Malleable Commitments from Group Actions and Zero-Knowledge Proofs for Circuits based on Isogenies

Mingjie Chen, Yi-Fu Lai, **Abel Laval**, Laurane Marco, Christophe Petit

October 15, 2023

Overview

Asymmetric cryptography allows for a wide variety of schemes with interesting features :

- Threshold signatures
- Fully Homomorphic Encryption
- Zero-knowledge proofs
- Oblivious transfer
- Verifiable Delay Functions
- *etc...*

Asymmetric cryptography allows for a wide variety of schemes with interesting features :

- Threshold signatures
- Fully Homomorphic Encryption
- Zero-knowledge proofs
- Oblivious transfer
- Verifiable Delay Functions
- *etc...*

Problem : Shor's algorithm breaks classical asymmetric cryptography

Asymmetric cryptography allows for a wide variety of schemes with interesting features :

- Threshold signatures
- Fully Homomorphic Encryption
- Zero-knowledge proofs
- Oblivious transfer
- Verifiable Delay Functions
- *etc...*

Post quantum cryptography :

- Lattices
- Codes
- Isogenies
- Multivariates polynomials
- Hash functions

Problem : Shor's algorithm breaks classical asymmetric cryptography

Asymmetric cryptography allows for a wide variety of schemes with interesting features :

- Threshold signatures
- Fully Homomorphic Encryption
- **Zero-knowledge proofs**
- Oblivious transfer
- Verifiable Delay Functions
- *etc...*

Post quantum cryptography :

- Lattices
- Codes
- **Isogenies**
- Multivariates polynomials
- Hash functions

Problem : Shor's algorithm breaks classical asymmetric cryptography

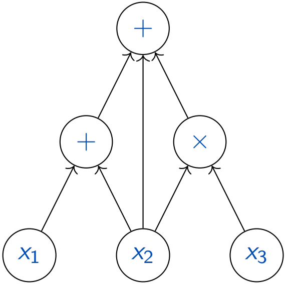
Proofs of knowledge, but... knowledge of what ?
of everything !

Proofs of knowledge, but... knowledge of what ?
of everything !

How ?

1. Construct a proof of knowledge for a NP-complete statement.
2. Reduce any other NP problem to this.

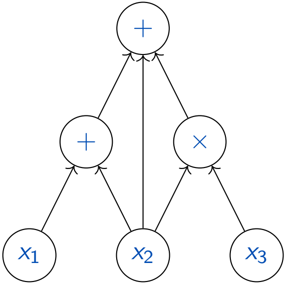
An arithmetic circuit encodes a polynomial.



$$\simeq (x_1 + x_2) + x_2 + x_2 x_3$$

Arithmetic Circuits

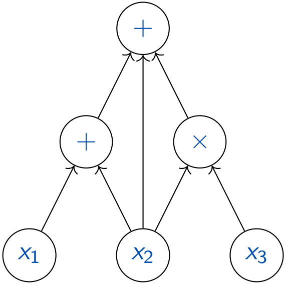
An arithmetic circuit encodes a polynomial.



$\simeq (x_1 + x_2) + x_2 + x_2 x_3$

- The SAT problem for arithmetic circuit :
  Given a polynomial f and an output value s, are there input values $x_1, \cdots, x_n$ such that $f(x_1, \cdots, x_n) = s$?

An arithmetic circuit encodes a polynomial.



$\simeq (x_1 + x_2) + x_2 + x_2 x_3$

- The SAT problem for arithmetic circuit :
  Given a polynomial f and an output value s, are there input values $x_1, \cdots, x_n$ such that $f(x_1, \cdots, x_n) = s$?

Theorem

The satisfiability problem for arithmetic circuits is NP-complete.

