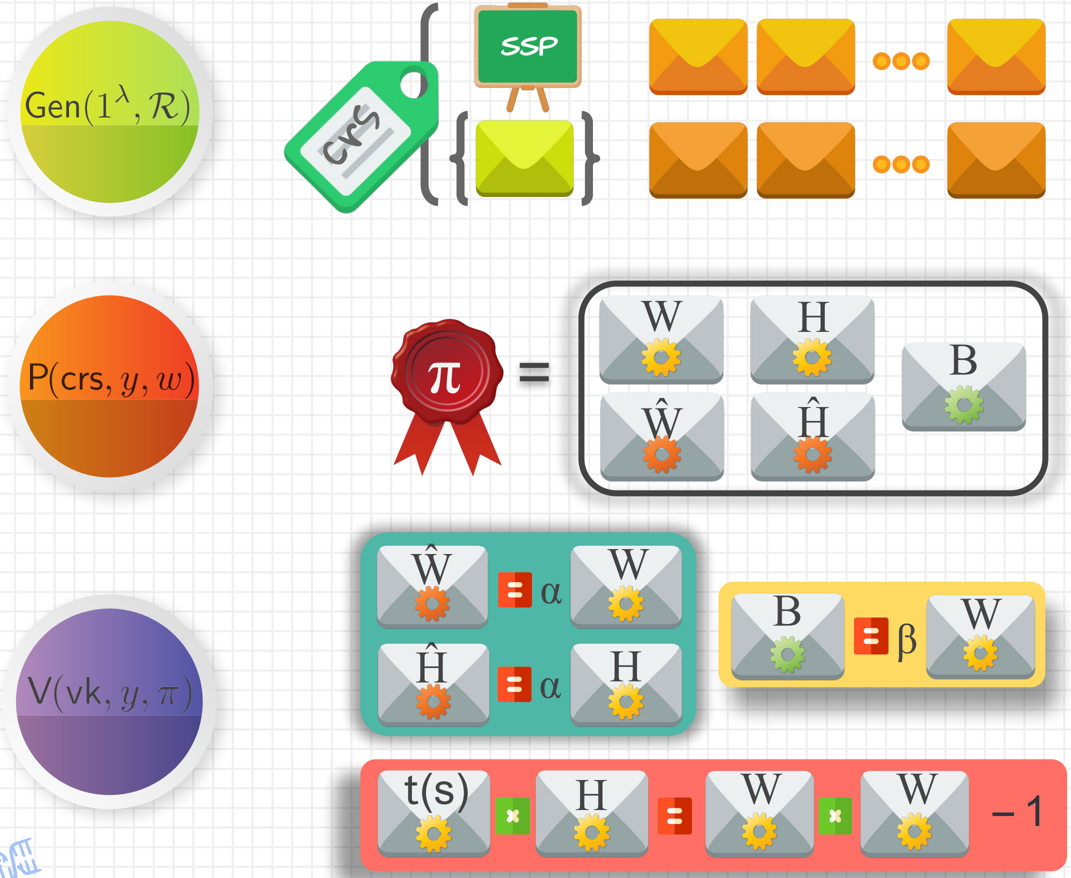
Verifier

Adding Zero-Knowledge

- ✘ randomize polynomial V(x) to hide witness

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

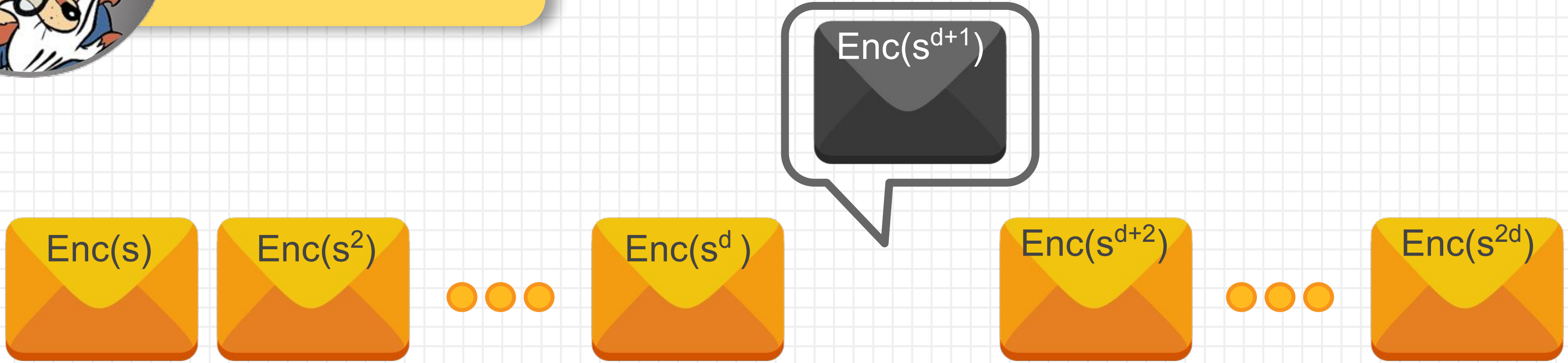
Review of the Protocol (Algorithms)



