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

Simulation extractable, subversion, and updatable NIZKs

Sebastian Ramacher, joint work with Behzad Abdolmaleki and Daniel Slamanig

March 4, 2020

AIT Austrian Institute of Technology, Vienna

Introduction

Zero-knowledge Proofs

NP-language L

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

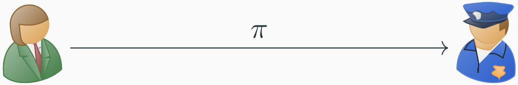
Prover $(\mathtt{x}, \mathtt{w})$

Verifier x

$\mathtt{x} \in L$ ✓

Making them Non-Interactive: ROM

Random-oracle model: Fiat-Shamir transform [FS86], Unruh transform [Unr15]

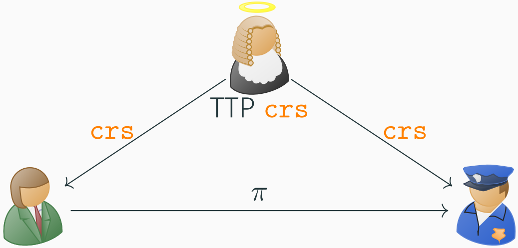


Prover $(\mathtt{x}, \mathtt{w})$ Verifier $\mathtt{x}$

$\pi \leftarrow \mathsf{Prove}_H(\mathtt{x}, \mathtt{w})$ $\mathsf{Verify}_H(\mathtt{x}, \pi) = 1 \Rightarrow \mathtt{x} \in L$

Making them Non-Interactive: CRS

Common reference string model



Prover $(\mathtt{x}, \mathtt{w})$ Verifier $\mathtt{x}$

$\pi \leftarrow \mathsf{Prove}(\mathtt{crs}, \mathtt{x}, \mathtt{w})$ $\mathsf{Verify}(\mathtt{crs}, \mathtt{x}, \pi) = 1 \Rightarrow \mathtt{x} \in L$

Important Properties

Prover cannot cheat

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

> Soundness

- Property desired by the verifier

Important Properties

Prover cannot cheat

- Prover unable to produce valid proofs for $\mathtt{x} \notin L$
- > Soundness
- Property desired by the verifier

Verifier does not learn any information on witness $\mathtt{w}$

- 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
- SKS also called simulation (sound) extractability (SE)

