SNARKs Tutorial: Introduction & Circuit-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 nondeterministic verifier) has been an intriguing question for the past two decades. Other than CS proofs in the random oracle model [Micah, 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 ’08].
SNARK

Background

Verification Framework for SNARKs

Construction Security

The END

Verifier

Prover
Delegated Computation

Verifier computes \( f(x) = y \)
Prover claims a statement

Verifier

Claim
\[ y = f(x) \]

Prover
Verifier does not trust

Verifier

\[ f(x) = y \]

Corrupted Prover

\[ y \neq f(x) \]
Proof Systems: Since the 1980s

A note on efficient zk-proofs and arguments

Joe Kilian

[Kil92]
Interactive Proof Protocol [Kil92]

Verifier

Prover

Claim

$y = f(x)$
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

Corrupted Prover

\[ y = f(x) \]
Correctness and Soundness

Verifier

Corrupted Prover

Verify

$\pi$

$y \neq f(x)$
Extra Properties for the Proof

Succinct Proof

Efficient Verification

Knowledge Soundness
Pre-Processing for Efficient Arguments

- [Mic00] Computationally sound proofs
  - Silvio Micali

  - J. Groth

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

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

Verifier

π

Prover

crs

crs
Strong Assumptions

- [Mic00] Computationally sound proofs
  - Silvio Micali

  - J. Groth

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

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

- [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

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

- **[GGPR13]**
  - QSP and succinct NIZKs without PCPs

- **[PHGR13]**
  - Pinocchio: Nearly practical verifiable computation
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

\[ A \]

\[ \varepsilon \]

Adversary

\[ \pi \]

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

\[ w \]

extractor

\[ w \]
Zero-Knowledge 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

\((crs, vk)\)

trapdoor

Simulator

\(\pi'\)

\(\pi\)

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

\(w\)

Prover
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
Pre-Processing:

\( \text{crs}: \) common reference string

- Secret coins
- Expensive
- Subversion

Efficient Protocols from Knowledge Assumptions
Frameworks

SNARK Background

Frameworks for SNARKs

Construction Security

The END

SNARKs

SSP

\[ \{ v_i(w) \}_{i=0}^{m} \]

SNARK (π)

34
### SNARK: Methodology

#### Target Statement
\[ R(y,w) = 1 \]

#### Computational Model (Representation)
- **Computation** \( y = F(x) \)
- **Boolean Circuit SAT**
- **Arithmetic Circuit SAT**

#### SNARK under Knowledge Assumptions
- **PCP**: Probabilistically Checkable Proofs
- **QSP / SSP**: Quadratic / Square Span Programs
- **QAP / SAP**: Q / S Arithmetic Programs
- **ECRH**: Extractable Collision-Resistant Hash
- **PKE**: Power Knowledge of Exponent
Computation: Circuit SAT

Verifier

Claim \( f(x) = y \)

Prover
NP witness: Too long!

Verifier

Prover

Witness for Circuit SAT

NP statement

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

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

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

Coefficients of solution $p(x)$:
$p_0, p_1, p_2, \ldots p_d$

Witness for Circuit SAT

Not Succinct
Evaluate polynomial in one point $s$

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

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

Verifier

Prover
Evaluate polynomial in one point $s$

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

$P(x) t(x) = v(x)$

$P(s) t(s) = v(s)$
The evaluation point should be hidden

\[ P'(s) = P(s) \]

\[ P'(x) \neq P(x) \]

Verifier

Prover
The evaluation point should be hidden

Verifier

Prover

Solves

\[ t(s) = v(s) \]
The evaluation point should be hidden
Encoding of evaluation point $s$ – not enough!
Encoding Properties

\[ \text{Enc}(\text{s}) \quad \text{Enc}(\text{s}^2) \quad \text{Enc}(\text{s}^d) \quad \ldots \quad \text{Enc}(\text{P}(\text{s})) = \sum p_i \quad \text{Enc}(\text{s}'_i) \]

Encoding:
- **linearly** homomorphic

Verifier

Prover
Not Knowledge Sound

\[ \text{Enc}(s) \rightarrow \text{Enc}(s^2) \rightarrow \ldots \rightarrow \text{Enc}(s^d) \rightarrow \text{Proof } \pi = \text{Enc}(P(s)) \]

Verifier

Not an Argument of Knowledge!

