Malleable SNARKs and Their Applications

Suvradip Chakraborty[1] Dennis Hofheinz[2] Roman Langrehr[2] Jesper Buus Nielsen[3]
Christoph Striecks[4] Daniele Venturi[5]

[1]Visa Research

[2]ETH Zurich

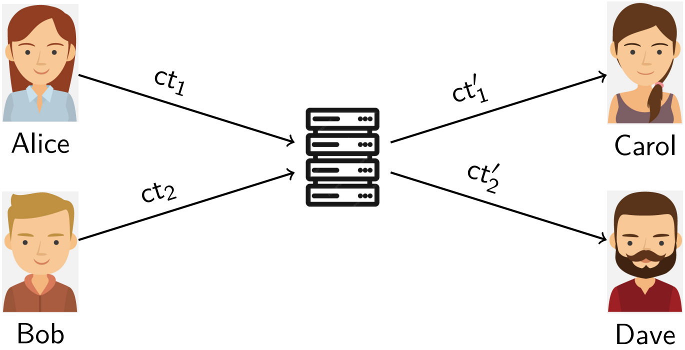
[3]Aarhus University

[4]AIT Austrian Institute of Technology

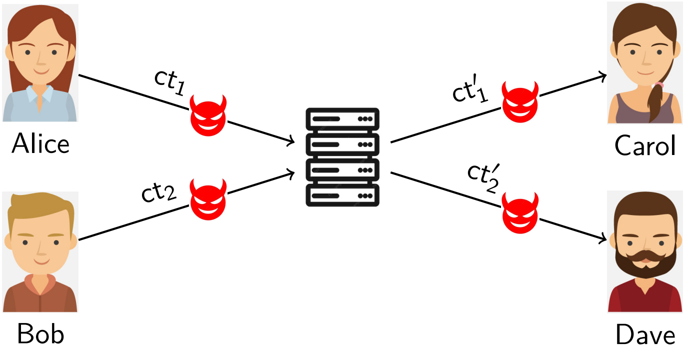
[5]Sapienza University of Rome

2025-05-08

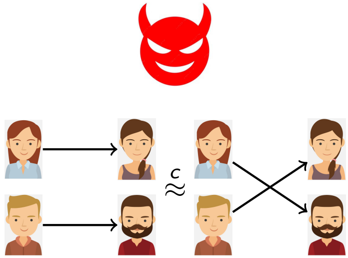
Motivation: Mix-Nets (over-simplified)



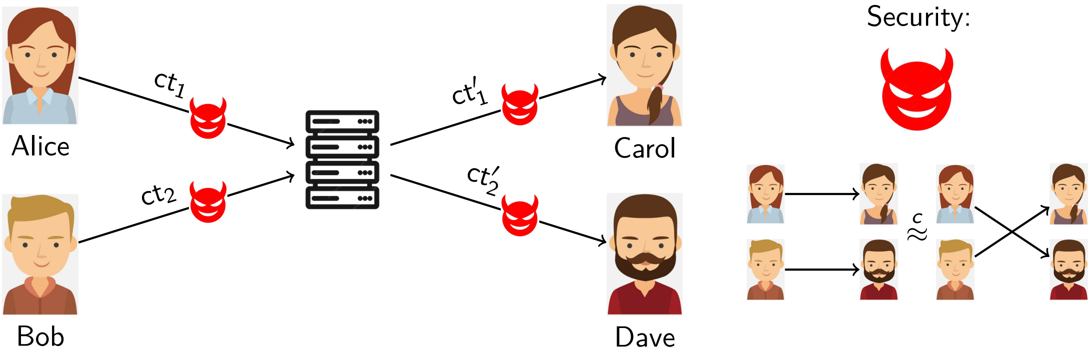
Motivation: Mix-Nets (over-simplified)



Security:



Motivation: Mix-Nets (over-simplified)



Solution:  rerandomizes ciphertexts (+ anonymity of the encryption scheme)

IND-RCCA secure rerandomizable PKE

Given: IND-CPA secure rerandomizable PKE
Needed: IND-RCCA secure rerandomizable PKE