NIZKs in the CRS Model

- Zero-knowledge contradicts extractor
- Soundness contradicts simulator

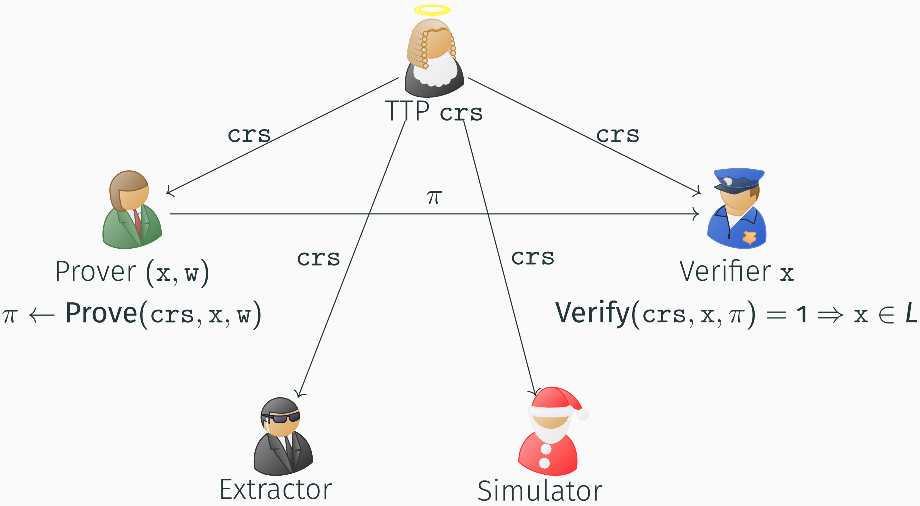
NIZKs in the CRS Model

- Zero-knowledge contradicts extractor
- Soundness contradicts simulator

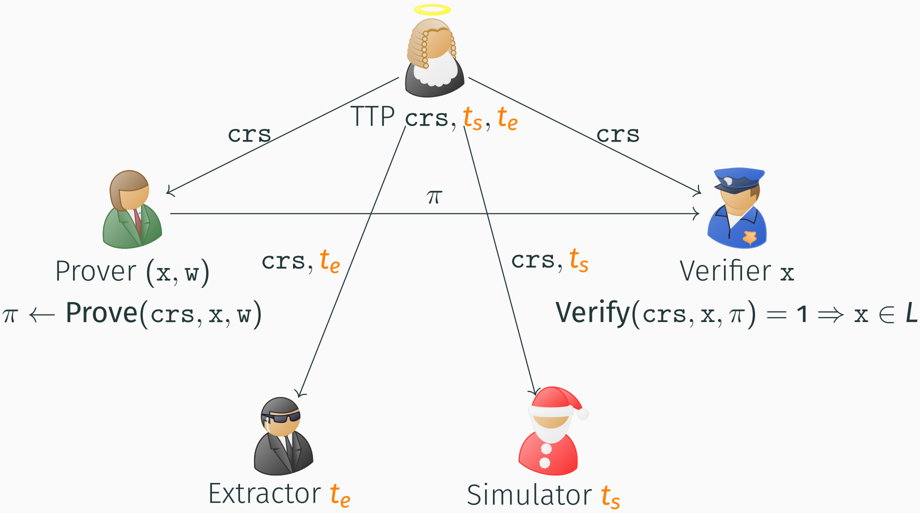
They need to have more power

- Extractor gets extraction trapdoor
- Simulator gets simulation trapdoor

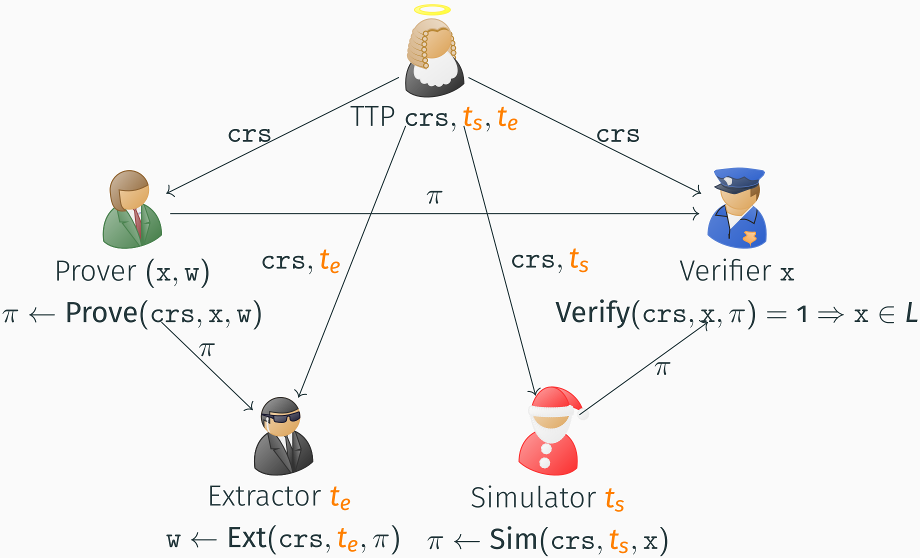
NIZKs in the CRS Model



NIZKs in the CRS Model



NIZKs in the CRS Model



Achieving Simulation Extractability

Soundness to Knowledge Soundness

Folklore compiler [SP92]: “$\mathtt{x} \in \boldsymbol{L}$ and I have encrypted $\mathtt{w}$”

- Ω: perfectly correct IND-CPA public-key encryption
- Extend CRS with a public key of Ω: pk_Ω

Soundness to Knowledge Soundness

Folklore compiler [SP92]: "$\mathtt{x} \in L$ and I have encrypted $\mathtt{w}$"

