Lift-and-Shift: Obtaining Simulation Extractable Subversion and Updatable SNARKs Generically

Behzad Abdolmaleki¹ and Sebastian Ramacher² and Daniel Slamanig²
Young Researcher Crypto Seminar, 16.10.2020

¹University of Tartu, ²AIT Austrian Institute of Technology, Vienna
Introduction
Zero-knowledge Proofs

**NP-language** \( L \)

- Prover wants to convince verifier that some \( x \in L \)
- Without revealing information beyond the statement \( x \in L \)
- Define relation \( R_L : x \in L \iff \exists w : (x, w) \in R_L \)

![Diagram showing interaction between Prover and Verifier](image)
Making them Non-Interactive: CRS

Common reference string model

\[ \pi \leftarrow \text{Prove}(\text{crs}, x, w) \]

\[ \text{Verify}(\text{crs}, x, \pi) = 1 \Rightarrow x \in L \]
Important Properties

Prover cannot cheat

- Prover unable to produce valid proofs for $x \notin L$

  - Soundness

- Property desired by the verifier
Important Properties

Prover cannot cheat

- Prover unable to produce valid proofs for $x \notin L$

  > Soundness

- Property desired by the verifier

Verifier does not learn any information on witness $w$

- Real proofs cannot be distinguished from simulated proofs

  > Zero-knowledge

- Property desired by the prover
Important Properties

Proofs of Knowledge

• Special extractor can extract witness from proofs

  Knowledge Soundness
Important Properties

Proofs of Knowledge

- Special extractor can extract witness from proofs

  > Knowledge Soundness

Strong versions

- (Knowledge) Soundness also holds if adversary can query simulated proofs

  > Simulation (knowledge) soundness

- Also called simulation (sound) extractability (SE)
In a real world protocol:

- Adversary sees many different proofs
- Might be possible to turn proof $\pi$ for word $x$ into a proof $\pi' \neq \pi$
- Or worse: turn into a proof $\pi'$ for a different word $x' \neq x$
On Simulation Soundness

In a real world protocol:

- Adversary sees many different proofs
- Might be possible to turn proof $\pi$ for word $x$ into a proof $\pi' \neq \pi$
- Or worse: turn into a proof $\pi'$ for a different word $x' \neq x$

Hence

- Adversary may query proofs
- Must produce a proof not queried before

Similar issue for signatures: one-time EUF-CMA – EUF-CMA – strong EUF-CMA
NIZKs in the CRS Model

- Zero-knowledge contradicts extractor
- Knowledge soundness contradicts simulator
NIZKs in the CRS Model

- Zero-knowledge contradicts extractor
- Knowledge soundness contradicts simulator

They need to have more power

- Extractor gets extraction trapdoor
- Simulator gets simulation trapdoor
NIZKs in the CRS Model

TTP $\text{crs}$

Prover $(x, w)$

$\pi \leftarrow \text{Prove}(\text{crs}, x, w)$

Verifier $x$

$\text{Verify}(\text{crs}, x, \pi) = 1 \Rightarrow x \in L$

Extractor

Simulator
NIZKs in the CRS Model

TTP \( crs, t_s, t_e \)

Prover \((x, w)\)

\[ \pi \leftarrow \text{Prove}(crs, x, w) \]

Verifier \(x\)

\[ \text{Verify}(crs, x, \pi) = 1 \Rightarrow x \in L \]

Extractor \(t_e\)

Simulator \(t_s\)
NIZKs in the CRS Model

\[
\begin{align*}
\pi & \leftarrow \text{Prove}(\text{crs}, x, w) \\
\text{Extractor } t_e & \leftarrow \text{Ext}(\text{crs}, t_e, \pi) \quad \pi & \leftarrow \text{Sim}(\text{crs}, t_s, x) \\
\text{Veri/f_\_er} & x \quad \text{Verify}(\text{crs}, x, \pi) = 1 \Rightarrow x \in L
\end{align*}
\]
Achieving Simulation Extractability
Extend statement to

\[ c = \Omega.\text{Enc}(pk_\Omega, w; r_0) \land ((x, w) \in R_L \lor (\mu = f_s(pk_{\Sigma^1}) \land \rho = \text{Commit}(s; r_1))) \]

and sign \((x, c, \mu, \pi_{L'})\) with \(sk_{\Sigma^1}\)

\(\text{crs}\) extended with \(\rho, pk_\Omega; s, r_0\) simulation trapdoor, \(sk_\Omega\) extraction trapdoor

- \(\Omega\): public-key encryption
- \(\Sigma^1\): strong one-time signature
- \(f\): PRF
- \textbf{Commit}: Commitment
Extend statement to

\[ c = \Omega \cdot \text{Enc}(pk_\Omega, w; r_0) \land ((x, w) \in R_L \lor (\mu = f_s(pk_{\Sigma^1}) \land \rho = \text{Commit}(s; r_1))) \]

and sign \((x, c, \mu, \pi_{L'})\) with \(sk_{\Sigma^1}\)

crs extended with \(\rho, pk_\Omega, s, r_0\) simulation trapdoor, \(sk_\Omega\) extraction trapdoor