IND-RCCA secure rerandomizable PKE

Given: IND-CPA secure rerandomizable PKE

Needed: IND-RCCA secure rerandomizable PKE

Dec oracle returns $\diamond$ if ciphertext decrypts to $m_0^\star$ or $m_1^\star$

IND-RCCA secure rerandomizable PKE

Given: IND-CPA secure rerandomizable PKE
Needed: IND-RCCA secure rerandomizable PKE

IND-CPA PKE

Naor-Yung transform [NY90]
$\mathsf{ct}' = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

ct_1 and ct_2 encrypt the same message

$\downarrow$

IND-CCA PKE

IND-RCCA secure rerandomizable PKE

Given: IND-CPA secure rerandomizable PKE
Needed: IND-RCCA secure rerandomizable PKE

IND-CPA PKE

Naor-Yung transform [NY90]
$\mathsf{ct}' = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

ct_1 and ct_2 encrypt the same message

IND-CCA PKE

IND-CPA rerandomizable PKE

Naor-Yung transform
$\mathsf{ct}' = (\mathsf{ct}_1, \mathsf{ct}_2, \pi)$

ct_1 and ct_2 encrypt the same message

IND-RCCA rerandomizable PKE

IND-RCCA secure rerandomizable PKE

Given: IND-CPA secure rerandomizable PKE
Needed: IND-RCCA secure rerandomizable PKE

IND-CPA PKE

Naor-Yung transform [NY90]
$\mathrm{ct}' = (\mathrm{ct}_1, \mathrm{ct}_2, \pi)$

ct_1 and ct_2 encrypt the same message

IND-CCA PKE

IND-CPA rerandomizable PKE

c

Proof system must support:

- rerandomization
- adapt the statement to rerandomized ciphertexts

sage

IND-RCCA rerandomizable PKE

Malleable NIZKs [CKLM12]

- NP relation $\mathcal{R}$
- Allowed transformations $\mathcal{T}$

such that

$$\forall (x, w) \in \mathcal{R}, (T_x, T_w) \in \mathcal{T} \rightarrow (T_x(x), T_w(w)) \in \mathcal{R}$$

Malleable NIZKs [CKLM12]

- NP relation $\mathcal{R}$
- Allowed transformations $\mathcal{T}$

such that

$$\forall (x, w) \in \mathcal{R}, (T_x, T_w) \in \mathcal{T} \rightarrow (T_x(x), T_w(w)) \in \mathcal{R}$$

A malleable NIZK for $\mathcal{R}$ and $\mathcal{T}$ is

- a NIZK for $\mathcal{R}$
- with the following extra feature

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

$\boxed{x \text{ is true}}$ $\boxed{T_x(x) \text{ is true}}$

Malleable NIZKs [CKLM12]

- NP relation $\mathcal{R}$
- Allowed transformations $\mathcal{T}$

such that

$$\forall (x, w) \in \mathcal{R}, (T_x, T_w) \in \mathcal{T} \rightarrow (T_x(x), T_w(w)) \in \mathcal{R}$$

A malleable NIZK for $\mathcal{R}$ and $\mathcal{T}$ is

- a NIZK for $\mathcal{R}$
- with the following extra feature

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

$\boxed{x \text{ is true}}$ $\boxed{T_x(x) \text{ is true}}$

Derivation-privacy: (π, π') indistinguishable from directly generated proofs for x and $T_x(x)$

Malleable NIZKs [CKLM12]

- NP relation $\mathcal{R}$
- Allowed transformations $\mathcal{T}$

such that

$$\forall (x, w) \in \mathcal{R}, (T_x, T_w) \in \mathcal{T} \to (T_x(x), T_w(w)) \in \mathcal{R}$$

A malleable NIZK for $\mathcal{R}$ and $\mathcal{T}$ is

- a NIZK for $\mathcal{R}$
- with the following extra feature

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

$\boxed{x \text{ is true}}$ $\boxed{T_x(x) \text{ is true}}$

Derivation-privacy: (π, π') indistinguishable from directly generated proofs for x and $T_x(x)$
Simulation-soundness: Takes into account that simulated statements can be modified with $\mathcal{T}$

