SNARKs Tutorial: Introduction & SSP-Based Construction

Anca Nitulescu
"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 non-deterministic 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, CRYPTO’98].
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 Framework
for SNARKs

SNARK Construction
Security

The END

Verifier

Prover
Delegated Computation

Verifier computes $f(x) = y$

Prover

Task
Prover claims a statement

Verifier

Claim

y = f(x)

Prover
Verifier does not trust

$y \neq f(x)$

Verifier

Corrupted Prover

$f(x) = y$
A note on efficient zk-proofs and arguments

Joe Kilian

[Kil92]
Interactive Proof Protocol [Kil92]

Verifier

Prover

Claim

$y = f(x)$

$q/a$
Decentralised Setting: Multiple Verifiers?
Proof Systems: Non-Interactive Arguments

[ Mic00 ]
Computationally sound proofs

Joe Kilian

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

Silvio Micali
Removing Interaction with Random Oracle
Non-Interactive Proof Protocol [Mic00]
Succinct Non-interactive ARGument of Knowledge
Correctness and Soundness

Verifier

\[ y \neq f(x) \]

Corrupted Prover
Extra Properties for the Proof

Succinct Proof  Efficient Verification  Knowledge Soundness
Pre-Processing for Efficient Arguments

- [Mic00] Computationally sound proofs
  - Silvio Micali

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

- [DCL08] Succinct NP proofs from an extractability assumption
  - G. Di Crescenzo
  - Helger Lipmaa

  - J. Groth
One round Interaction

Verifier

Prover

π

crs

crs
Strong Assumptions

- [Mic00] Computationally sound proofs
  - Silvio Micali

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

- [DCL08] Succinct NP proofs from an extractability assumption
  - G. Di Crescenzo
  - Helger Lipmaa

  - J. Groth

- [GW11] Separating succinct non-interactive arguments from all falsifiable assumptions
  - Craig Gentry
  - Daniel Wichs
State-of-the-art

Pinocchio: Nearly practical verifiable computation
B. Parno, J. Howell, C. Gentry, M. Raykova

QSP and succinct NIZKs without PCPs
R. Gennaro, C. Gentry, B. Parno, M. Raykova

SNARGs via linear interactive proofs
B. Parno, J. Howell, C. Gentry, M. Raykova

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

[PHGR13]
[GGPR13]
[BCI+13]
SNARK with Preprocessing

\[ \text{Gen}(1^\lambda, \mathcal{R}) \rightarrow (\text{crs}, \text{vk}) \]

\[ \text{Prove}(\text{crs}, y, w) \rightarrow \pi : (y, w) \in \mathcal{R} \]

\[ \text{Verify}(\text{vk}, y, \pi) \rightarrow 0/1 \]
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

\[ A_{\text{crs}} \]

\[ \pi \]

\[ (y, w) \in R \]

\[ w \]

\[ \varepsilon_{\text{crs}} \]

Adversary

extractor

\[ w \]
Zero-Knowledge SNARK

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

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

**Knowledge Soundness**
a witness can be efficiently extracted from the prover

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

**Argument**
soundness holds only against computationally bounded provers
Zero-Knowledge
Zero-Knowledge

\[(\text{crs, } \text{vk})\]

\text{trapdoor} \\
\text{Simulator} \\
\pi' \\
\text{trapdoor, } y \\
\approx \\
\pi \\
\text{(y, w) } \in \mathcal{R} \\
\mathcal{w}
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 $\pi$

$\text{SSP} \left\{ v_i(w) \right\}_{i=0}^m \circ \circ \ 
t(x)$
### SNARK: Methodology

#### Target Statement

\[ R(y, w) = 1 \]

#### Computational Model (Representation)

- **Computation** \( y = F(x) \)
- **PCP**: Probabilistically Checkable Proofs
- **QSP / SSP**: Quadratic / Square Span Programs
- **QAP / SAP**: Q / S Arithmetic Programs

#### SNARK under Knowledge Assumptions

- **ECRH**: Extractable Collision-Resistant Hash
- **PKE**: Power Knowledge of Exponent
Computation: Circuit SAT

Verifier

Claim $f(x) = y$

Prover

NP statement $f(x) = y$
NP witness: Too long!