Commitment Schemes

Definition (Commitment Scheme)

A *commitment scheme* is a tuple $(\mathcal{M}, \mathcal{R}, \mathcal{C}, \mathsf{Commit}, \mathsf{Verify})$ where $\mathsf{Commit} : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ and $\mathsf{Verify} : \mathcal{M} \times \mathcal{R} \times \mathcal{C} \to \{0, 1\}$ are PPTA algorithms

With :

- $\mathcal{M}$: message space
- $\mathcal{R}$: randomness space
- $\mathcal{C}$: commitment space

Definition (Commitment Scheme)

A *commitment scheme* is a tuple $(\mathcal{M}, \mathcal{R}, \mathcal{C}, \mathsf{Commit}, \mathsf{Verify})$ where $\mathsf{Commit} : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ and $\mathsf{Verify} : \mathcal{M} \times \mathcal{R} \times \mathcal{C} \to \{0, 1\}$ are PPTA algorithms

With :

- $\mathcal{M}$: message space
- $\mathcal{R}$: randomness space
- $\mathcal{C}$: commitment space

Related security notions :

Hiding

An attacker cannot retrieve m or r from c.

Binding

It's hard to find $(m, r) \neq (m', r')$ that give the same commitment.

- Efficient solutions use homomophic property :

$$\forall m, m', r, r', \;\; \mathsf{Commit}(m + m', r + r') = \mathsf{Commit}(m, r) \cdot \mathsf{Commit}(m', r')$$

- Efficient solutions use homomophic property :

$$\forall m, m', r, r', \;\; \mathsf{Commit}(m+m', r+r') = \mathsf{Commit}(m, r) \cdot \mathsf{Commit}(m', r')$$

- Too restrictive for isogenies $\rightsquigarrow$ Relaxed notion : *malleability*.

Homomorphism and malleability

- Efficient solutions use homomophic property :

$$\forall m, m', r, r', \;\; \mathrm{Commit}(m + m', r + r') = \mathrm{Commit}(m, r) \cdot \mathrm{Commit}(m', r')$$

- Too restrictive for isogenies $\rightsquigarrow$ Relaxed notion : *malleability*.

Definition (Malleable Commitment)

A commitment scheme is malleable if :
Given a single commitment, we can derive a related second one.

Homomorphism and malleability

- Efficient solutions use homomophic property :

$$\forall m, m', r, r', \;\; \mathsf{Commit}(m + m', r + r') = \mathsf{Commit}(m, r) \cdot \mathsf{Commit}(m', r')$$

- Too restrictive for isogenies $\rightsquigarrow$ Relaxed notion : *malleability*.

Definition (Malleable Commitment)

A commitment scheme is malleable if :
Given a single commitment, we can derive a related second one.

We assume no structure on $\mathcal{M}$, $\mathcal{R}$ or $\mathcal{C}$ *a priori*.

- Efficient solutions use homomophic property :

$$\forall m, m', r, r', \; \mathsf{Commit}(m + m', r + r') = \mathsf{Commit}(m, r) \cdot \mathsf{Commit}(m', r')$$

- Too restrictive for isogenies $\rightsquigarrow$ Relaxed notion : *malleability*.

Definition (Malleable Commitment)

A commitment scheme is malleable if :
Given a single commitment, we can derive a related second one.

We assume no structure on $\mathcal{M}$, $\mathcal{R}$ or $\mathcal{C}$ *a priori*.

Very generic $\rightsquigarrow$ We adapt depending on available structure

(Admissible) Group Action Malleable Commitment

A *GAMC* is a commitment scheme exploiting additional structure for $\mathcal{M}$ and $\mathcal{R}$.

(Admissible) Group Action Malleable Commitment

A *GAMC* is a commitment scheme exploiting additional structure for $\mathcal{M}$ and $\mathcal{R}$.

Definition

A GAMC is a commitment scheme satisfying :

- $\mathcal{M}$ and $\mathcal{R}$ are groups. $\mathcal{C}$ is a set.
- We have a group action $\star : (\mathcal{M} \times \mathcal{R}) \times \mathcal{C} \to \mathcal{C}$
- $C_0 := \mathrm{Commit}(0_{\mathcal{M}}, 0_{\mathcal{R}})$
- $\mathrm{Commit}(m, r) := (m, r) \star C_0$.
- $(m', r') \star \mathrm{Commit}(m, r) = \mathrm{Commit}(m + m', r + r')$

(Admissible) Group Action Malleable Commitment

A *GAMC* is a commitment scheme exploiting additional structure for $\mathcal{M}$ and $\mathcal{R}$.

