SNARKs & QA-NIZKs with a Subverted Setup

Behzad Abdolmaleki

University of Tartu, Estonia

26.011.2019 / UT Wien, Vienna
Interactive Proofs

- Prover wants to convince Verifier that $x$ has some property (i.e. “I am the doctor!” Or “$x$ is in language $L$”)

$pk, sk$

I am the doctor!

$pk$

Prove it!

$sk$

Commit

Challenge

Responds

OK, $x \in L$

Witness $w$ ($x, w) \in R_L$
Non-Interactive Proofs

- Prover wants to convince Verifier that “$x$ is in language $L$”.

Witness $w$  
$(x, w) \in R_L$

Proof

OK, $x \in L$

Non-Interactive proofs usually are constructed in two models:

- Random Oracle (RO) Model [Micali 94]

  - Commit
  - Challenge
  - Responds

  $P \quad \rightarrow \quad V$

  $P \quad \leftarrow \quad V$

  $P \quad \downarrow \quad V$

  $\text{Com, Hash(Com), Responds}$

- Common Reference String (CRS) Model

  - Trusted Third Party (TTP) generates CRS
Non-interactive Zero-knowledge (NIZK)

\[ x \in L, w \]

\[ \pi = P(x, w) \]

Correctness

Soundness

Zero-knowledge

\[ V(x, \pi) \]

\[ V(x, \pi) \]

\[ V(x, \pi) \]
Non-interactive Zero-knowledge

$x \in L, w$

$\pi = P(x, w)$

$\pi \approx \pi^*$

$\pi^* = \text{Sim}(x)$

$V(x, \pi)$?

$V(x, \pi)$?
Non-interactive Zero-knowledge

- $P$ should be able to compute $\pi$ only when knowing valid $w$
- $Sim$ should be able to compute $\pi$ without knowing $w$
- $Sim$ must have some extra power

$x \in L, w$

$\pi = P(x, w)$

$\pi \approx \pi^*$

$\pi^* = Sim(x)$

$V(x, \pi)$?
NIZK in CRS model

\[ x \in L, w \]

\[ \pi = P(crs, x, w) \]

\[ \pi \approx \pi^* \]

\[ \pi^* = Sim(td, x) \]

\[ \mathcal{V}(crs, x, \pi) \]

\[ \mathcal{V}(crs, x, \pi) \]
Special Type of NIZKs: zk-SNARKs

Witness $w$ \[(x, w) \in R_L\]

$(CRS, td) \leftarrow CRSGen(1^n)$

proof $\leftarrow$ Prove$(CRS, \text{witness})$

\{1, 0\} $\leftarrow$ Verify$(CRS, \text{word, proof})$

We are interested to make them succinct as much as possible.

**Succinct**: Small size of proof and low verification complexity


Proof = 3 group elements
Inventions in NIZK: zk-SNARKs

- for whole NP without costly reductions
  - For known arithmetic circuit $C$ & partially unknown $x, y = C(x)$
- Very strong (non-black-box) cryptographic assumptions
  - Knowledge assumption: if $P$ outputted $x$ then $P$ must know $y$
    - [Gentry Wichs 2011]: non-black-box assumptions needed
- The CRS depends on concrete circuit
  - need to generate new CRS for each circuit!

- Applications: verifiable computation, privacy-preserving cryptocurrencies, ...
What If CRS Generator is Malicious?
Subversion security: Trust issues

Interested in **zero knowledge**
crs should contain trusted $\text{crs}_{zk}$

Interested in **soundness**
crs should contain trusted $\text{crs}_{snd}$

$$\pi = P(\text{crs}, x, w)$$

Need to trust $\text{crs}_{zk}$

Need to trust $\text{crs}_{snd}$

$$V(\text{crs}, x, \pi)$$

$\pi = P(\text{crs}, x, w)$

$(\text{crs}_{zk}, \text{crs}_{snd}), \text{td}$
Subversion security: Trust issues

Subversion ZK: zero knowledge even if CRS creator is malicious

Subversion SND: soundness even if CRS creator is malicious

Interested in zero knowledge
- crs should contain trusted crs\textsubscript{zk}

Interested in soundness
- crs should contain trusted crs\textsubscript{snd}

\( \pi = P (\text{crs}, x, w) \)

Research direction: minimize the needed trust

\( V (\text{crs}, x, \pi) ? \)

\((\text{crs}_{\text{zk}}, \text{crs}_{\text{snd}}), \text{td} \)

(x, w)
Sub-ZK in CRS model

If crs creator is malicious, there is no guarantee that:
- crs is correct
- td exists
- crs is from correct distribution
Impossibility Result

- [Bellare, Fuchsbauer, Scafuro 2016]

- [Abdolmaleki, Baghery, Lipmaa, Zajac 2017]
  - Efficient SND + Sub-ZK SNARK for NP
  - Asiacrypt 2017 “top three” paper
  - [Fuchsbauer 2018]: related/improved constructions