Verifier

Prover

Witness for Circuit SAT

NP statement

\[ f(x) = y \]
Solve equivalent problem instead

Veriﬁer

Circuit SAT solution

Prover

Polynomial problem
Given \( v(x) \), \( t(x) \).
Find \( P(x) \) such that
\[
P(x)t(x) = v(x)
\]
Solve equivalent problem instead

Polynomial problem
Given \( v(x), t(x) \).
Find \( P(x) \) such that
\[
P(x)t(x) = v(x)
\]

\[
P(x) = \sum p_i x^i
\]

Coefficients of solution \( P(x) \)
\[
p_0, p_1, p_2, \ldots, p_d
\]

Verifier

Prover
Solution as big as witness for Circuit SAT

Not Succinct

\[ P(x) = \sum p_i x^i \]

Coefficients of solution \( P(x) \)

\( p_0, p_1, p_2, \ldots p_d \)

Verifier

Prover
Evaluate polynomial in one point $s$

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

Verifier

Coefficients of solution $P(x)$

$p_0, p_1, p_2, \ldots, p_d$

Prover
Evaluate polynomial in one point $s$

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

$P(x) t(x) = v(x)$  \[\rightarrow\]  $P(s) t(s) = v(s)$

Verifier

Prover
The evaluation point should be hidden

\[ P'(x) \neq P(x) \]
The evaluation point should be hidden.

Verifier

Prover

$\text{Enc}(s)$

$\text{P}'(s)$

$t(s) = v(s)$

$\text{P}(x)$
Encoding of evaluation point $s$

Verifier

\[ P(x) \]

Prover

\[ P(s) = ? \]

\[ \text{Enc}(s) \]
Encoding Properties

Encoding:
- **linearly** homomorphic

\[
\text{Enc}(s) \quad \text{Enc}(s^2) \quad \text{...} \quad \text{Enc}(s^d) \quad \text{Enc}(P(s)) = \sum p_i \quad \text{Enc}(s^i)
\]
Not Knowledge Sound

Verifier

Not an Argument of Knowledge!

Prover

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \ldots \quad \text{Enc}(s^d) \]

Proof \( \pi = \text{Enc}(P(s)) \)
Extraction from the proof

\[ \pi = \text{Enc}(P(s)) \]

Verifier

Prover

Coefficients of \( P(x) \)
\( p_1, p_2, p_3, \ldots p_d \)
Non-falsifiable Assumption: Power Knowledge of Exponent

\[ \text{Enc}(s) \cdot \text{Enc}(s^2) \cdot \ldots \cdot \text{Enc}(s^d) \cdot \text{Enc}(\alpha s) \cdot \text{Enc}(\alpha s^2) \cdot \ldots \cdot \text{Enc}(\alpha s^d) \]

\[ = \text{Enc}(\sum p_i s^i) \]
Preprocessing: Double the Proof

\[
\begin{align*}
\text{Veriﬁer} & \quad \text{Prover} \\
\text{Enc}(s) & \quad \text{Enc}(\alpha) \\
\text{Enc}(s^2) & \quad \text{Enc}(\alpha s^2) \\
\text{Enc}(\alpha s) & \quad \text{Enc}(\alpha s^2) \\
\text{Enc}(s^d) & \quad \text{Enc}(\alpha s^d) \\
\end{align*}
\]

\[\pi = \pi P \overset{\pi}{P} = \text{Enc}(\alpha \text{P}(s)) = \text{Enc}(\alpha \text{P}(s))\]
Verification

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

Encoding:
- linearly homomorphic
- quadratic root detection
- image verification
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

\[ \{v_i(x)\}_i, \{w_i(x)\}_i, \{y_i(x)\}_i, t(x) \]
Build a table to interpolate polynomials