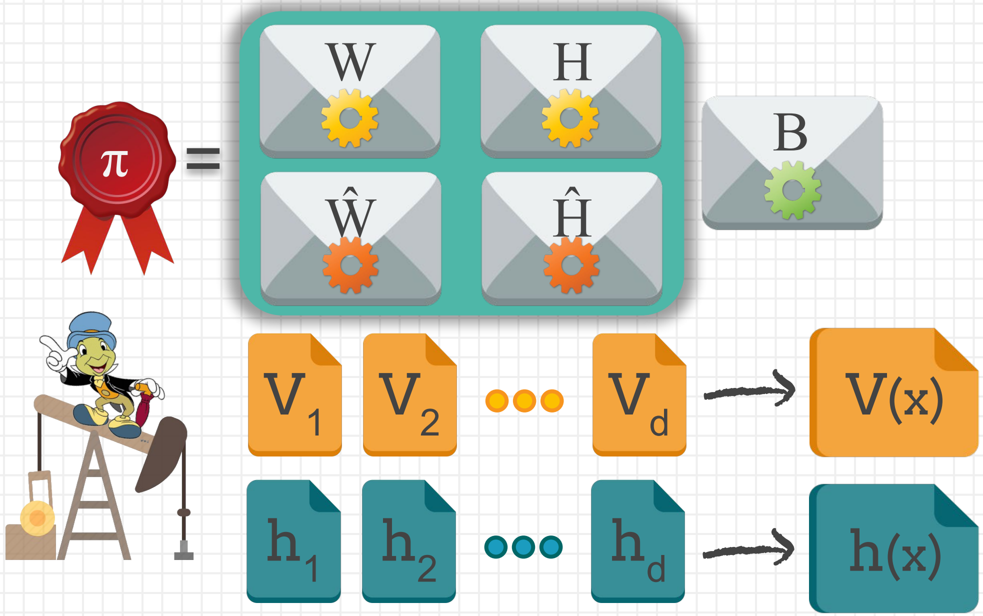
Security Analysis

Assumption PDH: Power Diffie-Hellman

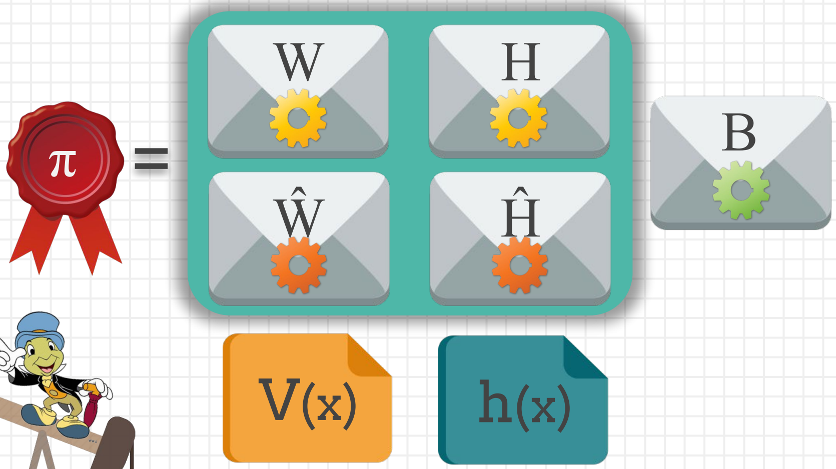
d-PDH



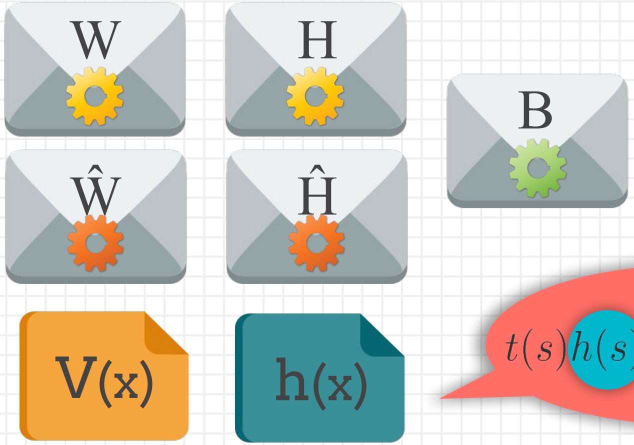
Security Reduction: Cheating Strategy



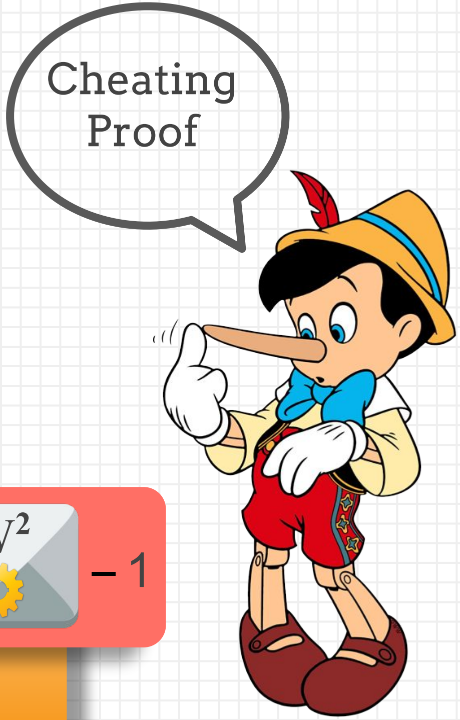