- Ω: perfectly correct IND-CPA public-key encryption
- Extend CRS with a public key of Ω: pk_Ω
- Extend proof with encryption of $\mathtt{w}$: $\mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \mathtt{w}; r)$

Soundness to Knowledge Soundness

Folklore compiler [SP92]: "$\mathtt{x} \in L$ and I have encrypted $\mathtt{w}$"

- Ω: perfectly correct IND-CPA public-key encryption
- Extend CRS with a public key of Ω: pk_Ω
- Extend proof with encryption of $\mathtt{w}$: $\mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \mathtt{w}; r)$
- Extend statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \wedge \mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \underline{\mathtt{w}}; \underline{r})$$

Soundness to Knowledge Soundness

Folklore compiler [SP92]: "$\mathtt{x} \in L$ and I have encrypted $\mathtt{w}$"

- Ω: perfectly correct IND-CPA public-key encryption
- Extend CRS with a public key of Ω: pk_Ω
- Extend proof with encryption of $\mathtt{w}$: $\mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \mathtt{w}; r)$
- Extend statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \wedge \mathsf{c} = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \underline{\mathtt{w}}; \underline{r})$$

- Put secret key in extraction trapdoor t_e

On Simulation Soundness

In a real world protocol:

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

On Simulation Soundness

In a real world protocol:

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

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

Soundness to Simulation Soundness

Folklore compiler [GMY03; Gro06]: "$\mathtt{x} \in L$ or I know a signature under a public key in the CRS"

- Σ: EUF-CMA signature scheme
- Σ^1: strong one-time signature scheme
- Extend CRS with a public key of Σ: pk_Σ

Soundness to Simulation Soundness

Folklore compiler [GMY03; Gro06]: “$\mathtt{x} \in L$ or I know a signature under a public key in the CRS”

- Σ: EUF-CMA signature scheme
- Σ^1: strong one-time signature scheme
- Extend CRS with a public key of Σ: pk_Σ
- For a proof
    - Generate new key pair for Σ^1

Soundness to Simulation Soundness

Folklore compiler [GMY03; Gro06]: "$\mathtt{x} \in L$ or I know a signature under a public key in the CRS"

- Σ: EUF-CMA signature scheme
- Σ^1: strong one-time signature scheme
- Extend CRS with a public key of Σ: pk_Σ
- For a proof
  - Generate new key pair for Σ^1
  - Extend statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \vee \Sigma.\mathsf{Verify}(\mathsf{pk}_\Sigma, \mathsf{pk}_{\Sigma^1}, \underline{\sigma}) = 1$$

Soundness to Simulation Soundness

Folklore compiler [GMY03; Gro06]: "$\mathtt{x} \in L$ or I know a signature under a public key in the CRS"

- Σ: EUF-CMA signature scheme
- Σ^1: strong one-time signature scheme
- Extend CRS with a public key of Σ: pk_Σ
- For a proof
  - Generate new key pair for Σ^1
  - Extend statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \vee \Sigma.\mathsf{Verify}(\mathsf{pk}_\Sigma, \mathsf{pk}_{\Sigma^1}, \underline{\sigma}) = 1$$

  - Sign proof with sk_{Σ^1}

Soundness to Simulation Soundness

Folklore compiler [GMY03; Gro06]: "$\mathtt{x} \in L$ or I know a signature under a public key in the CRS"

- Σ: EUF-CMA signature scheme
- Σ^1: strong one-time signature scheme
- Extend CRS with a public key of Σ: pk_Σ
- For a proof
  - Generate new key pair for Σ^1
  - Extend statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \vee \Sigma.\mathsf{Verify}(\mathsf{pk}_\Sigma, \mathsf{pk}_{\Sigma^1}, \underline{\sigma}) = 1$$

  - Sign proof with sk_{Σ^1}
- Put secret key of Σ in simulation trapdoor t_s

The C∅C∅ Framework [KZM+15]

Extend statement to

$$c = \Omega.\mathsf{Enc}(\mathsf{pk}_\Omega, \underline{\mathtt{w}}; \underline{r_1}) \wedge \big((\mathtt{x}, \underline{\mathtt{w}}) \in R_L \vee \big(\mu = f_{\underline{s}}(\mathsf{pk}_{\Sigma^1}) \wedge \rho = \mathsf{Commit}(\underline{s}; \underline{r_0})\big)\big)$$

