Rinocchio:
SNARKs for Ring Arithmetic

joint work with **Chaya Ganesh**,
Eduardo Soria-Vazquez

Anca Nitulescu

Protocol Labs

Outline

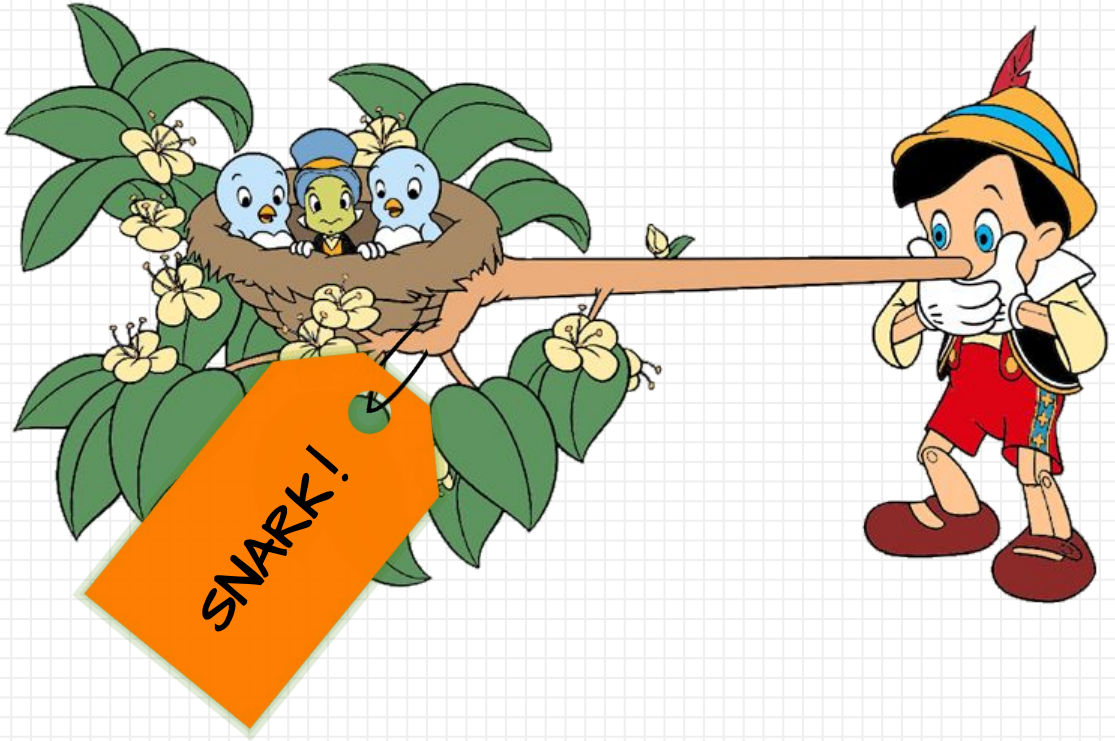
SNARKs
Background

Framework
for Rinocchio

Challenges
New tools

Conclusion

Succinct **N**on-interactive **AR**gument of **K**nowledge



SNARK: Definitions and Properties

SNARK
Background

Framework
for Rinocchio

Challenges
New tools

Conclusion

SNARKs: Proof system

Verifier

$y = f(x)$

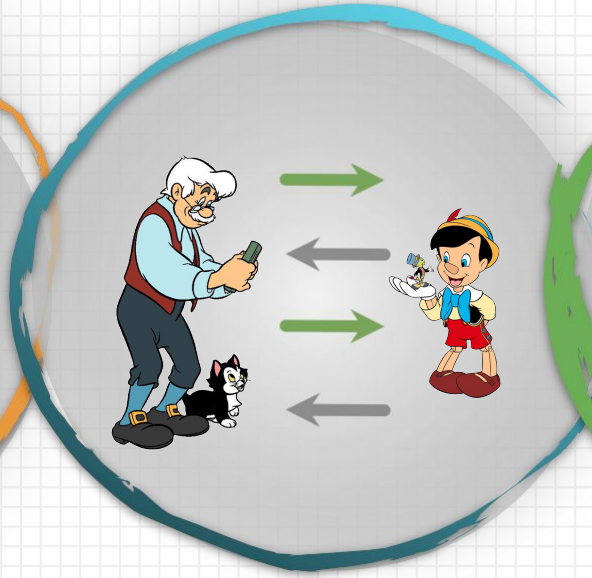
π

Prover

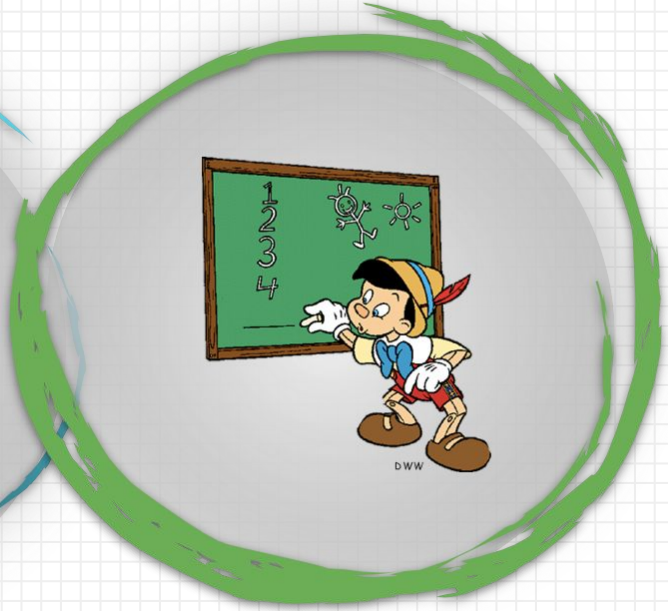
Properties

Succinct Proofs

Non-Interactive

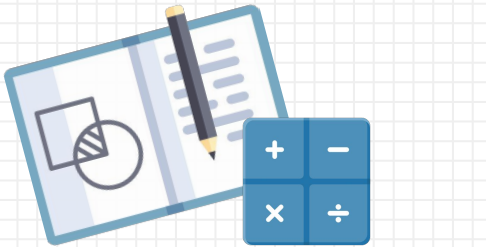


Knowledge Soundness



Usecase: Outsourcing Computation

Verifier



Prover

Prover claims a statement

Verifier

Claim

$y = f(x)$

Prover

Verifier is able to check

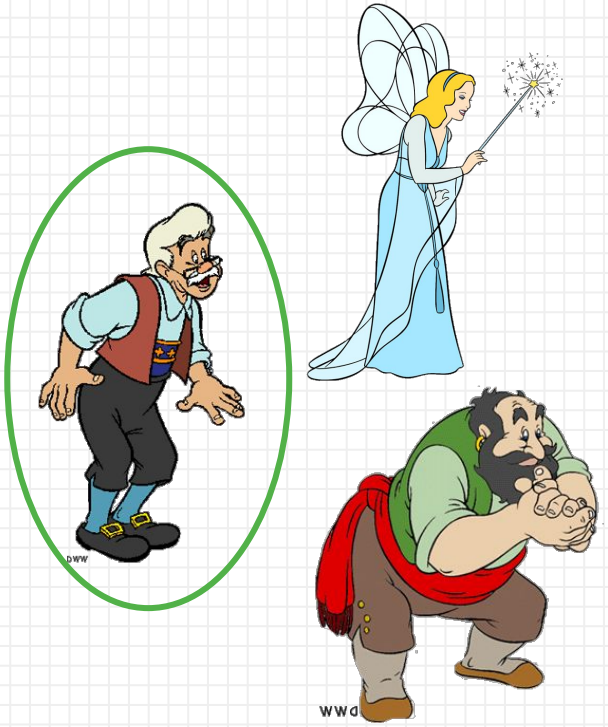
Verifier

Verify

$y' \neq f(x)$

Corrupted Prover

Publicly vs. Designated-Verifier



Verifiers

Claim

Proof π

Prover

SNARKs

PKE

ECRH

GGM

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

[GGPR13] QSP and succinct NIZKs without PCPs

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

[PHGR13] Pinocchio: Nearly practical verifiable computation

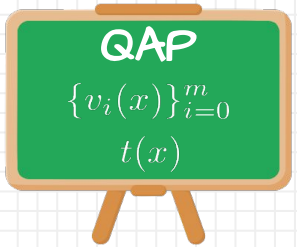
[BCI+13] SNARGs via linear interactive proofs

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

[Groth16] On the Size of Pairing-based Non-interactive Arguments

J. Groth

SNARK: Methodology



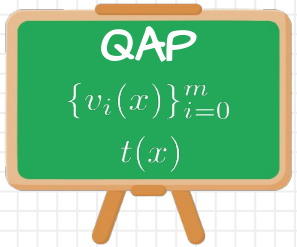
Target Statement
R(y,w)=1

Computational Model
(Representation)