Prover
Extraction from the proof

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

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

d-PKE

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

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

\[ \text{Enc}(P) \quad \text{Enc}(\alpha P) \]

\[ p_1 \quad p_2 \quad \ldots \quad p_d \quad \text{Enc}(P) \]

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

Verifier

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

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

Prover
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

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

\[
\begin{align*}
QAP & \{v_i(x)\}_i, \{w_i(x)\}_i, \\
& \{y_i(x)\}_i, t(x)
\end{align*}
\]
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>$v_3(r_5) = 1$</td>
<td>$w_4(r_5) = 1$</td>
<td>$y_5(r_5) = 1$</td>
</tr>
<tr>
<td>$v_1(r_6) = v_2(r_6) = 1$</td>
<td>$w_5(r_6) = 1$</td>
<td>$y_6(r_6) = 1$</td>
</tr>
<tr>
<td>$v_i(r_5) = 0, \ i \neq 3$</td>
<td>$w_i(r_5) = 0, \ i \neq 4$</td>
<td>$y_i(r_5) = 0, \ i \neq 5$</td>
</tr>
<tr>
<td>$v_i(r_6) = 0, \ i \neq 1,2$</td>
<td>$w_i(r_6) = 0, \ i \neq 5$</td>
<td>$y_i(r_6) = 0, \ i \neq 6$</td>
</tr>
</tbody>
</table>
Division property: Common Roots \( r_5, r_6 \)

Left inputs

\[ a_1 a_2 \]

\[ a_3 a_4 \]

\[ r_5 \]

\[ r_6 \]

\[ a_5 a_6 \]

Right inputs

\[ v_i \]

\[ w_i \]

Outputs

\[ y_i \]

\[ v_3(r_5) = 1 \]

\[ w_4(r_5) = 1 \]

\[ y_5(r_5) = 1 \]

\[ v_1(r_6) = v_2(r_6) = 1 \]

\[ w_5(r_6) = 1 \]

\[ y_6(r_6) = 1 \]

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

58
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

Find $h(x)$

$\mathbf{SSP}$

$\{v_i(x)\}_{i=0}^{m}$

$t(x)$

[DFGK14]
Step 1: Linearization of logic gates

Square Span Program

Find \( h(x) \)

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

Square Span Program

\[ a_1 \quad a_2 \quad a_3 \]

\[ a_4 \quad a_5 \]

\[ \begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array} \]

\[ \begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array} \]

\(-2a_1-2a_2+4a_4 \in \{0,2\}\)

\(2a_3+2a_4-4a_5 \in \{0,2\}\)

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

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

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

$$\left( \bigvee a + (\delta - 1) \right) \circ \left( \bigvee a + (\delta - 1) \right) = 1$$
Step 3. Polynomial Interpolation

\[
V^m a^{\delta-1} \circ (V^a + \delta - 1) = 1
\]

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

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

\[
v_0(r_j) = \delta_j - 1 \quad \quad v_i(r_j) = V_{ji}
\]
Step 4. Polynomial Problem SSP

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

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

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

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

\[
\begin{align*}
  t(x) & \mid V(x)^2 - 1 \\
  V(x) & = v_0(x) + \sum_{i=1}^{m} a_i v_i(x) \\
  t(x) & = \prod_{j=1}^{d} (x - r_j)
\end{align*}
\]
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 Instantiations: Linearity

**DLog Group** $\mathbb{G}$

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

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

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

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

DLog Group $\mathbb{G}$

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

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

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

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

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

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

$E_{pk}(\sum p_i s^i) = \sum p_i E_{pk}(s^i)$
Quadratic Root Detection - Pairings

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

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

Quadratic root detection **public**

\[ t(s) h(s) \overset{?}{=} p(s) \]

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

Publicly Verifiable

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

Quadratic root detection **public**

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

Designated Verifiable

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

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

Quadratic root detection needs **sk**

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

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

\[ g^s, g^{s^2}, \ldots, g^{s^d} \]

\[ g^{s\alpha}, g^{\alpha s^2}, \ldots, g^{\alpha s^d} \]