<table>
<thead>
<tr>
<th>Left inputs</th>
<th>Right inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>a₁ a₂</td>
<td>a₃ a₄</td>
<td>yᵢ</td>
</tr>
<tr>
<td>r₅</td>
<td>v₅</td>
<td>w₅</td>
</tr>
<tr>
<td>a₅</td>
<td></td>
<td>y₆</td>
</tr>
<tr>
<td>r₆</td>
<td>v₆(𝑟₅) = 1</td>
<td>w₆(𝑟₅) = 1</td>
</tr>
<tr>
<td>vᵢ(𝑟₆) = 0, i ≠ 3</td>
<td>wᵢ(𝑟₆) = 0, i ≠ 4</td>
<td>yᵢ(𝑟₆) = 0, i ≠ 5</td>
</tr>
<tr>
<td>vᵢ(𝑟₆) = 0, i ≠ 1,2</td>
<td>wᵢ(𝑟₆) = 0, i ≠ 5</td>
<td>yᵢ(𝑟₆) = 0, i ≠ 6</td>
</tr>
</tbody>
</table>
Division property: Common Roots $r_5^*, r_6$

<table>
<thead>
<tr>
<th></th>
<th>$v_3(r_5) = 1$</th>
<th>$w_4(r_5) = 1$</th>
<th>$y_5(r_5) = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1(r_6)$</td>
<td>$v_2(r_6) = 1$</td>
<td>$w_5(r_6) = 1$</td>
<td>$y_6(r_6) = 1$</td>
</tr>
</tbody>
</table>

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

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.

\[ V(x) = \sum_{i=0}^{m} a_i v_i(x) \]

and \( t(x) h(x) = V(x) W(x) - Y(x) \)
Square Span Programs

SSP

Find \( h(x) \)

\( h(x) = p(x) \)

[DFGK14]

\[
\{ v_i(x) \}_{i=0}^m = l(x)
\]
Step 1: Linearization of logic gates

Find $h(x)$

\[ h(x) = p(x) \]

Square Span Program

\[ -2a_1 - 2a_2 + 4a_4 \in \{0,2\} \]

\[ 2a_3 + 2a_4 - 4a_5 \in \{0,2\} \]

\[ L(a_i) = v_{0j} + \sum_{i=1}^{m} a_i v_{ij} \in \{0,2\} \]
### Step 2. Square constraint

<table>
<thead>
<tr>
<th>OR gate</th>
<th>AND gate</th>
<th>XOR gate</th>
<th>Output = 1</th>
<th>Entries = bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-a - b + 2c \in {0,1})</td>
<td>(a + b - 2c \in {0,1})</td>
<td>(a + b + c \in {0,2})</td>
<td>(3 - 3c \in {0,1})</td>
<td>(2a, 2b \in {0,2})</td>
</tr>
</tbody>
</table>

- \(a + b + c \in \{0,2\}\)
- \(3 - 3c \in \{0,1\}\)
- \(2a, 2b \in \{0,2\}\)

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

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

\[
\left( \lor_a + \delta - 1 \right) \circ \left( \lor_a + \delta - 1 \right) = 1
\]
Step 3. Polynomial Interpolation

\[
\begin{align*}
\left( v_0(r_j) + \sum_{i=1}^{m} a_i v_i(r_j) \right)^2 - 1 &= 0 \\
v_0(r_j) &= \delta_j - 1 \\
v_i(r_j) &= V_{ji}
\end{align*}
\]

\[\forall \{ r_j \} \in \mathbb{F}^d \]
Step 4. Polynomial Problem SSP

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

\[
\forall \{r_j\} \in \mathbb{R}^d
\]

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

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 $\mathbb{G}$

$\langle g \rangle = \mathbb{G}, \ E(nc(s)) = g^s$

$g^s, g^{s^2}, \ldots, g^{s^d}$

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

$g^{\sum_i p_i s^i} = \prod (g^{s^i})^{p_i}$
DLog Encoding vs Encryption Scheme

**DLog Group** \( \mathbb{G} \)

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

**Encryption:** \( E_{pk}(m) = c \)

**Decryption:** \( D_{sk}(c) = m \)

\[ Enc(p(s)) = g^{p(s)} \]