Security Reduction: Cheating Strategy



Polynomial Division does not Hold

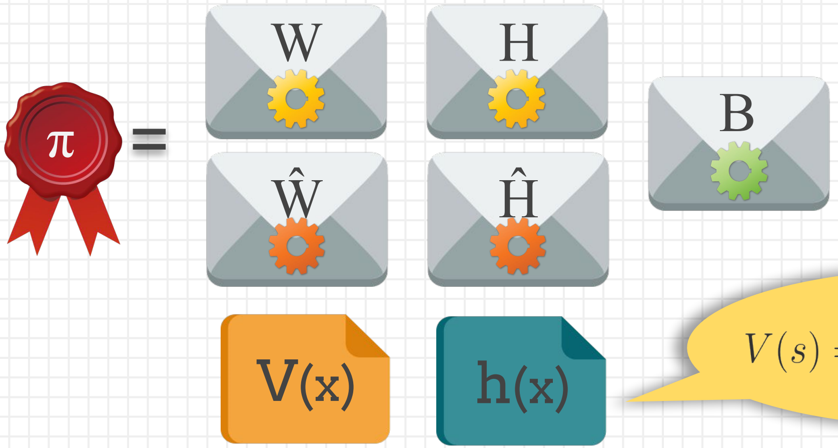

=


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

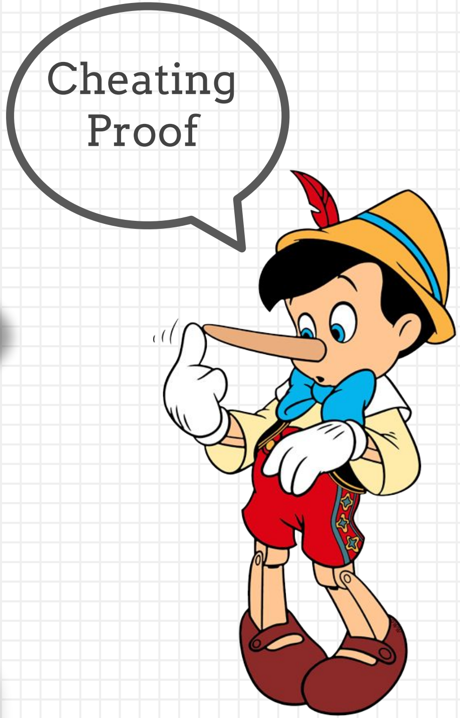


- $t(x)h(x) \neq V^2(x) - 1$, but