Encodings Secure
under
Knowledge Assumptions

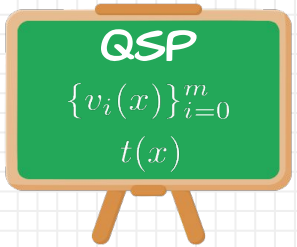
Arithmetic Circuit SAT	**QAP / SAP** over **Field = Z_p**	**PKE** Power Knowledge of Exponent **GGM** Generic Group Model

SNARK: Methodology



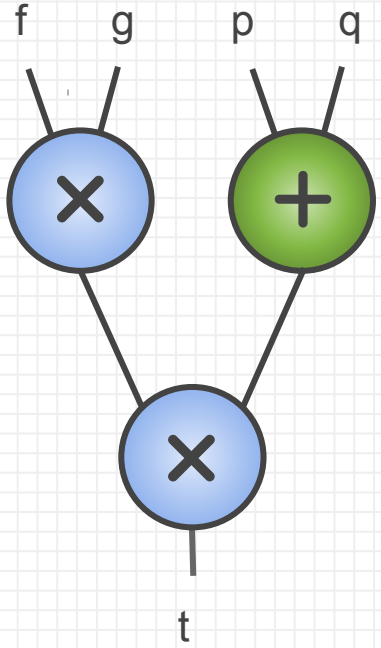
Target Statement R(y,w)=1	Computational Model (Representation)	Encodings Secure under Knowledge Assumptions
Arithmetic Circuit SAT	QAP / SAP over Field = Z_p	PKE Power Knowledge of Exponent GGM Generic Group Model
Boolean Circuit SAT		

SNARK: Methodology



Target Statement R(y,w)=1	Computational Model (Representation)	Encodings Secure under Knowledge Assumptions
Arithmetic Circuit SAT	QAP / SAP over Field = Z_p	**PKE** Power Knowledge of Exponent **GGM** Generic Group Model
Boolean Circuit SAT	QSP / SSP over Field = Z_p	**PKE:** Power Knowledge of Exponent

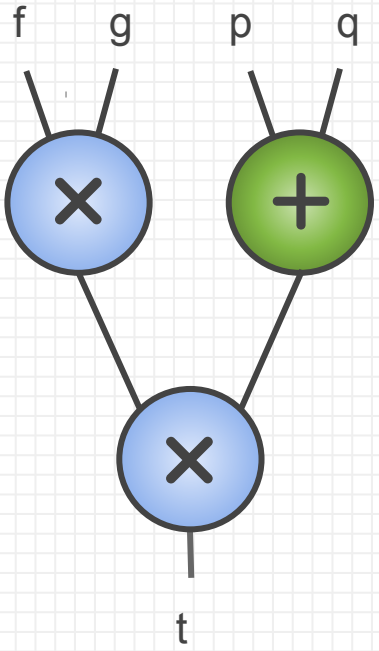
Circuit over Field

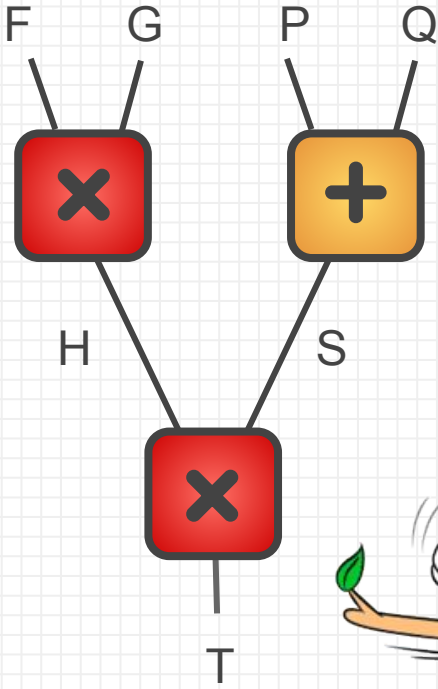


$\mathbb{Z}_p$

$p = 254\text{-bit prime}$

Circuit over Field vs Circuit over Ring





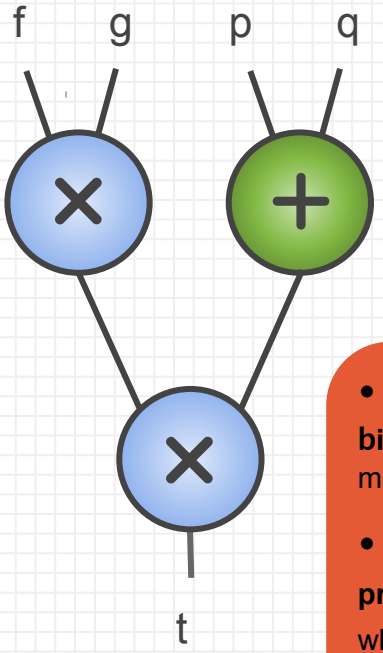
$\mathbb{Z}_p$

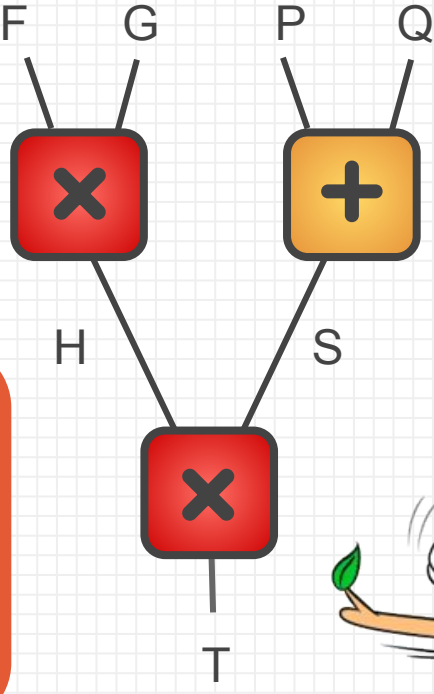
$p = 254$-bit prime

$\mathbb{Z}_{2^k}$

RSA

Circuit over Field vs Circuit over Ring





$\mathbb{Z}_p$

$\mathbb{Z}_{2^k}$

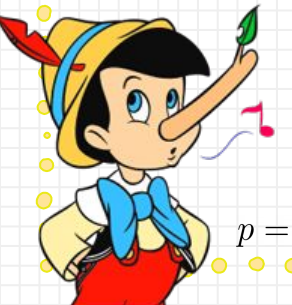
RSA

- modular reduction x mod 2^k

bit decomposition cost = m+1 $\otimes$ where $m = \lceil \log(x_{max}) \rceil$

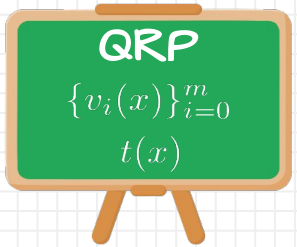
- RSA modulus are larger than p

product of ring elements = $O(m^{1.58})$ $\otimes$ where ring element A = array of m words of 254 bits each



$p = 254\text{-bit prime}$

SNARK: Methodology



Target Statement R(y,w)=1	Computational Model (Representation)	Encodings Secure under Knowledge Assumptions
Arithmetic Circuit SAT	QAP / SAP over Field = Z_p	**PKE** Power Knowledge of Exponent **GGM** Generic Group Model
Boolean Circuit SAT	QSP / SSP over Field = Z_p	**PKE:** Power Knowledge of Exponent
General Circuits over Rings	QRP over Ring = R	**Augmented PKE:** Power Knowledge of Encoding

Contribution for short

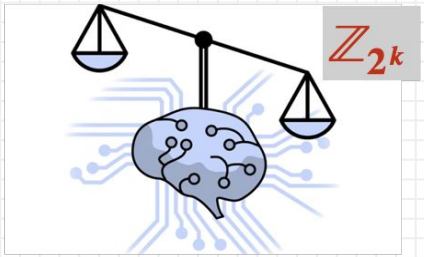
[PHGR13]
Pinocchio: Nearly practical verifiable computation

[GNS21]
Rinocchio

More SNARKs applications

Outsourcing computation
(on encrypted data)

Anonymous Credentials



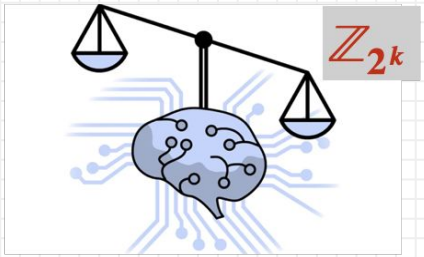
Fairness in
Machine Learning

More SNARKs applications

Rinocchio

Outsourcing computation
(on encrypted data)