<table>
<thead>
<tr>
<th></th>
<th>WI</th>
<th>ZK</th>
<th>Sub-ZK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SND</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Sub-SND</td>
<td>Possible</td>
<td>Impossible</td>
<td>Impossible</td>
</tr>
</tbody>
</table>
[ABLZ17] recipe for Sub-ZK SNARK

- Design a **public** algorithm $CV$ for checking $crs$ is correct
  - If $CV(crs) = 1$: there exists some $td$

**Proving Sub-ZK**

If $CV(crs) = 0$: no need to simulate
If $CV(crs) = 1$:
  - Use extractor $Ext$ to recover (unique) $td$ from $crs$
  - Simulate by using extracted $Ext$
  - $Ext$ exists by assumption
Sub-ZK SNARK

Witness \( w \) \((x, w) \in R_L\)

\((\text{CRS}, \text{td}) \leftarrow \text{CRSGen}(1^n)\)

\( \text{OK, } x \in L \)
Witness \( w \) 
\((x, w) \in R_L\)

\[(CRS, td) \leftarrow \text{CRSGen}(1^n)\]

\(\text{proof} \leftarrow \text{Prove}(CRS, \text{witness})\)

\(\{1, 0\} \leftarrow \text{Verify}(CRS, \text{word, proof})\)

If \( 1 \leftarrow \text{CV}(crs) \)

Then

\(\text{proof} \leftarrow \text{Prove}(CRS, \text{witness})\)

OK, \( x \in L \)
Summary of SNARK Branch

- Privacy-preserving technologies
- Interactive
- Non-interactive
- Random Oracle Model
- CRS Model
- zk-SNARKs
- Updatable zk-SNARKs
- Subversion zk-SNARKs

ZK proofs

Privacy-preserving Coins (Zcash,...), Smart Contracts (HAWK,...), E-voting ....
More Faster NIZKs for Linear Language:

Quasi-Adaptive NIZK (QA-NIZK)
Quasi-Adaptive NIZK

Witness \( (x, w) \in R_L \)

\( \rho \)

CRS

proof

OK, \( x \in L \)

- Language parameter \( \rho \) is chosen before the CRS
- \( \rho \) cannot adaptively depend on the CRS
- Usually, \( \rho = \) some public key

• Applications in constructing efficient cryptographic primitives
  (commitment schemes, IBE, signature schemes, …)

QA-NIZK

- “[Kiltz, Wee 2015]
  - most efficient known QA-NIZK for SUBSPACE language

Task of QA-NIZK for SUBSPACE:
- Fix language parameter $[\rho]_1 \in G^{n \times m}$
- Prove in zero knowledge that $[\hat{x}]_1 = [\rho]_1 \overline{w}$ for some $\overline{w} \in \mathbb{Z}_p^m$

$$L = \{[\hat{x}]_1 \in G_1^n \mid \exists \overline{w} \in \mathbb{Z}_p^m \text{ s.t. } [\hat{x}]_1 = [\rho]_1 \overline{w}\}$$
Our recipe: Sub-QA-NIZK [ALSZ18]

- Design a public algorithm PKV for checking crs is correct
  - If PKV(ρ, crs) = 1: there exists some td

- Proving Sub-ZK
  - If PKV(ρ, crs) = 0: no need to simulate
  - If PKV(ρ, crs) = 1:
    - Use extractor Ext to recover td from crs Simulate by using extracted Ext
    - Ext exists by KWKE assumption.

- Extraction of td requires non-black-box “knowledge assumption”
Our recipe: Sub-QA-NIZK [ALSZ18]

- Design a public algorithm PKV for checking crs is correct
  - If PKV(ρ, crs) = 1: there exists some td
- Proving Sub-ZK
  - If PKV(ρ, crs) = 0: no need to simulate
  - If PKV(ρ, crs) = 1:
    - Use extractor Ext to recover td from crs
    - Simulate by using extracted Ext
  - Ext exists by KWKE assumption.

- Extraction of td requires non-black-box knowledge assumption.

Weakness of the current Sub-QANIZK and open problems:
- It only works for some especial construction and for extraction it needs some invertible matrix (crs elements)
- It does not fit with Simulation Sound Extractable property
- The PKV is not general and for different security parameter one needs to define a new algorithm PKV.
Recent Inventions in NIZK

<table>
<thead>
<tr>
<th>SNARK</th>
<th>QA-NIZK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

succinct non-interactive argument of knowledge

quasi-adaptive non-interactive zero knowledge
Recent Inventions in NIZK

<table>
<thead>
<tr>
<th></th>
<th>SNARK</th>
<th>Succinct QA-NIZK</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Versatility</strong></td>
<td>Whole NP</td>
<td>Limited class of languages</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>3 group elements</td>
<td>1 group element</td>
</tr>
<tr>
<td><strong>Assumptions</strong></td>
<td>Non-black-box (knowledge)</td>
<td>Very standard</td>
</tr>
</tbody>
</table>

[Groth 2010], [Lipmaa 2012], [Gennaro Gentry Parno Raykova 2013], [Parno Howell Gentry Raykova 2013], [Groth 2016], [Groth Maller 2017], [Abdolmaleki Baghery Lipmaa Zajac 2017], ...