Not in the Proper Span

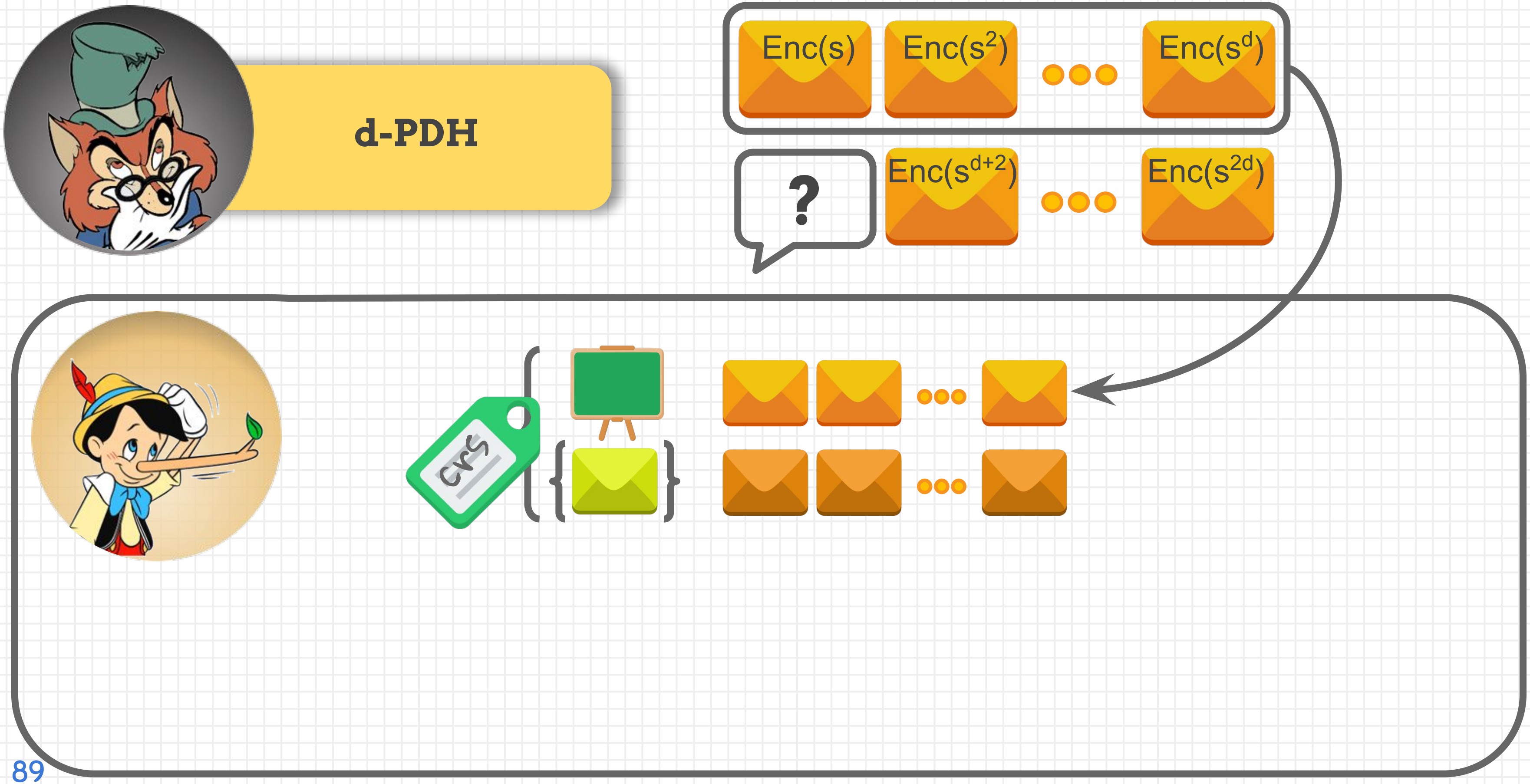


$$V(s) = v_0(s) + \sum_{i=1}^{m} a_i v_i(s)$$

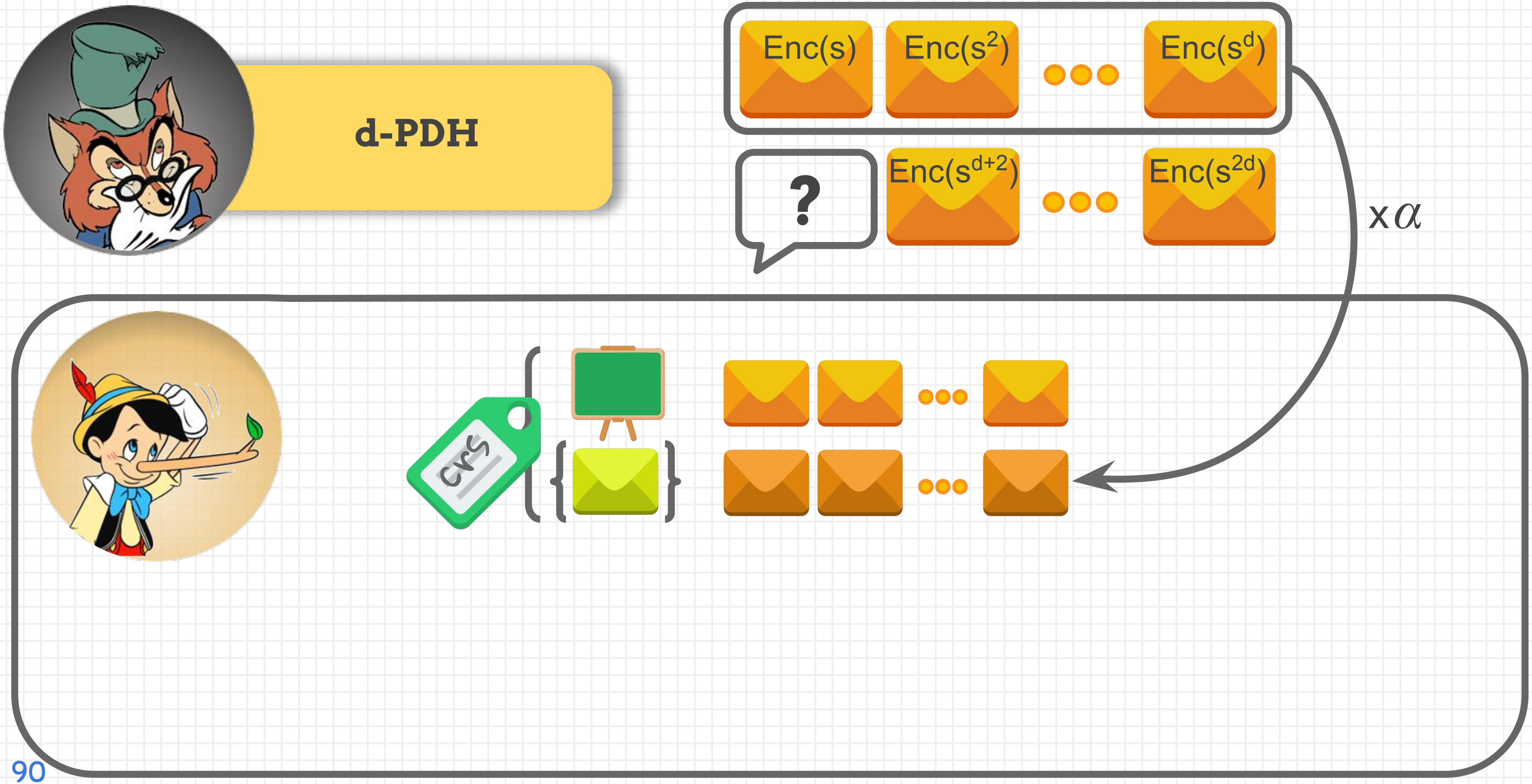


- t(x)h(x)$\neq V^2(x) - 1$, but $\mathsf{Enc}(t(s))H = W^2 - 1$
- V(x) $\notin \mathsf{Span}(v_1, \ldots v_m)$, but