Anonymous Credentials



Fairness in
Machine Learning

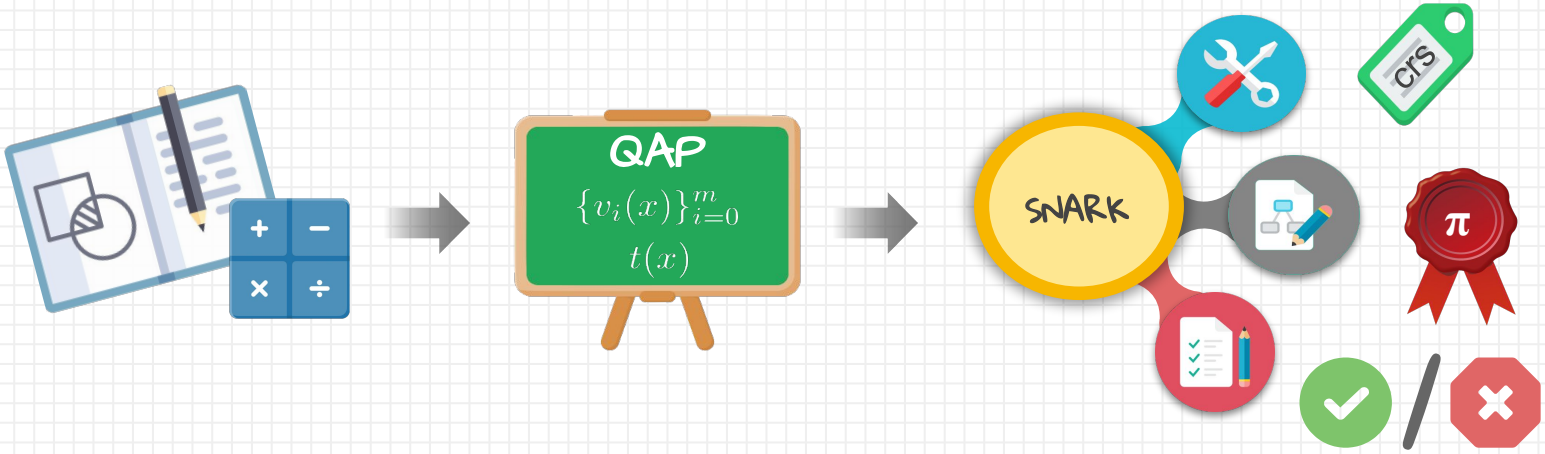
Key Steps to Build SNARKs

SNARK
Background

Framework
for Rinocchio

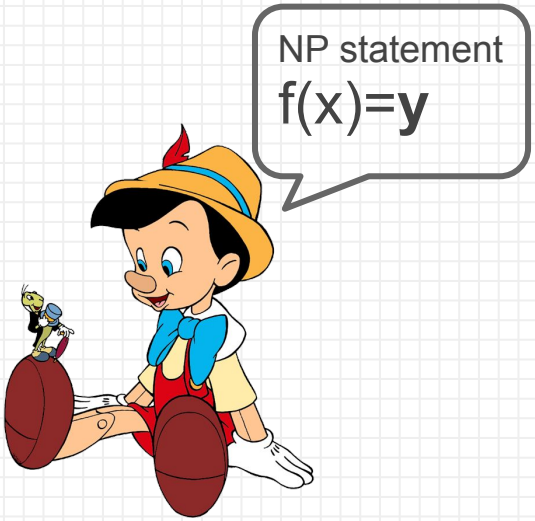
Challenges
New tools

Conclusion



Proving NP statements

Verifier

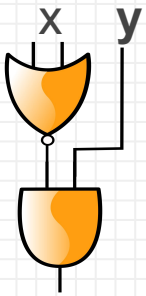


Prover

Prover solves equivalent problem instead

NP statement
f(x)=**y**

compilation



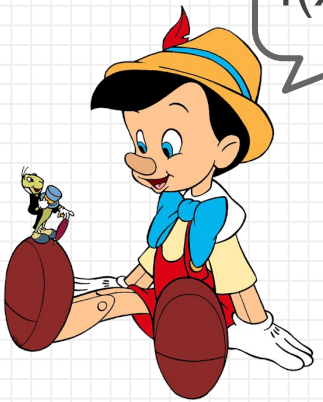
Circuit SAT solution

Polynomial problem

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

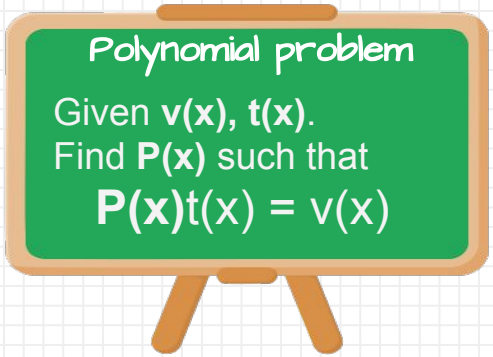
Verifier

NP statement
f(x)=**y**

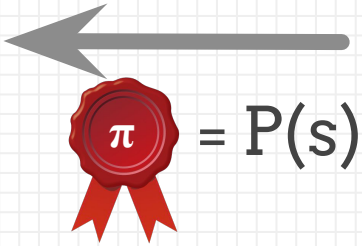


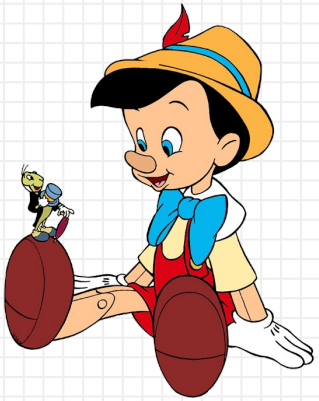
Prover