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

Circuit for \( f(\cdot) \)

Proof:
Evaluate in a point

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

Square Span Program

Verify the proof

\( t(s)h(s)=p(s) \)
\( p(s)=V(s)^2-1 \)
Evaluate solution in $s$

$V(s), h(s)$

$SSP$

$v_0(x), v_1(x), …, v_m(x)$

$t(x)$

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

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

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

$Proof$

$\pi$

$Enc(V(s))$, $Enc(h(s))$
Enforce Linear Span

SSP

\[ v_0(x), v_1(x), \ldots, v_m(x) \]

\[ t(x) \]

\[ V(x), h(x) = \]

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

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

\[ \pi \]

\[ \text{Enc}(V(s)) \]

\[ \text{Enc}(h(s)) \]

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

\[ i = 0, m \]
Evaluate in a point $V(s)$, $h(s)$

$Enc(s^2)$

$SSP v(x), v_1(x), ..., v_m(x)$

$t(x)$

$\pi$

$Enc(h(s)) Enc(V(s)) = Enc(\beta v_i(s))$

$i = 0, m$

Not an Argument of Knowledge!
Setup and Proof

Evaluate in a point $V(s), h(s)$

SSP: $t(x)$
$v_0(x), v_1(x), \ldots, v_m(x)$

$\text{Enc}(\alpha s)$
$\text{Enc}(\alpha s^2)$
$\text{Enc}(\alpha s^d)$

$\text{Enc}(\beta v_i(s))$

$i = 0, m$

$\pi = \text{Enc}(V(s)) = \text{Enc}(h(s)) = \text{Enc}(\alpha V(s)) = \text{Enc}(\alpha h(s)) = \text{Enc}(\beta V(s))$

$\text{Enc}(s)$
$\text{Enc}(s^2)$
$\text{Enc}(s^d)$
Verifier

Verify the proof

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

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

Verify the proof

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

Verifier

\[ B \overset{=}{} W = \beta \]
Verifier

Verify the proof

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

\[ \pi \]

\[ \text{Enc}(t(s)) \]

\[ W \]
\[ \hat{W} \]
\[ H \]
\[ \hat{H} \]
\[ B \]

\[ W^2 \]

\[ -1 \]
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)

\[ \text{Gen}(\lambda, R) \]

\[ \text{P}(\text{crs}, y, w) \]

\[ \text{V}(\text{vk}, y, \pi) \]

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

\[ t(s) \times H = W \times W \times W - 1 \]
Security Analysis
Assumption PDH: Power Diffie–Hellman

\[ \text{Enc}(s) \quad \text{Enc}(s^2) \quad \text{Enc}(s^d) \quad \text{Enc}(s^{d+1}) \quad \text{Enc}(s^{d+2}) \quad \text{Enc}(s^{2d}) \]
Security Reduction: Cheating Strategy

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

\[ V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_d \rightarrow V(x) \]

\[ h_1 \rightarrow h_2 \rightarrow \ldots \rightarrow h_d \rightarrow h(x) \]

Solve d-PDH

Cheating Proof
Security Reduction: Cheating Strategy

\[ \pi = W \text{ V}(x) \quad H \text{ h}(x) \]

\[ \hat{W} \hat{H} B \]

Solve d-PDH

Cheating Proof
Polynomial Division does not Hold

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

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

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

\[ \pi = \begin{bmatrix} W & H \\ \hat{W} & \hat{H} \end{bmatrix} \begin{bmatrix} B \\ V(x) \end{bmatrix} \]

\[ h(x) \]

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

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

\[ \beta \]

\[ W \]
Reduction to d-PDH

d-PDH

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

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

{Crs}

89
Reduction to d-PDH

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

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

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

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

\[
\text{SSP}
\]

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

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

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

\[ \text{CrS} \]

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

d-PDH

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

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

SSP

Crs
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
Reduction to d-PDH

\[ \text{d-PDH} \]

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

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

\[ \text{V}(x) \quad \text{h}(x) \]
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

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

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

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

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