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

\[ E_{pk}(\sum_i p_i s^i) = \sum p_i E_{pk}(s^i) \]
Quadratic root detection public

\[ t(s) h(s) = p(s) \]

\[ e(g^{t(s)}, g^{h(s)}) = e(g^{p(s)}, g) \]
Publicly Verifiable vs Designated Verifiable

Publicly Verifiable:

\[
\langle g \rangle = G, \quad \langle \tilde{g} \rangle = \tilde{G}
\]

\[
Enc(s) = g^s \quad e : G \times G \rightarrow \tilde{G} \quad e(g^a, g^b) = \tilde{g}^{ab}
\]

Quadratic root detection public

\[
t(s)h(s) = p(s)
\]

\[
e(g^{t(s)}, g^{h(s)}) = e(g^{p(s)}, g)
\]

Designated Verifiable:

Encryption:

\[
E_{pk}(m) = c
\]

Decryption:

\[
D_{sk}(c) = m
\]

Quadratic root detection needs sk

\[
t(s)h(s) = p(s)
\]

\[
p(s) \quad h(s)
\]

\[
E(p(s)) \quad E(h(s))
\]
Assumption on Discrete Log Encoding

\[ g^s \cdot g^{s^2} \cdots g^{s^d} \]

\[ g^{\alpha s} \cdot g^{\alpha s^2} \cdots g^{\alpha s^d} \]

\[ \varepsilon \]

\[ g^{p_1} \cdot g^{p_2} \cdots g^{p_d} = g^{\sum p_i s^i} \]
SNARK from SSP

Proof:
Evaluate in a point
\[ p(s), h(s) \]

Find \( h(x) \)
\[ t(x)h(x) = p(x) \]

Verify the proof
\[ t(s)h(s) = p(s) \]
\[ p(s) = V(s)^2 - 1 \]
Evaluate solution in $s$

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

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

$$t(x)h(x) = V(x)^2 - 1$$
Evaluate in a point $V(s), h(s)$

**Enforce Linear Span**

$SSP$

$v_0(x), v_1(x), ..., v_m(x)$

$\pi (x)$

$V(x), h(x) = \pi = \{Enc(V(s)), Enc(h(s)), Enc(\beta V(s))\}$

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

$t(x) h(x) = V(x)^2 - 1$
Not an Argument of Knowledge!
Setup and Proof

 SSP: \( t(x) \)

\( v_i(x), v(x) - v_m(x) \)

\[ \text{Enc}(\alpha s) \]
\[ \text{Enc}(\alpha s^2) \]
\[ \ldots \]
\[ \text{Enc}(\alpha s^d) \]

\[ \text{Enc}(\beta v_i(s)) \]

\[ \pi = \text{Enc}(V(s)) \]
\[ \text{Enc}(h(s)) \]
\[ \text{Enc}(\alpha V(s)) \]
\[ \text{Enc}(\alpha h(s)) \]
\[ \text{Enc}(\beta V(s)) \]
Verifier

\[ t(s)h(s) = p(s) \]

\[ \pi = \begin{array}{c} \hat{W} \\ \hat{H} \end{array} = \begin{array}{c} W \\ H \end{array} \]

Verifier
Verifier

Verify the proof

Verify

\[ t(s)h(s) = p(s) \]
Verifier

\[ \pi = \text{Verify} \]