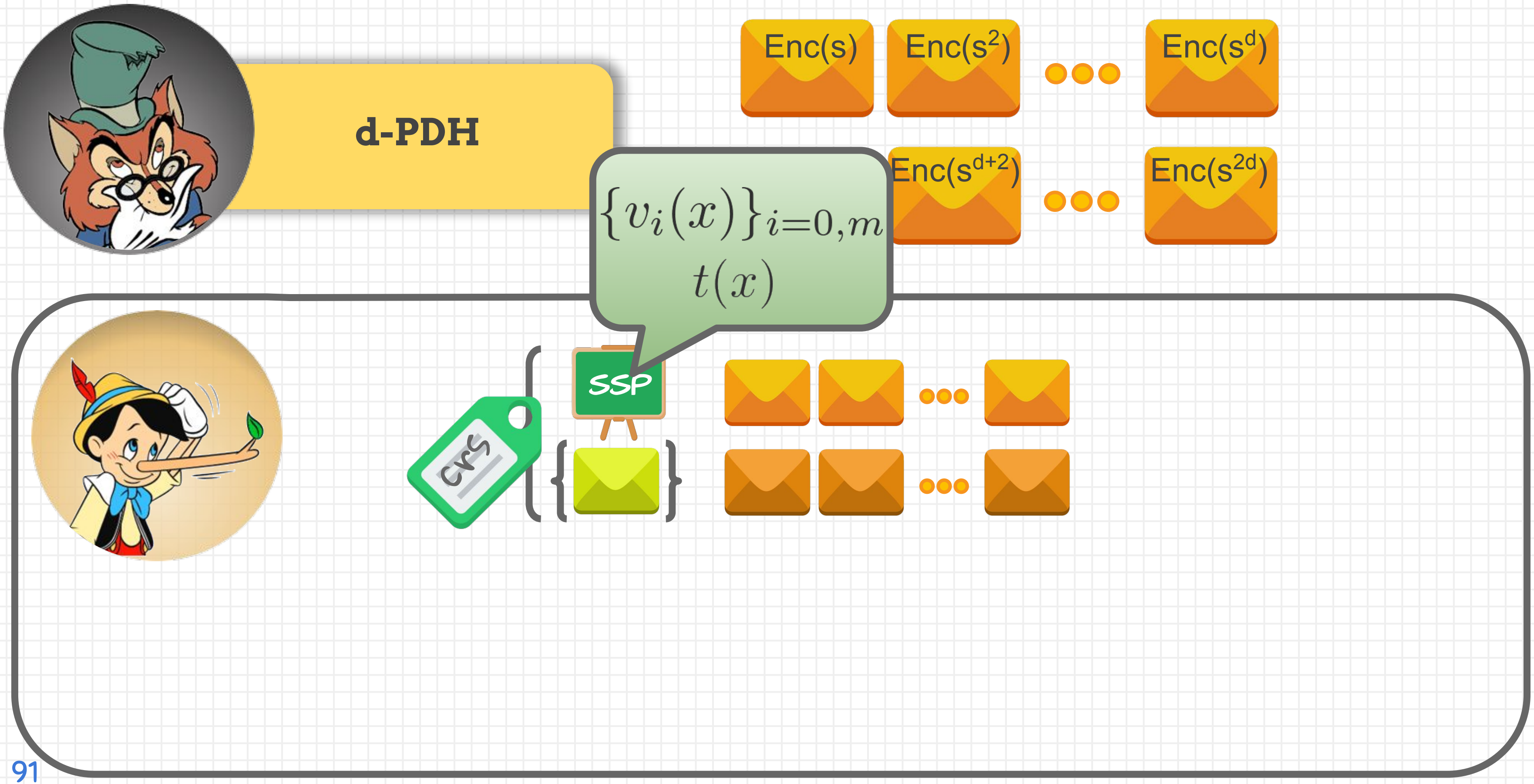
Reduction to d-PDH



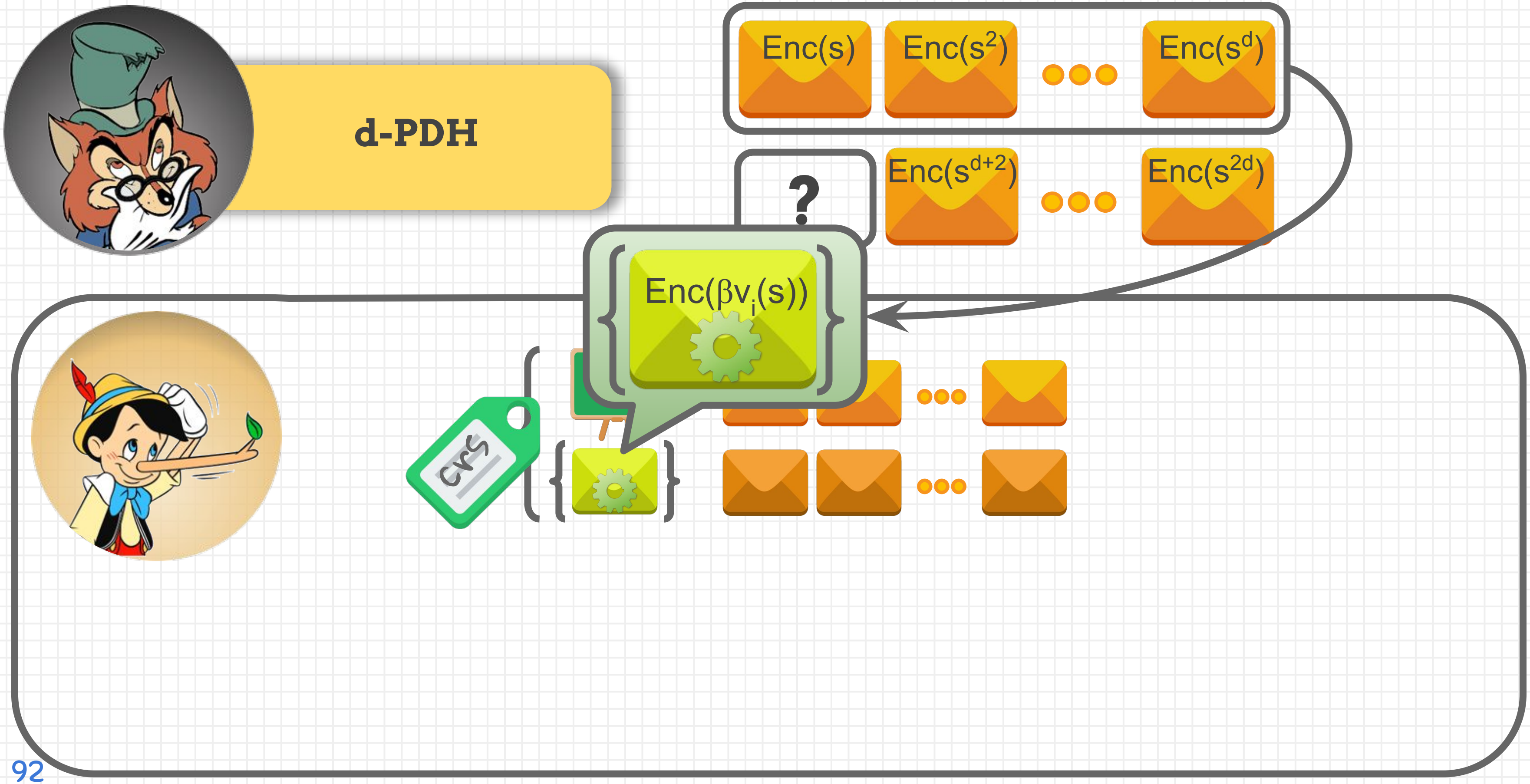
Reduction to d-PDH



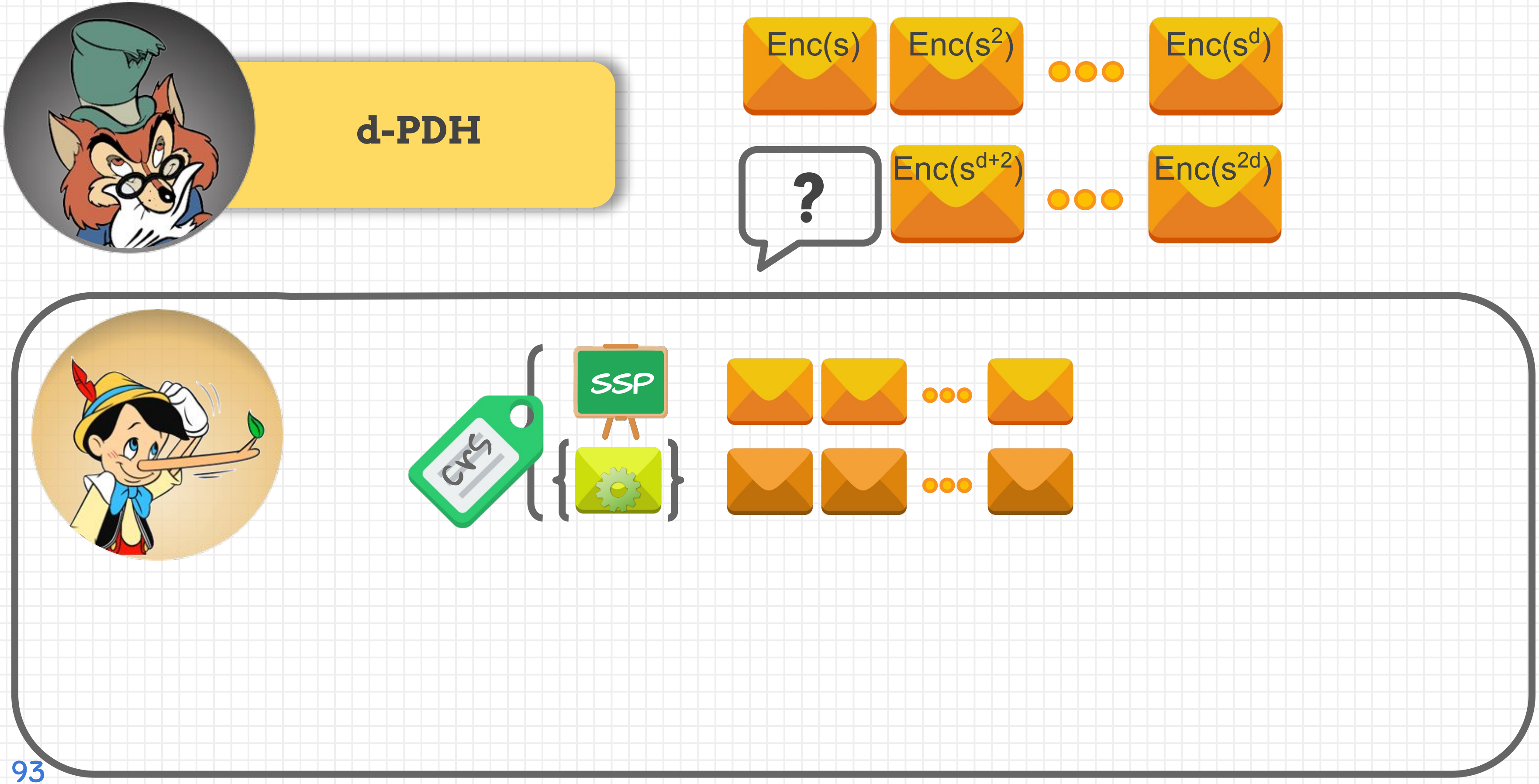
Reduction to d-PDH



Reduction to d-PDH

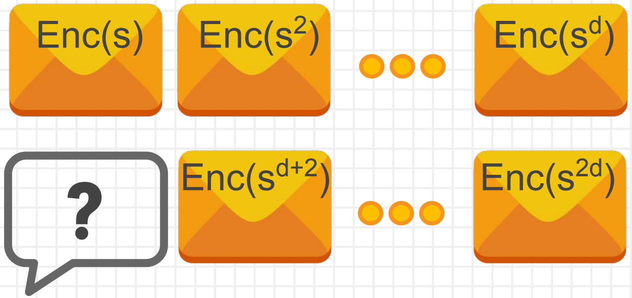


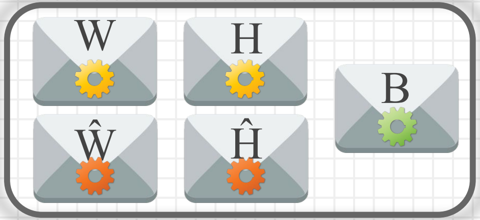
Reduction to d-PDH



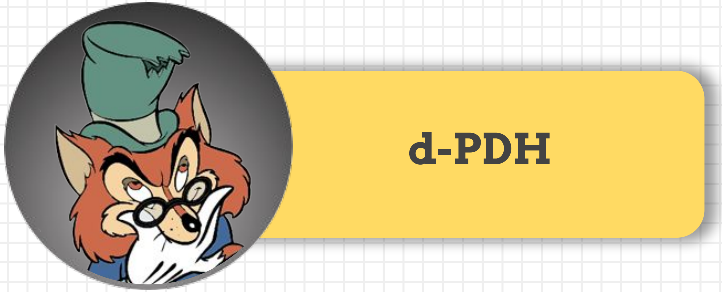
Reduction to d-PDH