Instantiations of malleable NIZKs

Groth-Sahai proofs [CKLM12]:

- ✓ Very efficient
- ✗ Only for a specific $\mathcal{R}$ (pairing-product equations)
- ✗ Only for a specific $\mathcal{T}$
- ✗ Not post-quantum secure

Instantiations of malleable NIZKs

Groth-Sahai proofs [CKLM12]:

- ✓ Very efficient
- ✗ Only for a specific $\mathcal{R}$ (pairing-product equations)
- ✗ Only for a specific $\mathcal{T}$
- ✗ Not post-quantum secure

Generically from SNARKs [CKLM13]:

- ✓ Any NP-relation $\mathcal{R}$
- ✓ Any set of valid transformations $\mathcal{T}$
- ✓ Post-quantum instantiation possible
- ✗ Number of transformations is bounded
- ✗ Not derivation private

Instantiations of malleable NIZKs

Groth-Sahai proofs [CKLM12]:

- ✓ Very efficient
- ✗ Only for a specific $\mathcal{R}$ (pairing-product equations)
- ✗ Only for a specific $\mathcal{T}$
- ✗ Not post-quantum secure

Generically from SNARKs [CKLM13]:

- ✓ Any NP-relation $\mathcal{R}$
- ✓ Any set of valid transformations $\mathcal{T}$
- ✓ Post-quantum instantiation possible
- ✗ Number of transformations is bounded
- ✗ Not derivation private

Generically from SNARKs [our work]:

- ✓ Any NP-relation $\mathcal{R}$
- ✓ Any set of valid transformations $\mathcal{T}$
- ✓ Post-quantum instantiation possible
- ✓ Unbounded transformations
- ✓ Derivation private

Malleable NIZKs from SNARKs

x is true

$x' = T_x(x)$ is true

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

Malleable NIZKs from SNARKs

x is true

$x' = T_x(x)$ is true

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

$\exists w : (x, w) \in \mathcal{R}$

Malleable NIZKs from SNARKs

x is true

$x' = T_x(x)$ is true

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

$\exists w : (x, w) \in \mathcal{R}$

$\exists (T_x, T_w) \in \mathcal{T}, x, \pi :$
$x' = T_x(x) \wedge \pi$ proves x

Malleable NIZKs from SNARKs

$$\boxed{x \text{ is true}} \qquad \boxed{x' = T_x(x) \text{ is true}}$$

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

NP-Relation: $\exists w : (x, w) \in \mathcal{R}$ or $\exists (T_x, T_w) \in \mathcal{T}, x, \pi : x' = T_x(x) \wedge \pi$ proves x

Malleable NIZKs from SNARKs

x is true

$x' = T_x(x)$ is true

$$\pi \xrightarrow{(T_x, T_w)} \pi'$$

NP-Relation: $\exists w : (x, w) \in \mathcal{R}$ or $\exists (T_x, T_w) \in \mathcal{T}, x, \pi :$ $x' = T_x(x) \wedge \pi$ proves x

Recursive usage of SNARKs

- succinctness avoids blow-up of proof size/verification time
- ✓zero-knowledge ✓derivation private (by zero-knowledge of the SNARK)
- also used for incrementally verifiable computation (IVC), proof carrying data (PCD), blockchains (to compress proofs)

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$

$\Downarrow$

Recursive extraction

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$

$\Downarrow$

Recursive extraction

Issue 1: Runtime of the extractor can explode

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$ $\Downarrow$ Recursive extraction

Issue 1: Runtime of the extractor can explode

Solution: We assume a fast extractor:

$$\text{Time}_{\mathcal{E}} \leq \text{Time}_{\mathcal{A}} + \text{poly}(\lambda)$$

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$ $\Downarrow$ Recursive extraction

Issue 1: Runtime of the extractor can explode

Issue 2: Recursion might never stop

Solution: We assume a fast extractor:

$$\text{Time}_{\mathcal{E}} \leq \text{Time}_{\mathcal{A}} + \text{poly}(\lambda)$$

Soundness

Soundness (SNARK variant): $\forall \mathcal{A}$ outputting (x, π) s.t π proves x
$\exists \mathcal{E}$ outputting w s.t. $(x, w) \in \mathcal{R}$.