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

$\mathtt{crs}$ extended with ρ, pk_Ω; s, r_0 simulation trapdorr, sk_Ω extraction trapdoor

- Ω: public-key encryption
- Σ^1: strong one-time signature
- f: PRF
- Commit: Commitment

Instantation of C∅C∅

- Ω: RSA
- Σ^1: RSA-PSS

Instantation of C∅C∅

- Ω: RSA
- Σ^1: RSA-PSS
  Not optimal; better alternatives: Boneh-Boyen [BB04], Groth sOTS

Instantation of C∅C∅

- Ω: RSA
- Σ^1: RSA-PSS
  Not optimal; better alternatives: Boneh-Boyen [BB04], Groth sOTS
- f: SHA256
- **Commit**: SHA256

Instantation of C∅C∅

- Ω: RSA
- Σ^1: RSA-PSS
  Not optimal; better alternatives: Boneh-Boyen [BB04], Groth sOTS
- f: SHA256
- **Commit**: SHA256
  Proving pre-image of a random oracle

Instantation of C∅C∅

- Ω: RSA
- Σ^1: RSA-PSS
  Not optimal; better alternatives: Boneh-Boyen [BB04], Groth sOTS
- f: SHA256
- **Commit**: SHA256
  Proving pre-image of a random oracle

How to commit to the PRF key while retaining provable security?

The OC∅C∅ Framework [ARS20]

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

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

The OC∅C∅ Framework [ARS20]

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

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

Change statement to

$$(\mathtt{x}, \underline{\mathtt{w}}) \in R_L \vee \left(\mu = f_{\underline{s}}(\mathsf{pk}_{\Sigma 1}) \wedge \rho = f_{\underline{s}}(\beta)\right)$$

Instantiation of OC∅C∅

Framework	Symmetric primitive	PRF / Commitment	# of constraints
C∅C∅	SHA256	HMAC PRF + hash com.	244,992
OC∅C∅	SHA256	HMAC PRF	222,720
	SHAKE256	Sponge PRF	76,800
	MiMC-$(1025, 646)$	Sponge PRF	1,292
	GMiMC-$(1024, 4, 332)$	Sponge PRF	1,998
	Poseidon-$(1536, 2, 10, 114)$	Sponge PRF	804
	Vision-$(1778, 14, 10)$	Sponge PRF	2,800
	Rescue-$(1750, 14, 10)$	Sponge PRF	1,680
	LowMC-$(1024, 256, 1, 1027)$	Sponge PRF	4,288

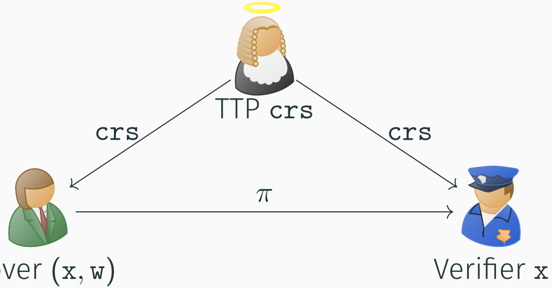
Instantiation of OC∅C∅

Beware:

- Numbers from before recent attacks appeared on ePrint
- Numbers are lower bounds assuming PRFs are fixed-value key-binding
- Alternatively: More expensive tree-based construction

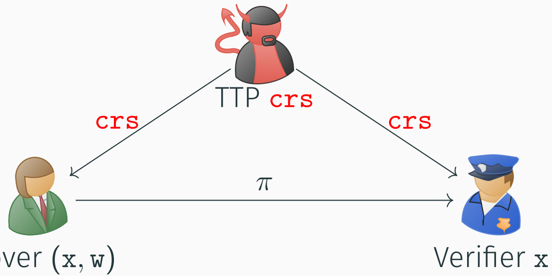
Subversion and Updatability

CRS Generator



$\pi \leftarrow \textsf{Prove}(\mathtt{crs}, \mathtt{x}, \mathtt{w})$ $\qquad$ $\textsf{Verify}(\mathtt{crs}, \mathtt{x}, \pi) = 1 \Rightarrow \mathtt{x} \in L$

CRS Generator



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

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

What if the CRS generator is malicious?