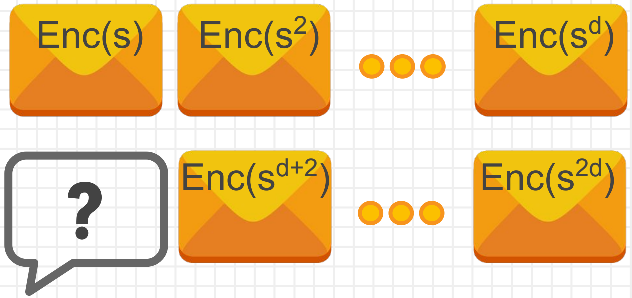
d-PDH

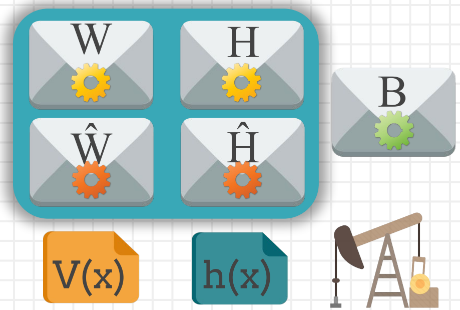




Reduction to d-PDH







Reduction to d-PDH

d-PDH

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

? Enc(s^{d+2}) ... Enc(s^{2d})

W H B

Ŵ Ĥ

V(x)

h(x)

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

Reduction to d-PDH

d-PDH

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