In our case: $\mathcal{E}$ can output

- w s.t. $(x, w) \in \mathcal{R}$
- $(T_x, T_w) \in \mathcal{T}, \hat{x}, \hat{\pi}$ s.t. $x = T_x(\hat{x})$ and $\underline{\hat{\pi} \text{ proves } \hat{x}}$ $\Downarrow$ Recursive extraction

Issue 1: Runtime of the extractor can explode

Issue 2: Recursion might never stop

Solution: We assume a fast extractor:

$$\text{Time}_{\mathcal{E}} \leq \text{Time}_{\mathcal{A}} + \text{poly}(\lambda)$$

$\boxed{\text{Main technical challenge}}$

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs

Warm-up solution: Counters [CKLM13]

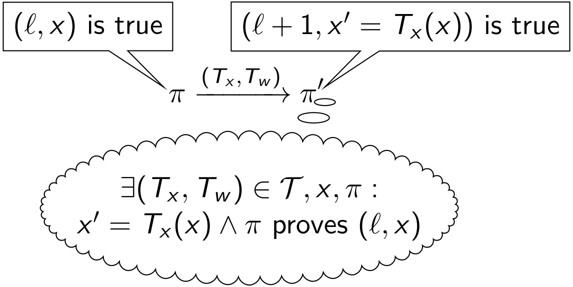
- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion
  - Recursion is only allowed for $\ell < B$

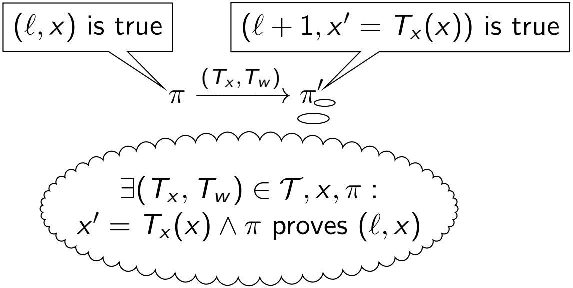
Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion
  - Recursion is only allowed for $\ell < B$



Warm-up solution: Counters [CKLM13]

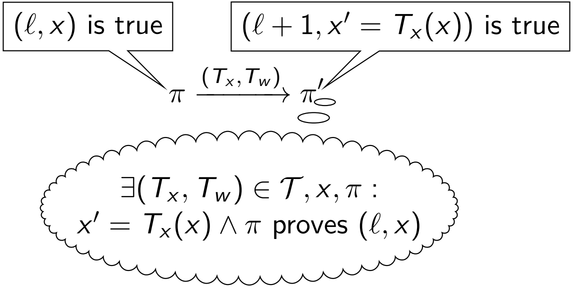
- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion
  - Recursion is only allowed for $\ell < B$



✓ Soundness

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion
  - Recursion is only allowed for $\ell < B$

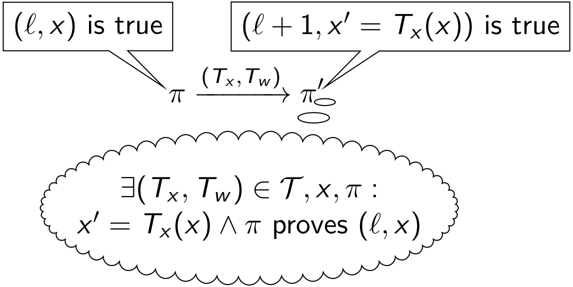


✓ Soundness

✗ Limited derivation privacy (counters leak # recursions)

Warm-up solution: Counters [CKLM13]