Verifying the proof

\[ t(s)h(s) = p(s) \]
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)

\[
\delta = \mathcal{H}(\mathcal{W}) - 1
\]
Security Analysis
Assumption PDH: Power Diffie–Hellman

\[ \text{d-PDH} \]

\[ \text{Enc}(s), \text{Enc}(s^2), \text{Enc}(s^d), \text{Enc}(s^{d+2}), \text{Enc}(s^{2d}) \]
Security Reduction: Cheating Strategy

\[ \pi \Rightarrow \begin{array}{c} W \quad H \\ \hat{W} \quad \hat{H} \end{array} \quad B \]

\[ V_1 \quad V_2 \quad \ldots \quad V_d \Rightarrow V(x) \]

\[ h_1 \quad h_2 \quad \ldots \quad h_d \Rightarrow h(x) \]

Solve d-PDH

Cheating Proof
Security Reduction: Cheating Strategy

\[ \pi = \begin{bmatrix} W & H \\ \hat{W} & \hat{H} \end{bmatrix} \]

\[ V(x) \quad h(x) \]

Solve d-PDH

Cheating Proof
Polynomial Division does not Hold

\[ \pi = \frac{W}{H} = \frac{\hat{W}}{\hat{H}} = B \]

\[ V(x) \quad h(x) \]

\[ t(s)h(s) = V(s)^2 - 1 \]

- \[ t(x)h(x) \neq V^2(x) - 1 \text{, but} \]

\[ \text{Enc}(t(s)) \times H = W^2 - 1 \]
Not in the Proper Span

\[ \pi = \begin{bmatrix} W & H \\ \hat{W} & \hat{H} \end{bmatrix} B \]

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

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

\[ \begin{bmatrix} B \\ \beta \end{bmatrix} = W \]
Reduction to d-PDH

d-PDH

\[ \text{Enc}(s) \rightarrow \text{Enc}(s^2) \rightarrow \ldots \rightarrow \text{Enc}(s^d) \]

\[ ? \rightarrow \text{Enc}(s^{d+2}) \rightarrow \ldots \rightarrow \text{Enc}(s^{2d}) \]
Reduction to d-PDH

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \ldots \quad \text{Enc}(s^d) \]

\[ ? \quad \text{Enc}(s^{d+2}) \quad \ldots \quad \text{Enc}(s^{2d}) \]

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \ldots \quad \text{Enc}(s^d) \]

\[ \text{d-PDH} \]

\[ \times \alpha \]
Reduction to d-PDH

\[ \{ v_i(x) \}_{i=0,m} \]

\[ t(x) \]

Enc(s)  Enc(s^2)  \ldots  Enc(s^d)

Enc(s^{d+2})  \ldots  Enc(s^{2d})

SSP

Crs

\{ Enc(s) \}  \ldots
Reduction to d-PDH

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \cdots \quad \text{Enc}(s^d) \]

\[ \text{Enc}(s^{d+2}) \quad \cdots \quad \text{Enc}(s^{2d}) \]

\[ \text{Enc}(\beta v_i(s)) \]

\[ \{ \text{C} \} \]

\[ \{ \text{Envelopes} \} \]

\[ \{ \text{Envelopes} \} \]

\[ \{ \text{Envelopes} \} \]
Reduction to d-PDH

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \ldots \quad \text{Enc}(s^d) \]

\[ \text{Enc}(s^{d+2}) \quad \ldots \quad \text{Enc}(s^{2d}) \]

\[ \text{crs} \quad \text{SSP} \]

\{ \text{Crs} \}
Reduction to d-PDH

\[ \text{d-PDH} \]

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \cdots \quad \text{Enc}(s^d) \]

\[ \text{Enc}(s^{d+2}) \quad \cdots \quad \text{Enc}(s^{2d}) \]

\[ \text{Reduction to d-PDH crs} \]
Reduction to d-PDH

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

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

d-PDH

V(x)  h(x)
Reduction to $d$–PDH

\[ t(x)h(x) \neq V(x)^2 - 1, \text{ but } t(s)h(s) = V(s)^2 - 1 \]
Reduction to $d$-PDH

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

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

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

\[ p_{d+1} \cdot Enc(s^{d+1}) = - \sum_{i=1,...,d}^{d+2,...,2d} p_i \cdot Enc(s^i) \]
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