? Enc(s^{d+2}) ... Enc(s^{2d})

W H B

Ŵ Ĥ

V(x)

h(x)

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

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

Reduction to d-PDH

d-PDH

Enc(s^{d+1})

Enc(s) Enc(s^2) ... Enc(s^d) Enc(s^{d+2}) ... Enc(s^{2d})

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

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

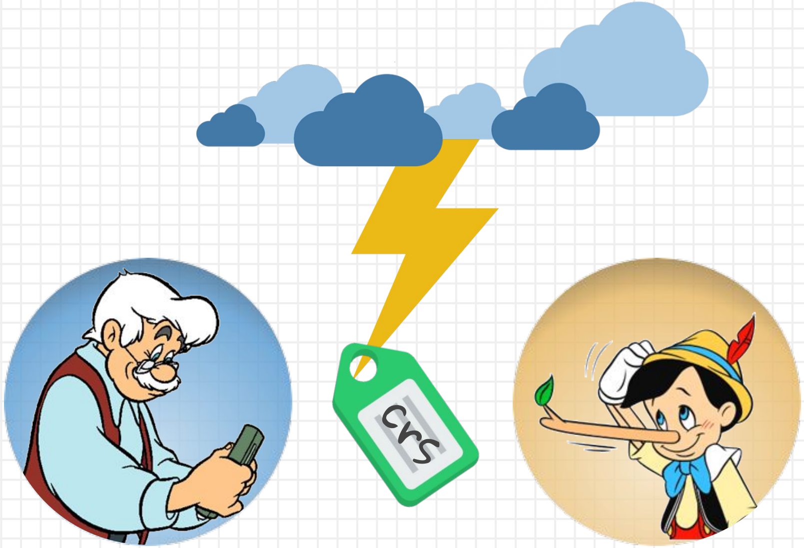
Conclusions

SNARK
Background

Framework
for SNARKs

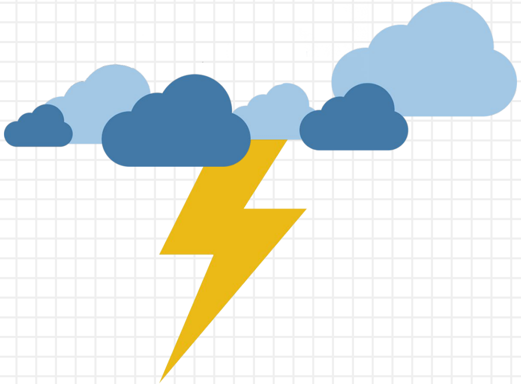
Construction
Security

The
END



Pre-Processing: Trusted Setup





Pre-Processing:
(**crs**: common reference string)

- ✘ Secret coins
- ✘ Expensive
- ✘ Subversion

Subversion-Resistant
Protocols

- ✘ Updatable **crs**
- ✘ Verifiable **crs**

THANK YOU

www.di.ens.fr/~nitulesc