Evaluate solution at point s



Verifier

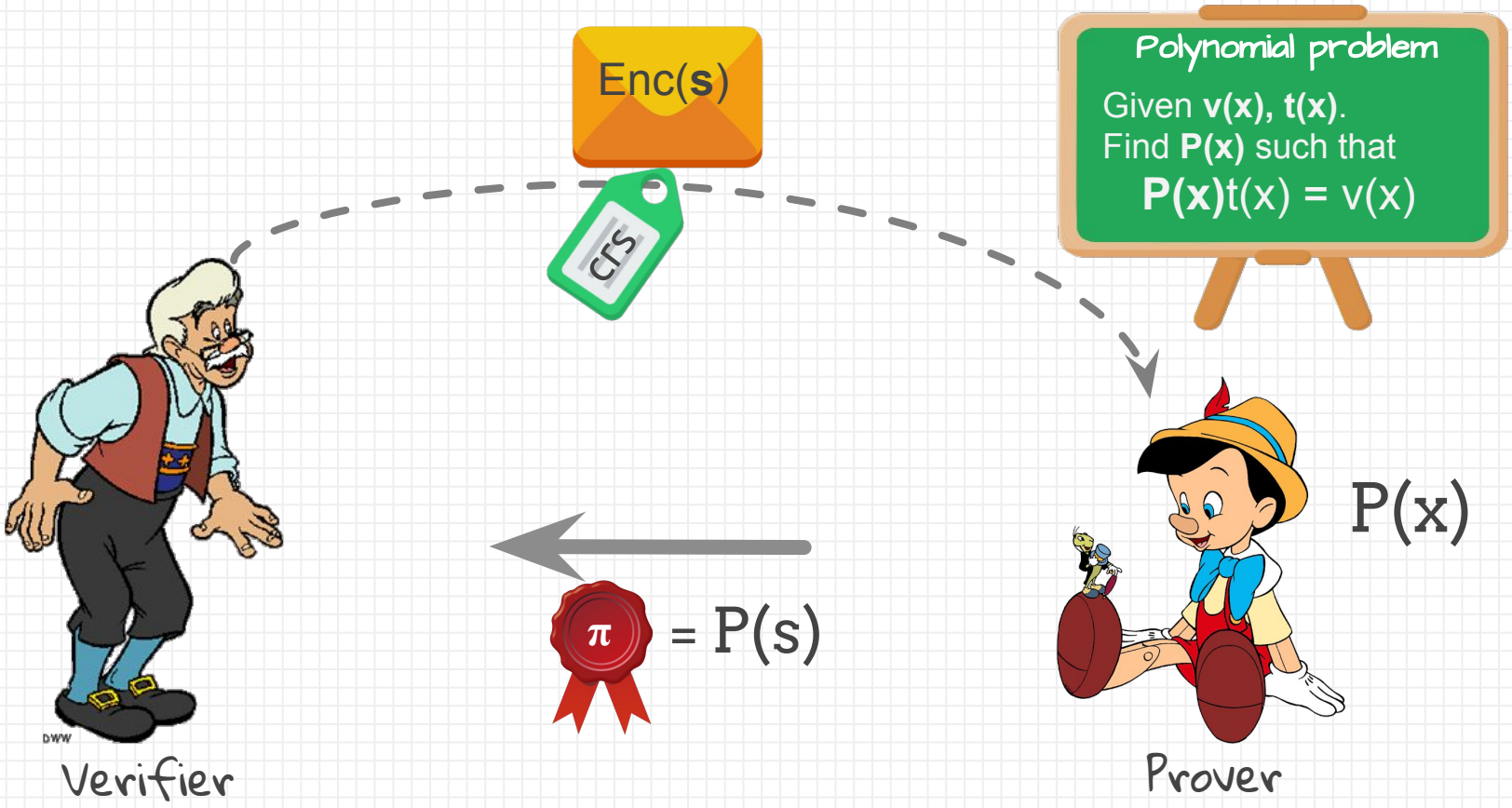




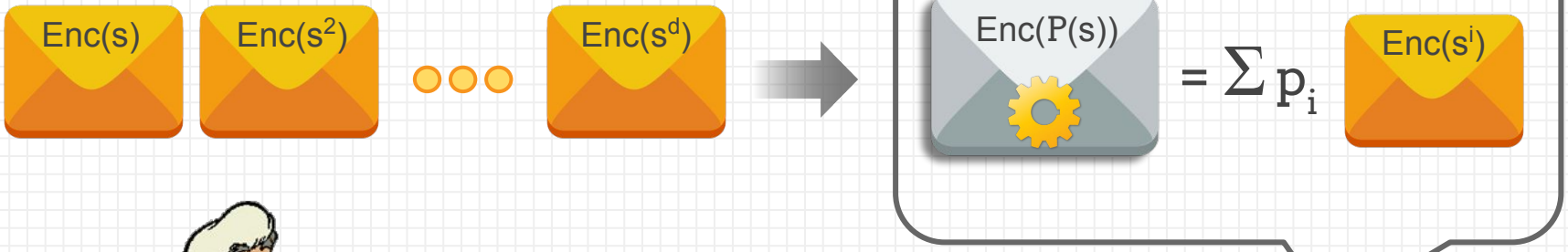
P(x)

Prover

Evaluate solution at point s



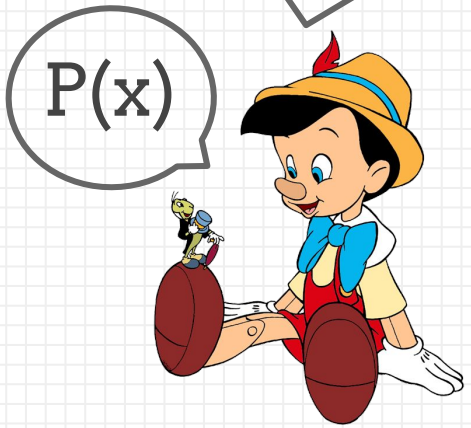
General SNARK framework



Verifier

Encoding:

- ***linearly*** homomorphic

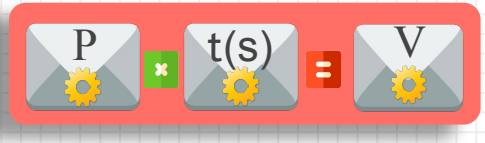


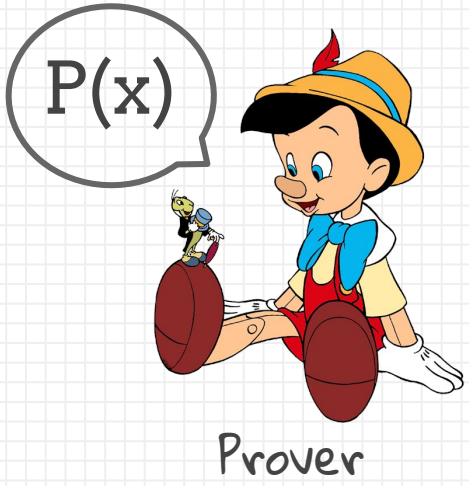
Prover

General SNARK framework

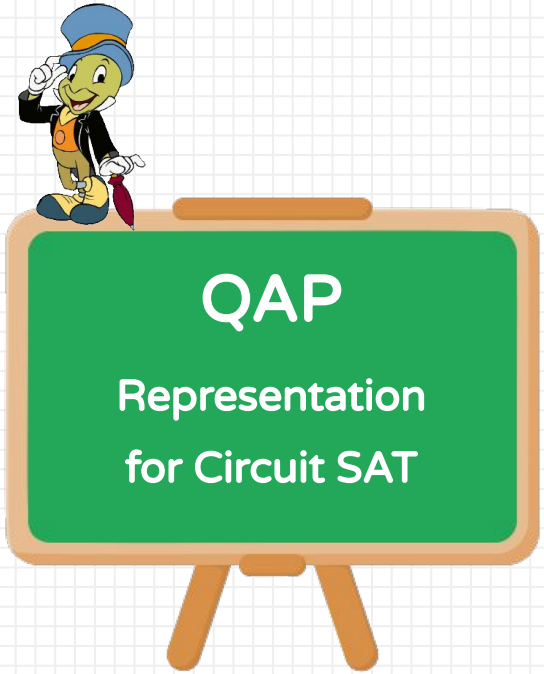
Encoding:

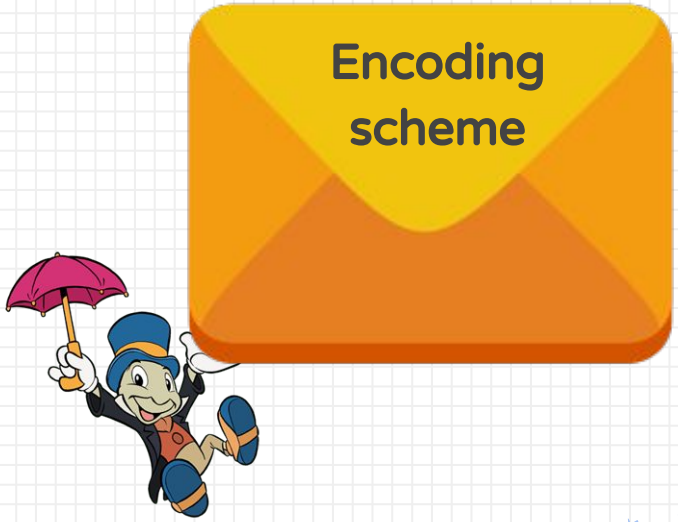
- ***linearly*** homomorphic
- quadratic root detection
- image verification



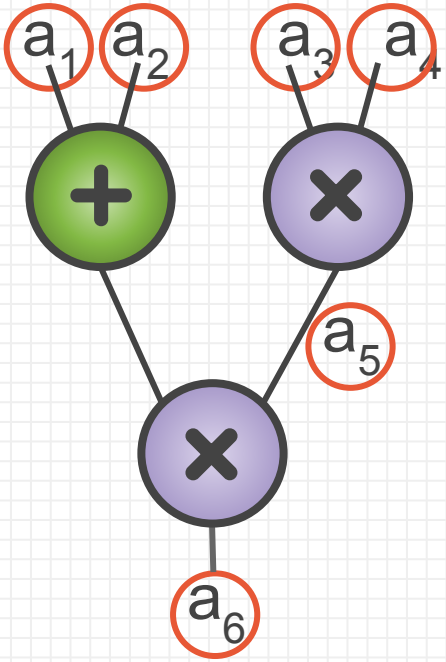


Main Ingredients for SNARKs





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.