- \(\Omega\): public-key encryption
- \(\Sigma^1\): strong one-time signature
- \(f\): PRF
- **Commit**: Commitment using SHA256
  Proving pre-image of a random oracle
The OCØCØ Framework [ARS20]

Fixed-value key-binding PRF [CMR98; Fis99]

- For a PRF $f$ with key $s$ and special value $\beta$, hard to find $s'$ with $f_s(\beta) = f_{s'}(\beta)$
Fixed-value key-binding PRF [CMR98; Fis99]

- For a PRF $f$ with key $s$ and special value $\beta$, hard to find $s'$ with $f_s(\beta) = f_{s'}(\beta)$

Change statement to

$$(x, w) \in R_L \lor (\mu = f_s(pk_{\Sigma_1}) \land \rho = f_s(\beta))$$

Allows instantiation with low complexity primitives
Subversion and Updatability
CRS Generator

\[ \pi \leftarrow \text{Prove}(\text{crs}, x, w) \]

\[ \text{Verify}(\text{crs}, x, \pi) = 1 \Rightarrow x \in L \]
CRS Generator

TTP

Prover \((x, w)\)

\[ \pi \leftarrow \text{Prove}(\text{crs}, x, w) \]

Verifier \(x\)

\[ \text{Verify}(\text{crs}, x, \pi) = 1 \Rightarrow x \in L \]
What if the CRS generator is malicious?

No guarantee that

- CRS is correct
- CRS from the correct distribution
- Trapdoors exist
What if the CRS generator is malicious?

No guarantee that

- CRS is correct
- CRS from the correct distribution
- Trapdoors exist

Perform CRS generation with MPC protocol

- Examples: zcash ceremony
- But in practice complicated, expensive and requires much effort beside technical realization
Subversion Resistance [BFS16]

- Subversion soundness: sound even if CRS subverted
- Subversion zero-knowledge: zero-knowledge even if CRS subverted
- Some combinations impossible

<table>
<thead>
<tr>
<th></th>
<th>WI</th>
<th>Zero-Knowledge</th>
<th>Subversion ZK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soundness</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Subversion soundness</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Updatable NIZK [GKM+18]

- Assume adversary has complete (or partial) control over \( \text{crs} \) generation
- Add \( \text{Ucrs} \) algorithm: outputs a new CRS and proof of update
- Also add \( \text{Vcrs} \): verifies CRS, updates and proofs

Idea: either \( \text{crs} \) was generated honestly or one update was done honestly

- Verifier updates CRS to ensure soundness
- Prover updates CRS to ensure zero-knowledge
Towards LAMASSU: Key-homomorphic Signatures / Updatable Signatures

Key-homomorphic signatures:

- Homomorphisms between private-key and public-key spaces: \( \mu : S \rightarrow P \)
  - Natural in the DLOG setting: \( x \mapsto g^x \)
  - Signatures can be adapted from \( pk \) to \( pk' = pk \cdot \mu(sk' - sk) \) if \( sk' - sk \) known
  - Examples: Schnorr, BLS, and many more

Updatable signatures:

- **Upk**: update \( pk \) and provide proof of update
- **Vpk**: verify update
- Idea: either original \( pk \) created honestly or update was done honestly
- Example: Schnorr in bilinear groups with BDH knowledge assumption
Compiler [DS19]: “\( x \in L \) or I can sign with a public key in the CRS”

- Extend statement to

\[
(x, w) \in R_L \vee pk' = pk \cdot \mu(sk' - sk)
\]

- Generate key pairs \((sk', pk')\) for \(\Sigma\) and \((sk^1, pk^1)\) for \(\Sigma^1\)
- Sign \(pk^1\) with \(sk'\) and sign the proof with \(sk^1\)

- \(\Sigma\): key-homomorphic EUF-CMA signature scheme
- \(\Sigma^1\): one-time signature scheme
- Extend CRS with a public key of \(\Sigma\): \(pk\)
- Put secret key \(sk\) of \(\Sigma\) in simulation trapdoor
Generic framework to obtain

• subversion or updatable
• and simulation extractable zk-SNARKs

Built from

• updatable signatures
• DS compiler for simulation soundess [DS19]
Conclusion
C∅C∅, OC∅C∅:

• C∅C∅ hard to instantiate correctly and efficiently
• Even if commitment with enough structure used, C∅C∅ does not seem to yield updatability
• sub-ZK SE SNARK if underlying SNARK already sub-ZK

LAMASSU:

• generic sub-ZK, updatable SE SNARK
• Open problems: key-homomorphic / updatable signatures from lattices, ...
Questions?

References