- Fixed bound B on recursion depth
- Add a counter ℓ to SNARK statements
  - $\ell = 1$ for non-recursive proofs
  - ℓ is incremented by 1 in each recursion
  - Recursion is only allowed for $\ell < B$



✓ Soundness

✗ Limited derivation privacy (counters leak # recursions)

✗ Only bounded number of recursions

Non-solution: One-way permutation

OWP $f : X \to X$

Counter ℓ is replaced by $\xi \in X$

- Initially: $\xi \leftarrow X$
- Recursion $\xi' = f(\xi)$

Idea: Extractor would have to break one-wayness if

$$\#\text{ recursion for extraction} > \#\text{ recursion to generate the proof (honestly)}$$

Non-solution: One-way permutation

OWP $f : X \to X$
Counter ℓ is replaced by $\xi \in X$

- Initially: $\xi \leftarrow X$
- Recursion $\xi' = f(\xi)$

Idea: Extractor would have to break one-wayness if

$$\# \text{ recursion for extraction} > \# \text{ recursion to generate the proof (honestly)}$$

Issue: Soundness has to hold for adversarially generated proofs
E.g. adversary can choose initial $\xi \in X$ from a different distribution.

Non-solution: One-way permutation

OWP $f : X \to X$

Counter ℓ is replaced by $\xi \in X$

- Initially: $\xi \leftarrow X$
- Recursion $\xi' = f(\xi)$

Idea: Extractor would have to break one-wayness if

$$\#\text{ recursion for extraction} > \#\text{ recursion to generate the proof (honestly)}$$

Issue: Soundness has to hold for <u>adversarially</u> generated proofs

E.g. adversary can choose initial $\xi \in X$ from a different distribution.

We need a variant of OWF that is secure if the adversary chooses the value x to invert.

Adversarial one-way functions (AOWFs)

Stateful 2-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$

1^{st} attempt: $\begin{array}{l} \mathcal{A}_1 \text{ outputs } y \\ \mathcal{A}_2 \text{ outputs } x \end{array}$ $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f(x) = y$

Adversarial one-way functions (AOWFs)

Stateful 2-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$

1st attempt: $\mathcal{A}_1$ outputs y, $\mathcal{A}_2$ outputs x — $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f(x) = y$

Unsatisfiable! $\mathcal{A}_1$ samples $x \leftarrow X$ and outputs $y := f(x)$. $\mathcal{A}_2$ outputs x.

Adversarial one-way functions (AOWFs)

Stateful 2-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$

1[st] attempt: $\begin{matrix} \mathcal{A}_1 \text{ outputs } y \\ \mathcal{A}_2 \text{ outputs } x \end{matrix}$ $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f(x) = y$

Unsatisfiable! $\mathcal{A}_1$ samples $x \leftarrow X$ and outputs $y := f(x)$. $\mathcal{A}_2$ outputs x.

Make the problem harder for the adversary:

2[nd] attempt: $\begin{matrix} \mathcal{A}_1 \text{ outputs } y \\ \mathcal{A}_2 \text{ outputs } x \end{matrix}$ $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f^n(x) = y$

Adversarial one-way functions (AOWFs)

Stateful 2-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$

1[st] attempt: $\mathcal{A}_1$ outputs y, $\mathcal{A}_2$ outputs x — $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f(x) = y$

Unsatisfiable! $\mathcal{A}_1$ samples $x \leftarrow X$ and outputs $y := f(x)$. $\mathcal{A}_2$ outputs x.

Make the problem harder for the adversary:

2[nd] attempt: $\mathcal{A}_1$ outputs y, $\mathcal{A}_2$ outputs x — $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f^n(x) = y$

If

- f is only sequentially computable

Adversarial one-way functions (AOWFs)

Stateful 2-stage adversary $(\mathcal{A}_1, \mathcal{A}_2)$

1st attempt: $\mathcal{A}_1$ outputs y, $\mathcal{A}_2$ outputs x — $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f(x) = y$

Unsatisfiable! $\mathcal{A}_1$ samples $x \leftarrow X$ and outputs $y := f(x)$. $\mathcal{A}_2$ outputs x.