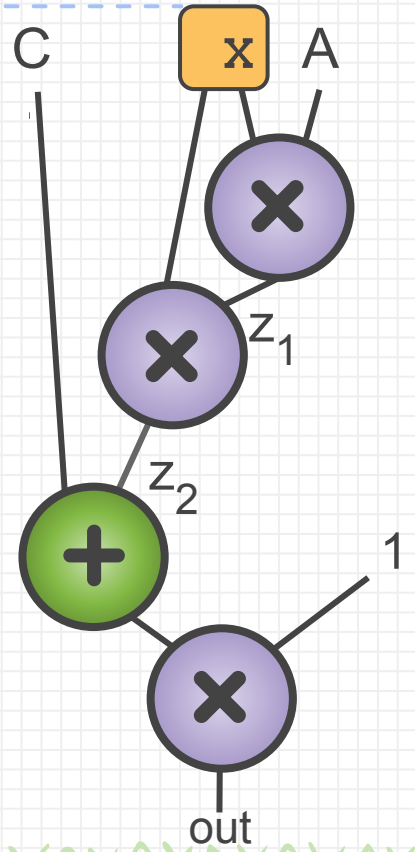
$$V(x) = \sum_{i=0}^{m} a_i v_i(x) \quad ...$$

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

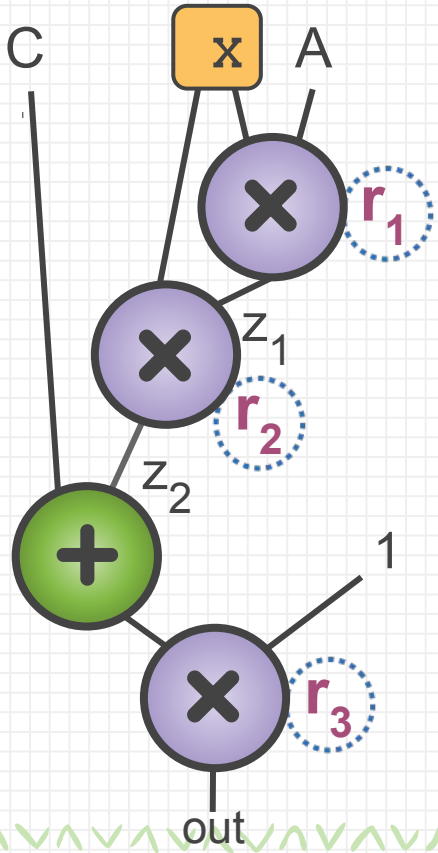
Example: Solution for equation $Ax^2 + C = 0$

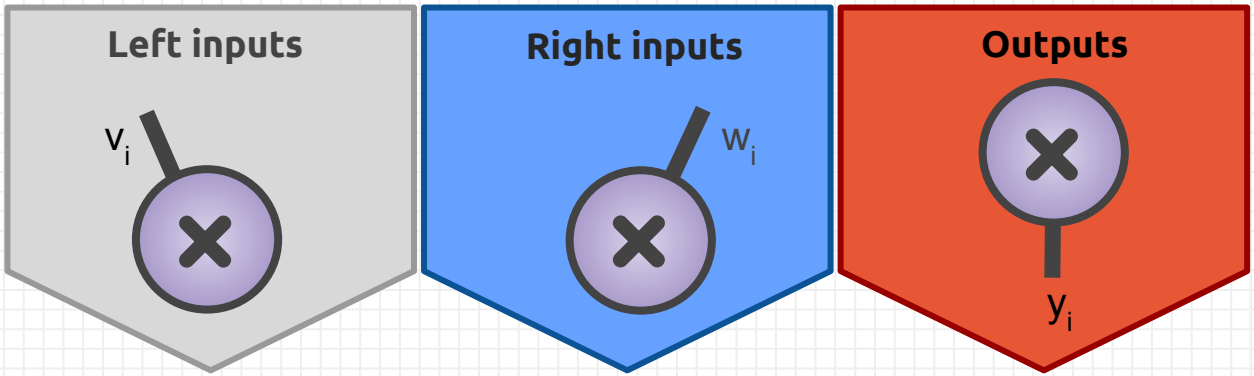
System:

$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$



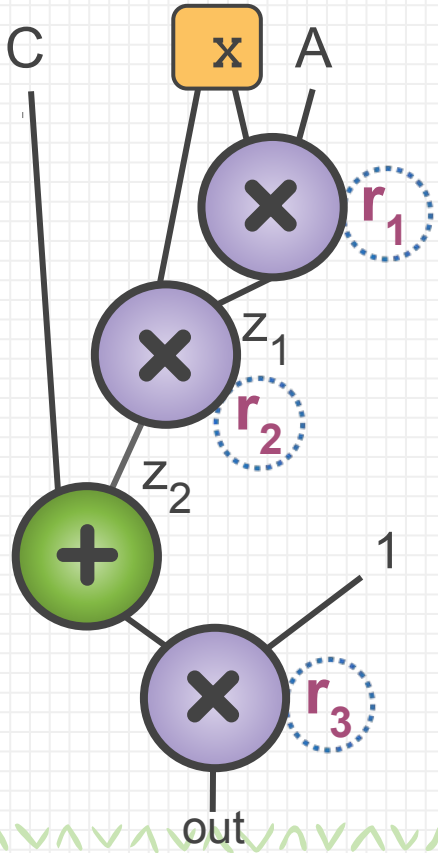
Indexes for wires: A=(0), C=(1), x=(2), z_1=(3), z_2=(4), out = (5)

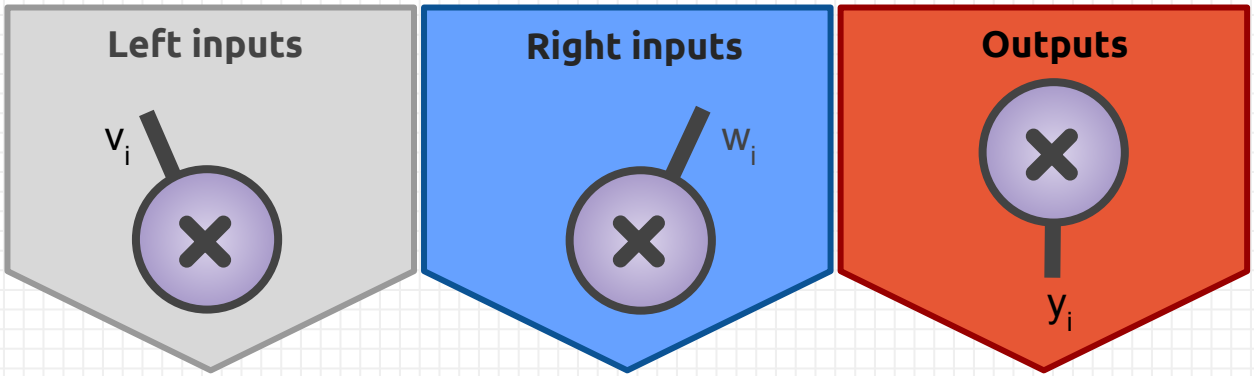




$v_i(r_1) = 1$ for i=2	$w_i(r_1) = 1$ for i=2	$y_i(r_1) = 1$ for i=3,
$v_i(r_2) = 1$ for i=2	$w_i(r_2) = 1$ for i=3	$y_i(r_2) = 1$ for i=4
$v_i(r_3) = 1$ for i=1,4	$w_i(r_3) = 0$ for all i	$w_i(r_3) = 1$ for i=5
$v_i(r_j) = 0$ for the rest	$w_i(r_j) = 0$ for the rest	$y_i(r_j) = 0$ for the rest

Indexes for wires: A=(0), C=(1), x=(2), z_1=(3), z_2=(4), out = (5)

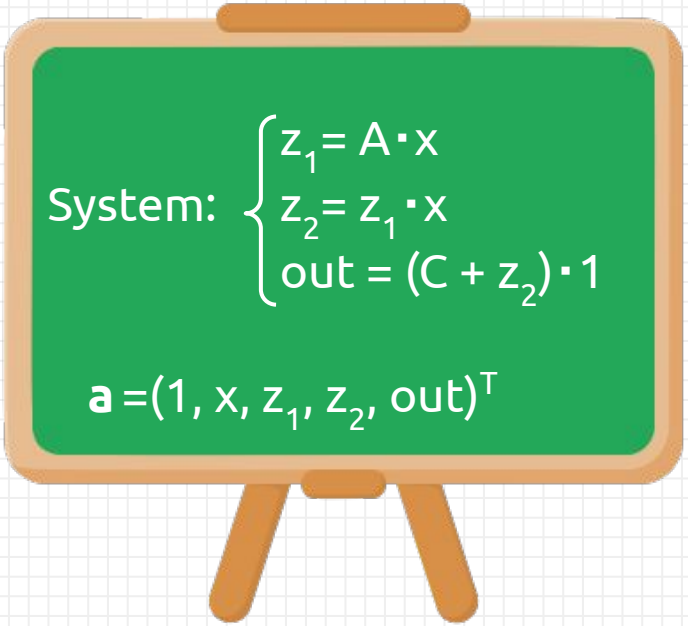




$v_i(r_1) = 1$ for i=2	$w_i(r_1) = 1$ for i=2	$y_i(r_1) = 1$ for i=3,
$v_i(r_2) = 1$ for i=2	$w_i(r_2) = 1$ for i=3	$y_i(r_2) = 1$ for i=4
$v_i(r_3) = 1$ for i=1,4	$w_i(r_3) = 0$ for all i	$w_i(r_3) = 1$ for i=5