Malicious CRS Generator

No guarantee that

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

Malicious CRS Generator

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 techical realization

Subversion Resistance [BFS16]

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

Subversion Resistance [BFS16]

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

	WI	Zero-Knowledge	Subversion ZK
Soundness	✓	✓	✓
Subversion soundness	✓	✗	✗

Constructions

Only ad-hoc constructions so far [ABL+17; Fuc18; Bag19]

General idea:

- Add public algorithm Vcrs
- If $\mathsf{Vcrs}(\mathtt{crs}) = 1$, CRS is valid and simulation trapdoor exists
- In sub-ZK proof, extract trapdoor from CRS for simulation

Updatable NIZK [GKM+18]

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

Idea: either $\mathtt{crs}$ was generated honestly or one update was done honestly

- Verifier updates CRS to ensure soundness
- Prover updates CRS to ensure zero-knowledge

Lamassu [ARS20]

Generic framework to obtain

- subversion or updatable
- and simulation extractable zk-SNARKs

Built from

- updatable signatures
- alternative compiler for simulation soundess [DS19]

Key-homomorphic Signatures

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

Simulation Soundness using Key-Homomorphic Signatures

Compiler [DS19]:

- Σ: key-homormorphic EUF-CMA signature scheme
- Σ^1: one-time signature scheme
- Extend CRS with a public key of Σ: pk
- For a proof
  - Generate key pairs $(\mathsf{sk}', \mathsf{pk}')$ for Σ and $(\mathsf{sk}^1, \mathsf{pk}^1)$ for Σ^1
  - Extend statement to

$$(\mathtt{x}, \mathtt{w}) \in R_L \vee \mathsf{pk}' = \mathsf{pk} \cdot \mu(\mathsf{sk}' - \underline{\mathsf{sk}})$$

  - Sign pk^1 with sk' and sign the proof with sk^1
- Put secret key of Σ in simulation trapdoor t_s

Obtain simulation extractable, subversion zk-SNARK

Updatable Signatures

Similar to updatable CRS

- **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

Obtain simulation extractable, updatable zk-SNARK

Comparison of SE SNARKs

	Gen.	Sub.	Upd.	crs	π
CØCØ [KZM+15]	✓	✓	✗	1κ	$2\mathbb{Z}_N, 1\kappa$
OCØCØ[G]	✓	✓	✗	2κ	$3\mathbb{G}, 3\mathbb{Z}_q, 1\kappa$
LAMASSU[S,G]	✓	✓	✗	$1\mathbb{G}$	$4\mathbb{G}, 5\mathbb{Z}_q$
LAMASSU[S,G]	✓	✓	✓	$1\mathbb{G}_1, 1\mathbb{G}_2$	$1\mathbb{G}_1, 1\mathbb{G}_2, 3\mathbb{G}, 5\mathbb{Z}_q$
LAMASSU[S,BB]	✓	✓	✗	$1\mathbb{G}$	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\mathbb{G}, 2\mathbb{Z}_q$
LAMASSU[S,BB]	✓	✓	✓	$1\mathbb{G}_1, 1\mathbb{G}_2$	$2\mathbb{G}_1, 2\mathbb{G}_2, 2\mathbb{Z}_q$
Groth-Maller [GM17]	✗	✗	✗	$(2n+5)\mathbb{G}_1, n\mathbb{G}_2$	–
Bowe-Gabizon [BG18]	✗	✗	✗	–	$1\mathbb{G}_1, 1\mathbb{G}_2$
Lipmaa (S^{se}_{qap}) [Lip19]	✗	✓	✗	$n\mathbb{G}_1$	$1\mathbb{G}_1$
Kim-Lee-Oh [KLO19]	✗	✓	✓	$n\mathbb{G}_1$	–
Atapoor-Baghery [AB19]	✗	✗	✗	1κ	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\kappa$
Baghery [Bag19]	✗	✓	✗	1κ	$1\mathbb{G}_1, 1\mathbb{G}_2, 1\kappa$

Conclusion

Conclusion

CØCØ, OCØCØ:

- CØCØ hard to instantiate correctly and efficiently
- Even if commitment with enough structure used, CØCØ does not seem to yield updatability
- sub-ZK SE SNARK if underlying SNARK already sub-ZK
- OCØCØ gives another application of fixed-value key-binding PRFs

LAMASSU:

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

Questions?