Make the problem harder for the adversary:

2nd attempt: $\mathcal{A}_1$ outputs y, $\mathcal{A}_2$ outputs x — $(\mathcal{A}_1, \mathcal{A}_2)$ wins if $f^n(x) = y$

If

- f is only <u>sequentially</u> computable
- n depends on the runtime of $\mathcal{A}_1$ $(\mathrm{Time}_{\mathcal{A}_1} < n \cdot \mathrm{Time}_{f(\cdot)})$

there is no trivial attack!

Candidates for AOWFs

Our candidate: A cryptographic hash function

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

✓ Soundness

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

✓ Soundness

✓ Unbounded # recursions

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

✓ Soundness

✓ Unbounded # recursions

✗ Derivation privacy

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

✓ Soundness

✓ Unbounded # recursions

✗ Derivation privacy

✓ Derivation privacy if AOWF is unlinkable (requires randomization)

Candidates for AOWFs

Our candidate: A cryptographic hash function

- Justification: RO is an AOWFs

✓ Soundness

✓ Unbounded # recursions

✗ Derivation privacy

✓ Derivation privacy if AOWF is <u>unlinkable</u> (requires randomization)

  - For our candidate this can be achieved by inputting additional random bits in the hash function

Applications

- Reverse firewalls (RF)
    - First RF for NIZKs/SNARKs

Applications

- Reverse firewalls (RF)
    - First RF for NIZKs/SNARKs
- IND-RCCA secure rerandomizable PKE (adaptation of [CKLM12])
    - First post-quantum scheme

Applications

- Reverse firewalls (RF)
  - First RF for NIZKs/SNARKs
- IND-RCCA secure rerandomizable PKE (adaptation of [CKLM12])
  - First post-quantum scheme
- IND-RCCA secure updatable encryption (adaptation of [KLR19])
  - First scheme with full security
  - First post-quantum scheme

Applications

- Reverse firewalls (RF)
  - First RF for NIZKs/SNARKs
- IND-RCCA secure rerandomizable PKE (adaptation of [CKLM12])
  - First post-quantum scheme
- IND-RCCA secure updatable encryption (adaptation of [KLR19])
  - First scheme with full security
  - First post-quantum scheme
- Targeted non-malleable (TNM-CCA1) homomorphic encryption (HE) (non-generic) (adaptation of [BSW12])
  - Needs higher arity transformations
  - First unlinkable scheme
  - First scheme with unbounded homomorphic operations

Applications

- Reverse firewalls (RF)
    - First RF for NIZKs/SNARKs
- IND-RCCA secure rerandomizable PKE (adaptation of [CKLM12])
    - First post-quantum scheme
- IND-RCCA secure updatable encryption (adaptation of
    - First scheme with full security
    - First post-quantum scheme
- Targeted non-malleable (TNM-CCA1) homomorphic encryption (HE) (non-generic) (adaptation of [BSW12])
    - Needs higher arity transformations
    - First unlinkable scheme
    - First scheme with unbounded homomorphic operations

Requires assumption specific to the underlying HE scheme

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS
  - Counter-based approach (similar to [CKLM13])

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS
  - Counter-based approach (similar to [CKLM13])
  - AOWF-based approach is more flexible

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS
  - Counter-based approach (similar to [CKLM13])
  - AOWF-based approach is more flexible
- Many applications of malleable SNARKS
  - Reverse firewalls
  - Variants of Naor-Yung

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS
  - Counter-based approach (similar to [CKLM13])
  - AOWF-based approach is more flexible
- Many applications of malleable SNARKS
  - Reverse firewalls
  - Variants of Naor-Yung

Open problems

- More AOWF candidates
  - Relation to time-lock puzzles/verfiable delay functions/proofs of sequential work

Summary & Open problems

Our contributions

- Malleable NIZK's from recursive SNARKS
  - Counter-based approach (similar to [CKLM13])
  - AOWF-based approach is more flexible
- Many applications of malleable SNARKS
  - Reverse firewalls
  - Variants of Naor-Yung

Open problems

- More AOWF candidates
  - Relation to time-lock puzzles/verfiable delay functions/proofs of sequential work
- Good candidates for SNARKs with fast extraction

References I

Dan Boneh, Gil Segev, and Brent Waters.
Targeted malleability: homomorphic encryption for restricted computations.
In Shafi Goldwasser, editor, ITCS 2012: 3rd Innovations in Theoretical Computer Science, pages 350–366, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.
doi:10.1145/2090236.2090264.

Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn.
Malleable proof systems and applications.
In David Pointcheval and Thomas Johansson, editors, Advances in Cryptology – EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 281–300, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Heidelberg, Germany.
doi:10.1007/978-3-642-29011-4_18.

References II

Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn.
Succinct malleable NIZKs and an application to compact shuffles.
In Amit Sahai, editor, TCC 2013: 10th Theory of Cryptography Conference, volume 7785 of Lecture Notes in Computer Science, pages 100–119, Tokyo, Japan, March 3–6, 2013. Springer, Berlin, Heidelberg, Germany.
doi:10.1007/978-3-642-36594-2_6.

Michael Klooß, Anja Lehmann, and Andy Rupp.
(R)CCA secure updatable encryption with integrity protection.
In Yuval Ishai and Vincent Rijmen, editors, Advances in Cryptology – EUROCRYPT 2019, Part I, volume 11476 of Lecture Notes in Computer Science, pages 68–99, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland.
doi:10.1007/978-3-030-17653-2_3.

References III

Moni Naor and Moti Yung.
Public-key cryptosystems provably secure against chosen ciphertext attacks.
In 22nd Annual ACM Symposium on Theory of Computing, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press.
doi:10.1145/100216.100273.

Pictures

Alice, Bob, and other faces, Server: freepik.com
Matryoshka doll: holz-leute.de

Transformations with arity ≥ 2

$\boxed{x_1 \text{ is true}}$

$$\begin{matrix}\pi_1 \\ \pi_2\end{matrix} \xrightarrow{(T_x, T_w)} \pi'$$

$\boxed{x_2 \text{ is true}}$

Transformations with arity ≥ 2

x_1 is true

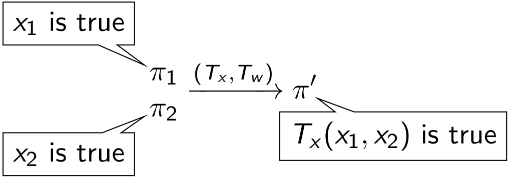
$$\begin{matrix} \pi_1 \\ \pi_2 \end{matrix} \xrightarrow{(T_x, T_w)} \pi'$$

x_2 is true

$T_x(x_1, x_2)$ is true

Counter-based approach: $\begin{matrix} (\ell_1, \pi_1) \\ (\ell_2, \pi_2) \end{matrix} \xrightarrow{(T_x, T_w)} (\max\{\ell_1, \ell_2\} + 1, \pi')$

Transformations with arity ≥ 2



Counter-based approach: $\begin{matrix}(\ell_1, \pi_1)\\(\ell_2, \pi_2)\end{matrix} \xrightarrow{(T_x, T_w)} (\max\{\ell_1, \ell_2\} + 1, \pi')$

- ✗ Only for constant-depth

Transformations with arity ≥ 2

x_1 is true

$$\begin{matrix} \pi_1 \\ \pi_2 \end{matrix} \xrightarrow{(T_x, T_w)} \pi'$$

x_2 is true

$T_x(x_1, x_2)$ is true

Counter-based approach: $\begin{matrix} (\ell_1, \pi_1) \\ (\ell_2, \pi_2) \end{matrix} \xrightarrow{(T_x, T_w)} (\max\{\ell_1, \ell_2\} + 1, \pi')$

- ✗ Only for constant-depth

AOWF-based approach:

- ✓ Unbounded depth
- Needs higher arity variant of AOWFs (works for hash functions)
- Statements and proofs must be input to the AOWF
- Extractor must cache extracted SNARKs