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

R1CS for vector $\mathbf{a}=(1, A, C, x, z_1, z_2, out)$



$\mathbf{a}=(1, x, z_1, z_2, out)^T$

$(A, 0, 0, 0, 0) \cdot \mathbf{a} \circ (0, 1, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 1, 0, 0) \cdot \mathbf{a}$

$(0, 0, 1, 0, 0) \cdot \mathbf{a} \circ (0, 1, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 0, 1, 0) \cdot \mathbf{a}$

$(C, 0, 0, 1, 0) \cdot \mathbf{a} \circ (1, 0, 0, 0, 0) \cdot \mathbf{a} = (0, 0, 0, 0, 1) \cdot \mathbf{a}$

$$\left[V \; a \right] \circ \left[W \; a \right] = Y \; a$$

R1CS for vector **a** =(1,A, C, x, z_1, z_2, out)

System:
$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

a =(1, x, z_1, z_2, out)^T

$$v_i(r_j) = V_{ji} \qquad \forall \{r_j\} \in \mathbb{F}^d$$

$$\left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) = \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

R1CS for vector **a** =(1,A, C, x, z_1, z_2, out)

System:
$$\begin{cases} z_1 = A \cdot x \\ z_2 = z_1 \cdot x \\ out = (C + z_2) \cdot 1 \end{cases}$$

a $=(1, x, z_1, z_2, out)^T$

$$v_i(r_j) = V_{ji} \qquad \forall \{r_j\} \in \mathbb{F}^d$$

$$\left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) = \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

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

Schwartz-Zippel Lemma over Fields

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

Lemma: Let $f \in \mathbb{F}[X]$ be a **non-zero** poly.

$$\Pr_{s \leftarrow \mathbb{F}}[f(s) = 0] \leq \frac{\deg(f)}{|\mathbb{F}|}$$

Encodings over Fields

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 vs General Encoding

DLog Group $\mathbb{G}$

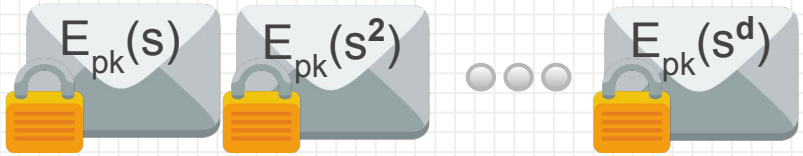
$$\langle g \rangle = \mathbb{G}, \quad Enc(s) = g^s$$

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

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

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

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



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

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

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**

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

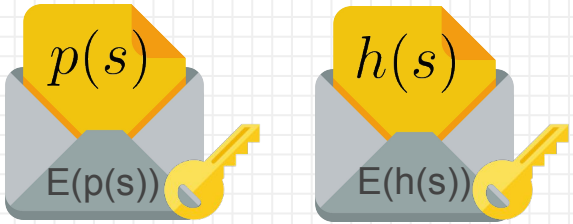
$$e(g^{t(s)}, g^{h(s)}) \stackrel{?}{=} e(g^{p(s)}, g)$$

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

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

Quadratic root detection needs **sk**

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



Assumptions on Discrete Log Encoding for Fields

d-PKE

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

g^{s} g^{s^2} ••• g^{s^d}

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

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

d-PDH

g^{s} g^{s^2} ••• g^{s^d}

? $g^{s^{d+2}}$ ••• $g^{s^{2d}}$

Technical Details

SNARK
Background

Framework
for Rinocchio

Challenges
New tools

Conclusion

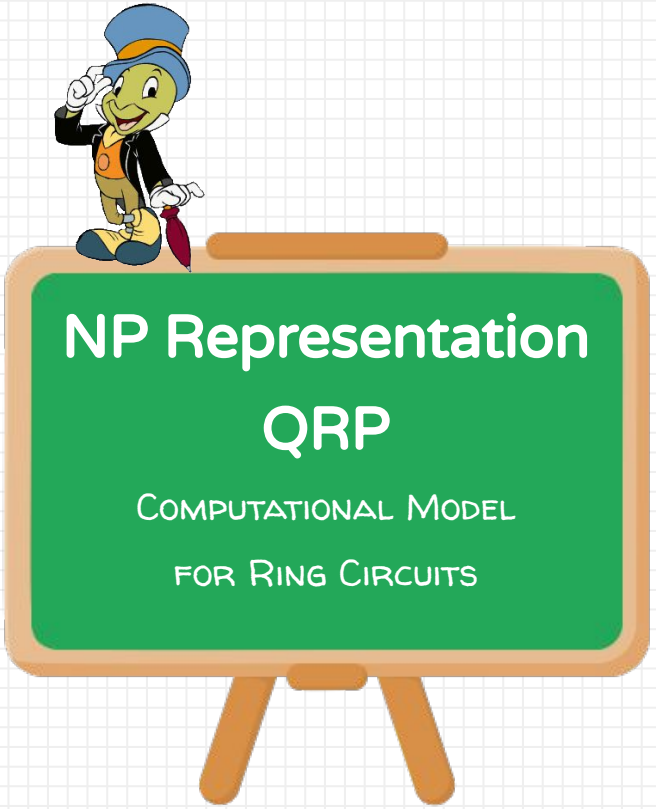
SNARKs for Ring Arithmetics

Generic framework for (zk)-SNARK over Ring

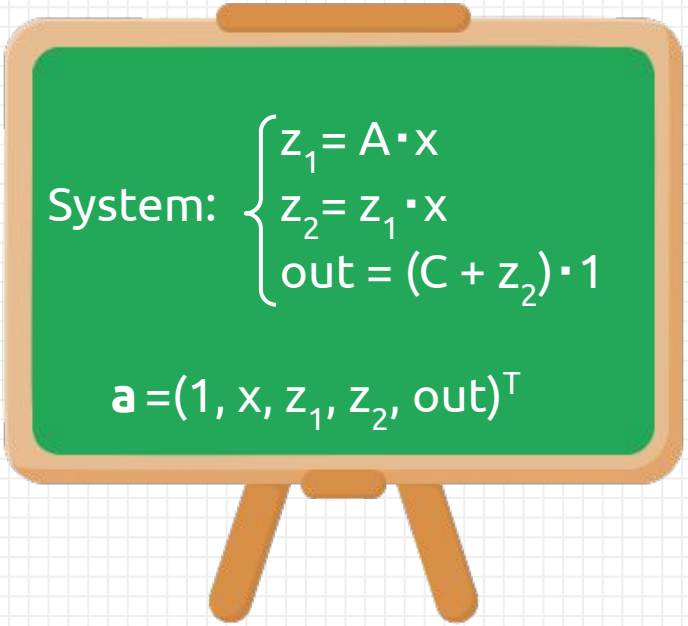
- ✘ Circuit-SAT for arithmetic circuits over commutative rings:
  Quadratic Ring Programs (QRP)
- ✘ Suitable secure encodings over Rings

Properties

- ✘ Fits FHE schemes arithmetics
- ✘ Designated Verifier only


NP Representation
QRP
Computational Model
for Ring Circuits

Proving a Solution for Equation $Ax^2 + C = 0$



$$\left[\begin{matrix} V \\ d \times m \end{matrix}\right] \mathbf{a} \circ \left[\begin{matrix} W \\ d \times m \end{matrix}\right] \mathbf{a} = \left[\begin{matrix} Y \\ d \times m \end{matrix}\right] \mathbf{a}$$

$$v_i(r_j) = V_{ji} \qquad \forall \{r_j\} \in \mathbb{F}^d$$

$$\left(\sum_{i=0}^{m} a_i v_i(x)\right)\left(\sum_{i=0}^{m} a_i w_i(x)\right) = \left(\sum_{i=0}^{m} a_i y_i(x)\right)$$

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

Polynomial Equation with Coefficients in a Ring

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

Necessary property over Rings for **Ideals** $I_j = (x - r_j)$

Isomorphism for **QRP** soundness ⇔ **Ideals** I_j are co-prime:

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \ldots \times \frac{R[x]}{I_d} \simeq R \times \ldots \times R$$

$$p(x) \longmapsto \big(p_1(x), \ldots, p_d(x)\big) \longmapsto \big(p(r_1), \ldots, p(r_d)\big)$$

Polynomial Equation with Coefficients in a Ring

Works for $\mathbb{R} = \mathbb{F}$, as then the ideals I_j are co-prime.

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

Necessary property over Rings for **Ideals** $I_j = (x - r_j)$

Isomorphism for **QRP** soundness ⇔ **Ideals** I_j are co-prime:

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \ldots \times \frac{R[x]}{I_d} \simeq R \times \ldots \times R$$

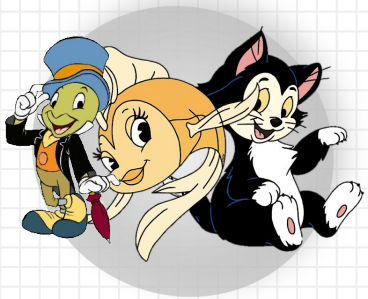
$$p(x) \longmapsto \big(p_1(x), \ldots, p_d(x)\big) \longmapsto \big(p(r_1), \ldots, p(r_d)\big)$$

Exceptional Sets: to the rescue!