Preprint: `https://eprint.iacr.org/2020/062.pdf`

AUSTRIAN INSTITUTE OF TECHNOLOGY

TARTU ÜLIKOOL · UNIVERSITAS TARTUENSIS · 1632

References

References i

[AB19] S. Atapoor and K. Baghery. Simulation extractability in groth's zk-snark. In *DPM/CBT@ESORICS*, volume 11737 of *LNCS*, pages 336–354. Springer, 2019.

[ABL+17] B. Abdolmaleki, K. Baghery, H. Lipmaa, and M. Zajac. A subversion-resistant SNARK. In *ASIACRYPT (3)*, volume 10626 of *LNCS*, pages 3–33. Springer, 2017.

[ARS20] B. Abdolmaleki, S. Ramacher, and D. Slamanig. Lift-and-shift: obtaining simulation extractable subversion and updatable snarks generically. Cryptology ePrint Archive, Report 2020/062, 2020. `https://eprint.iacr.org/2020/062`.

References ii

[Bag19] K. Baghery. Subversion-resistant simulation (knowledge) sound nizks. In *IMACC*, volume 11929 of *LNCS*, pages 42–63. Springer, 2019.

[BB04] D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BFS16] M. Bellare, G. Fuchsbauer, and A. Scafuro. Nizks with an untrusted CRS: security in the face of parameter subversion. In *ASIACRYPT (2)*, volume 10032 of *LNCS*, pages 777–804, 2016.

[BG18] S. Bowe and A. Gabizon. Making groth's zk-snark simulation extractable in the random oracle model. Cryptology ePrint Archive, Report 2018/187, 2018. `https://eprint.iacr.org/2018/187`.

References iii

[CMR98] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *STOC*, pages 131–140. ACM, 1998.

[DS19] D. Derler and D. Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Des. Codes Cryptogr.*, 87(6):1373–1413, 2019.

[Fis99] M. Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: improvements and applications. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 432–445. Springer, 1999.

[FS86] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[Fuc18] G. Fuchsbauer. Subversion-zero-knowledge snarks. In *PKC (1)*, volume 10769 of *LNCS*, pages 315–347. Springer, 2018.

[GKM+18] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. In *CRYPTO (3)*, volume 10993 of *LNCS*, pages 698–728. Springer, 2018.

[GM17] J. Groth and M. Maller. Snarky signatures: minimal signatures of knowledge from simulation-extractable snarks. In *CRYPTO (2)*, volume 10402 of *LNCS*, pages 581–612. Springer, 2017.

[GMY03] J. A. Garay, P. D. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In *EUROCRYPT*, volume 2656 of *LNCS*, pages 177–194. Springer, 2003.

[Gro06] J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, volume 4284 of *LNCS*, pages 444–459. Springer, 2006.

[KLO19] J. Kim, J. Lee, and H. Oh. Updatable crs simulation-extractable zk-snarks with a single verification. Cryptology ePrint Archive, Report 2019/586, 2019. `https://eprint.iacr.org/2019/586`.

[KZM+15] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, abhi shelat, and E. Shi. Coco: a framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.

[Lip19] H. Lipmaa. Simulation-extractable snarks revisited. Cryptology ePrint Archive, Report 2019/612, 2019. `https://eprint.iacr.org/2019/612`.

[SP92] A. D. Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *FOCS*, pages 427–436. IEEE Computer Society, 1992.

[Unr15] D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT (2)*, volume 9057 of *LNCS*, pages 755–784. Springer, 2015.