Sub-zk-SNARK

Sub-zk-QA-NIZK

The Tree of Succinct NIZK

- **NIZK in CRS Model**
  - **QA-NIZKs**
    - **Updatable** QANIZKs [Lipmaa18]
    - **Sub**-QANIZKs [ALSZ18]
  - **zk-SNARKs**
    - **Updatable** zk-SNARKs [GKM+18]
    - **Sub**- zk-SNARKs [ABLZ17] [Fuc17]

- **COCO Framework** [DS19] Framework
- **Simulation Sound Extractable (SSE)** zk-SNARKs
The Tree of Succinct NIZK

NIZK in CRS Model

QA-NIZKs

zk-SNARKs

Ad-Hoc based

Updatable QANIZKs [Lipmaa18]

Sub-QANIZKs [ALSZ18]

Ad-Hoc based

Updatable zk-SNARKs [GKM+18]

Sub- zk-SNARKs [ABLZ17] [Fuc17]

COCO Framework [DS19] Framework

Simulation Sound Extractable (SSE)

zk-SNARKs

(SSE)-sub-QA-NIZK (Join with Daniel)

Modular based:
Modular Constructions of Updatable QANIZKs
(Join with Daniel)
The Tree of Succinct NIZK

- NIZK in CRS Model
  - QA-NIZKs
  - zk-SNARKs

Just in one slide!

Framework SSE
- zk-SNARKs
- Sub zk-SNARKs
- Updatable zk-SNARKs

(Join with Sebastian and Daniel)
The Tree of Succinct NIZK

- NIZK in CRS Model
  - QA-NIZKs
  - zk-SNARKs

- Post-Quantum QA-NIZKs
- Post-Quantum zk-SNARKs

- Designated Verifier zk-SNARKs: 
  - [GMNO18]
  - [Anca19]
Framework SE (Sub & Updatable) zk-SNARK

LAMASSU (Join with Sebastian and Daniel)

\[
L' = \{ (x, w) \in R \mid cpk = pk \cdot f (csk - sk) \} \\
x' = (x, cpk, pk), w' = (w, csk - sk) \\
CRS' = (crs_{snark}, pk, cpk)
\]

Updatable Signature

SE zk-SNARK
SE sub zk-SNARK
SE updatable zk-SNARK
Post-Quantum zk-SNARKs

- NIZK in CRS Model
  - Post-Quantum QA-NIZKs
  - Post-Quantum zk-SNARKs
      - Designated Verifier zk-SNARKs:
          - [GMNO18]
          - [Anca19]
Lattice Based (Designated Verifier) zk-SNARKs

Dlog group based SNARK

CRS = \( (g^\alpha, g^\beta, \ldots) \)

\[ \pi = (g^A, g^B, g^C) \]

\[ e(g_1^A, g_2^B) = e(g_1^C, g_2^\cdot) \]

Paring property

Lattice-based SNARK

CRS = \( (E(\alpha), E(\beta), \ldots) \)

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

\[ V(\text{crs}, x, \pi) \]

\( \pi \)}
**Lattice Based (Designated Verifier) zk-SNARKs**

---

<table>
<thead>
<tr>
<th>$K(1^λ, Γ)$</th>
<th>$E(s, m) \rightarrow ct$</th>
<th>$\text{Dec}(s, (c_0, c_1)) \rightarrow m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s \leftarrow \mathbb{Z}_q^*$</td>
<td>$a \leftarrow \mathbb{Z}_q^n$</td>
<td>$m := (c_0 \cdot s + c_1) \mod p$</td>
</tr>
<tr>
<td>return $s$</td>
<td>$\sigma := g^a; \ e \leftarrow \chi_\sigma$</td>
<td>return $m$</td>
</tr>
<tr>
<td>$ct := (-a, a \cdot s + pe + m)$</td>
<td>$\pi = (g^A, g^B, g^C)$</td>
<td>$\pi = (En(A), En(B), En(C))$</td>
</tr>
</tbody>
</table>

---

**CRS** = $(g^α, g^β, ...)$

**P(***crs, x, w)***

**V(***crs, x, π)***

---

**Lattice-based SNARK**

CRS = $(En(α), En(β), ...)$

---

**Designated Verifier**

---

Paring property

$e(g^A_1, g^B_2) = e(g^C_1, g^2)$

$D_{sk}(π), D_{sk}(crs) : A \cdot B = C. (....)$
The Tree of Post-Quantum zk-SNARKs

Succinct NIZK in CRS Model

Post-Quantum QA-NIZKs
- ?
  - Updatable QA-NIZK
  - Subversion QA-NIZK
  - SE QA-NIZK

Post-Quantum zk-SNARKs
- zk-SNARKs[GMNO18]:
  - Designated Verifier
  - Proof Size = 5 cipher text
  - SSP based (Boolean circuit)
- zk-SNARKs[Anca19]:
  - Designated Verifier
  - Proof Size = 2 cipher text
  - SPA based (Arithmetic circuit)

Publicly verifiable zk-SNARKs
- SE zk-SNARKs
- ?

Subversion zk-SNARKs
- ?

Updatable zk-SNARKs
- ?
Reference


Thank you