Def: Let R be a commutative ring. A set $\mathbf{A} = \{g_1, \ldots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\, i \neq j,\ (g_i - g_j) \in R^*$$

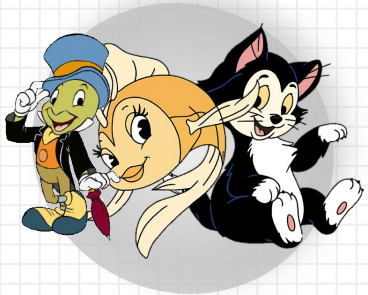
Exceptional sets have **no further** algebraic **structure.**
Not even <u>closure</u>!



Exceptional Sets

Def: Let R be a commutative ring. A set $\mathbf{A} = \{g_1, \ldots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\, i \neq j,\ (g_i - g_j) \in R^*$$



Exceptional sets have **no further** algebraic **structure.** Not even <u>closure</u>!

Given exceptional set $\mathbf{A}$, the **ideals** $I_j = \left(x - g_j\right)$ are **pairwise co-prime** (i.e. $\forall i \neq j,\ \ I_i + I_j = R[X]$).

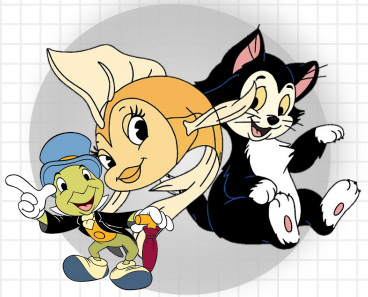
- Proof: $-\left(x - g_i\right) + (x - g_j) = (g_i - g_j) \in R^*$
- Meaning: We can apply CRT in R[X], for big enough $\mathbf{A} \subset R$.

Exceptional Sets

Def: Let R be a commutative ring. A set $\mathbf{A} = \{g_1, \ldots, g_n\} \subset R$ is ***exceptional*** iff:

$$\forall\, i \neq j,\ (g_i - g_j) \in R^*$$

Exceptional sets have **no further** algebraic **structure.** Not even closure!



Given exceptional set $\mathbf{A}$, the **ideals** $I_j = \left(x - g_j\right)$ are **pairwise co-prime** (i.e. $\forall i \neq j,\ \ I_i + I_j = R[X]$).

$$\frac{R[x]}{(t(x))} \simeq \frac{R[x]}{I_1} \times \ldots \times \frac{R[x]}{I_d} \simeq R \times \ldots \times R$$

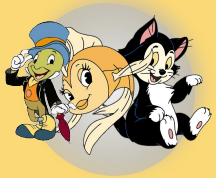
$$\mathrm{p}(x) \longmapsto \left(p_1(x), \ldots, p_d(x)\right) \longmapsto \left(p(g_1), \ldots, p(g_d)\right)$$

Schwartz-Zippel Lemma over Rings

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

Lemma: Let $f \in R[X]$ be a **non-zero** poly.

$$\Pr_{s \leftarrow A}[f(s) = 0] \leq \frac{\deg(f)}{|A|}$$



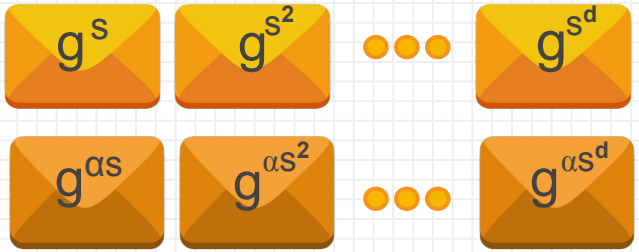
Encodings

Properties

Assumptions

Assumptions on Discrete Log Encoding for Fields

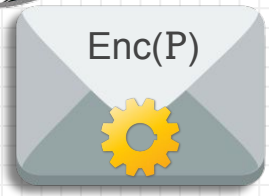
d-PKE

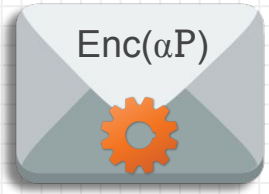


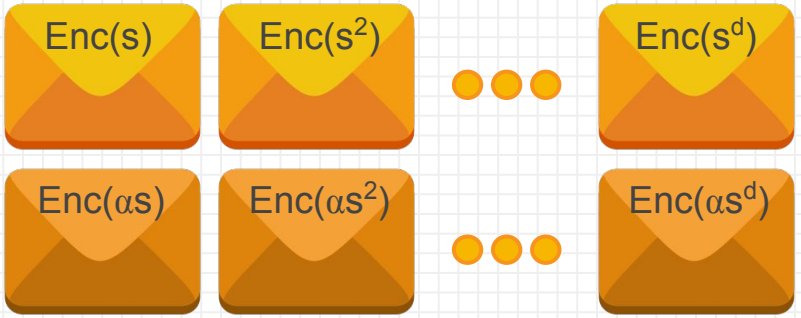
d-PDH

Augmented Power Knowledge of Encoding

d-PKE

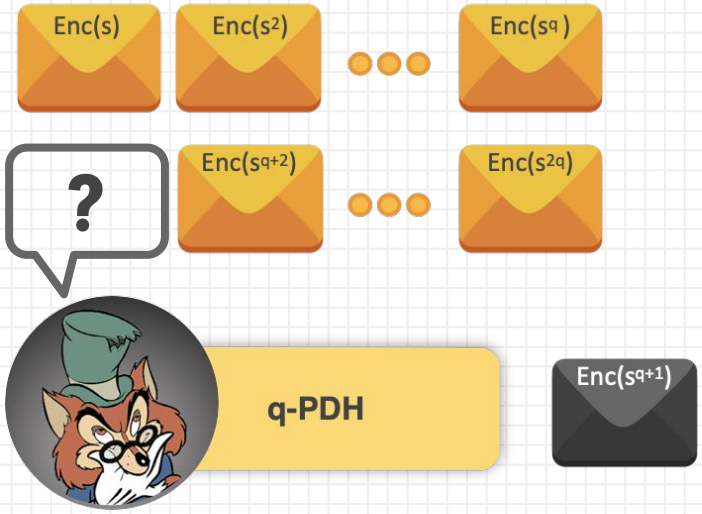




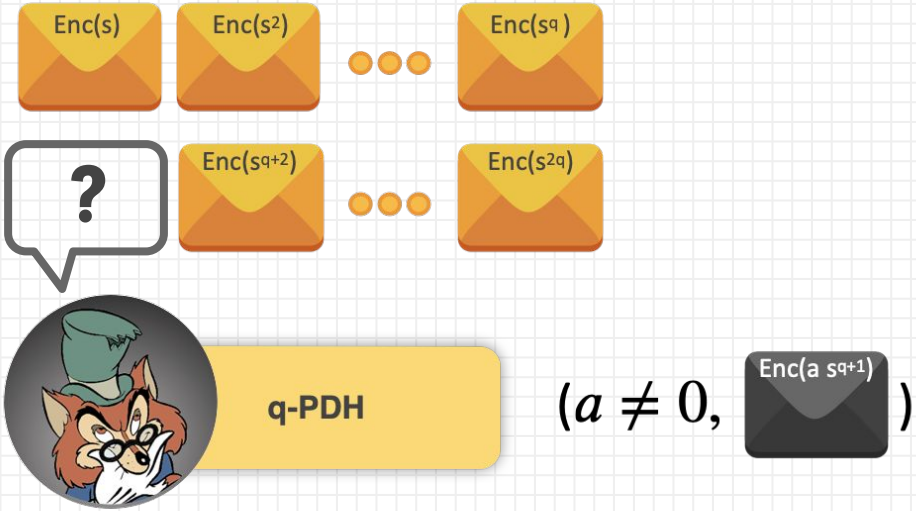


Ring Augmented PDH Assumption

$s \leftarrow \mathbb{F}^*$



$s \leftarrow A^*$, where A is an exceptional set such that $A^* \subset R^*$.

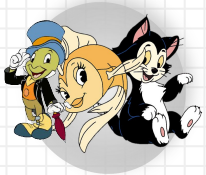


Encoding Instantiation for Ring $\mathbb{Z}_{2^k}$

Problem: There's no exceptional set *A* of size > 2 in $\mathbb{Z}_{2^k}$.

Sol: Move to $GR(2^k, d)$, the degree-*d* Galois extension of $\mathbb{Z}_{2^k}$.

$$a = a_0 + a_1 X + \ldots + a_{d-1} X^{d-1}; \quad a_i \in \mathbb{Z}_{2^k}$$



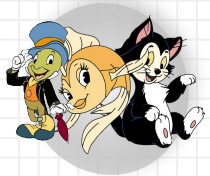
$$A = \{a_0 + a_1 X + \ldots + a_{d-1} X^{d-1} : a_i \in \{0,1\}\} \subset GR(2^k, d)$$

Encoding: *d* copies of a $\mathbb{Z}_{2^k}$– linearly homomorphic scheme, such as [JoyeLibert13].

$$E(a) = (Enc_{pk_1}(a_0), \ldots, Enc_{pk_d}(a_{d-1})); \quad a_i \in \mathbb{Z}_{2^k}$$

Encoding Instantiation for Ring $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$

Rings of the form $\mathcal{R}_q = \mathbb{Z}_q[X]/(h(X))$.



$= \{0,\ldots,p_1 - 1\} \subset \mathcal{R}_q; \quad q = \prod p_i$ s.t. $p_1 < p_2 < \ldots$

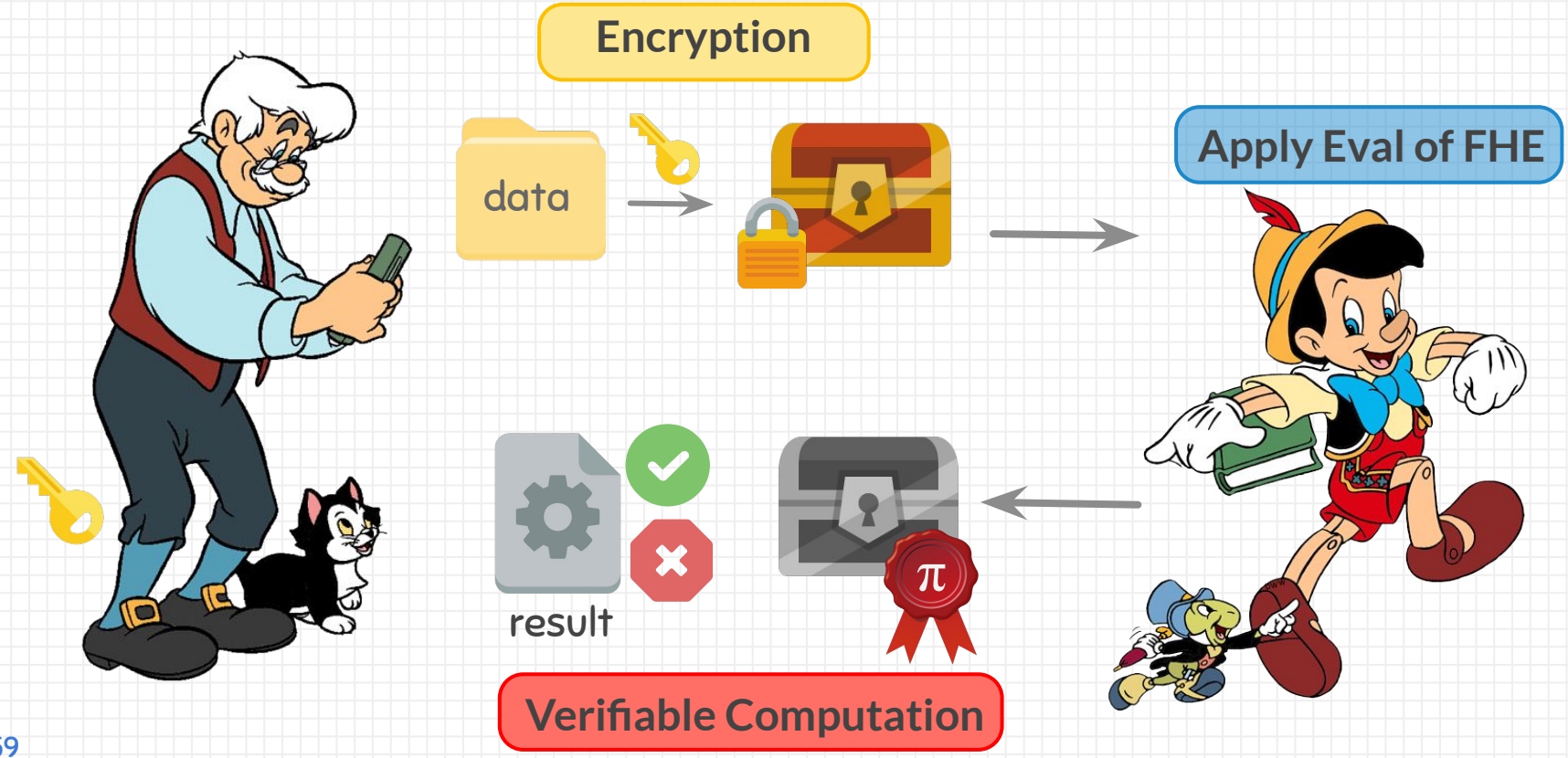
Compared with [FioreNitulescuPointcheval20]:

[FVP20] Boosting Verifiable Computation on Encrypted Data

Dario Fiore, Anca Nitulescu, David Pointcheval

- ✘ Significantly better choices for RLWE parameters
- ✘ FVP20 only supports rings with q prime → very inefficient FHE
- ✘ We enable operations for plaintext packing, modulo switching
- ✘ We are only designated-verifier

Application: Verifiable Computation on Encrypted Data



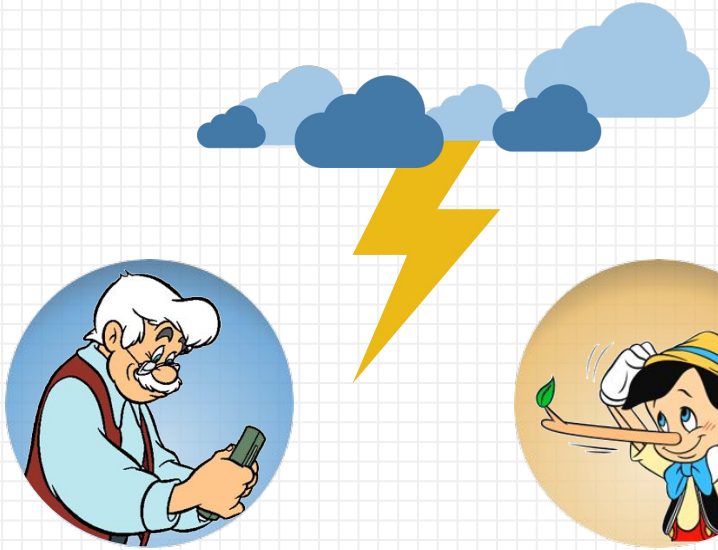
Conclusions

SNARK
Background

Framework
for Rinocchio

Challenges
New tools

Conclusion



Conclusions

Quadratic Programs and SNARKs over fields

- ✘ Lots of implementations, but they fall short in one aspect
- ✘ Emulating ring arithmetic on SNARKs is expensive and unfriendly to applications
- ✘ Today's cost: Compilation to circuits over fields, costly preprocessing

Rinocchio: SNARKs for Ring Arithmetics

- ✘ Circuit-SAT for arithmetic circuits over commutative rings: Quadratic Ring Programs (QRP)
- ✘ Better fits FHE schemes arithmetics $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$ even for q not prime
- ✘ Supports sub-circuits for special operations in FHE: modulo switching
- ✘ Designated Verifier only
- ✘ Can be turned Zero-Knowledge using Context Hiding techniques

Conclusions

Quadratic Programs and SNARKs over fields

- ✘ Lots of implementations, but they fall short in one aspect
- ✘ Emulating ring arithmetic on SNARKs is expensive and unfriendly to applications
- ✘ Today's cost: Compilation to circuits over fields, costly prepocessing

Rinocchio: SNARKs for Ring Arithmetics

- ✘ Circuit-SAT for arithmetic circuits over commutative rings: Quadratic Ring Programs (QRP)
- ✘ Better fits FHE schemes arithmetics $\mathbb{R}_q = \mathbb{Z}_q[x]/R(x)$ even for q not prime
- ✘ Supports sub-circuits for special operations in FHE: modulo switching
- ✘ Designated Verifier only
- ✘ Can be turned Zero-Knowledge using Context Hiding techniques

Open Questions

- ✘ Other Encodings over rings → publicly verifiable
- ✘ More efficient instantiations: Security assumptions over rings: L-O extractable vs PKE

THANK YOU

eprint.iacr.org/2021/322

Credits

Special thanks to all those who made and released these resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ Illustrations by Disneyclips, Iconfinder and Flaticon