Definition

A GAMC is a commitment scheme satisfying :

- $\mathcal{M}$ and $\mathcal{R}$ are groups. $\mathcal{C}$ is a set.
- We have a group action $\star : (\mathcal{M} \times \mathcal{R}) \times \mathcal{C} \to \mathcal{C}$
- $C_0 := \mathsf{Commit}(0_{\mathcal{M}}, 0_{\mathcal{R}})$
- $\mathsf{Commit}(m, r) := (m, r) \star C_0$.
- $(m', r') \star \mathsf{Commit}(m, r) = \mathsf{Commit}(m + m', r + r')$

In our case :

- $\mathcal{M}$ and $\mathcal{R}$ are groups of isogenies (with composition).
- $\mathcal{C}$ is a set of elliptic curves (up to isomorphism).

less structure ⟵ generic malleability — GAMC — homomorphism property ⟶ more structure

How to use GAMCs

Several GAMC $\rightsquigarrow$ Proof systems for addition and multiplication gates $\rightsquigarrow$ Proof system for arithmetic circuits

Features :

- Addition : makes use of malleability $\rightsquigarrow$ small proofs.
- Multiplication : requires Merkle tree trick $\rightsquigarrow$ bigger proofs and $|\mathcal{M}|$ restricted.

Interlude : CSIDH

- CSIDH (for *Commutative Supersingular Isogeny Diffie-Hellman*) is a key exchange mechanism.
- Analog of the Diffie-Hellman for isogenies.

Interlude : CSIDH

- CSIDH (for *Commutative Supersingular Isogeny Diffie-Hellman*) is a key exchange mechanism.
- Analog of the Diffie-Hellman for isogenies.

Diffie-Hellman	CSIDH
$\xrightarrow{g^a}$	$\xrightarrow{\mathfrak{a}\cdot E_0}$
$\xleftarrow{g^b}$	$\xleftarrow{\mathfrak{b}\cdot E_0}$
$g^{ab} = g^{ba}$	$\mathfrak{a}\mathfrak{b} \cdot E_0 = \mathfrak{b}\mathfrak{a} \cdot E_0$

- CSIDH (for *Commutative Supersingular Isogeny Diffie-Hellman*) is a key exchange mechanism.
- Analog of the Diffie-Hellman for isogenies.

Diffie-Hellman	CSIDH
$\xrightarrow{g^a}$	$\xrightarrow{\mathfrak{a}\cdot E_0}$
$\xleftarrow{g^b}$	$\xleftarrow{\mathfrak{b}\cdot E_0}$
$g^{ab} = g^{ba}$	$\mathfrak{a}\mathfrak{b} \cdot E_0 = \mathfrak{b}\mathfrak{a} \cdot E_0$

- $\mathfrak{a}$ and $\mathfrak{b}$ are ideals in $\mathcal{C}\ell(\mathcal{O})$: the *ideal class group* of $\mathbb{Z}[\pi]$.
- E_0 is a curve in SS_p : the set of supersingular curves "over $\mathbb{F}_p$".
- Natural group action of $\mathcal{C}\ell(\mathcal{O})$ over SS_p.
  This is the underlying group action of the GAMC.

An instance of a GAMC

In the CSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathcal{C}\ell(\mathcal{O})$ are groups of ideals (encoding isogenies).
- $\mathcal{C} := \mathsf{SS}_p \times \mathsf{SS}_p$.
- $C_0 := (E_0, E_1)$.

Malleability is given by

$$(\mathfrak{m}, \mathfrak{r}) \star (E, E') := (\mathfrak{r} \cdot E, \mathfrak{m}\mathfrak{r} \cdot E')$$

Enough to define a GAMC for arithmetic circuits !

An instance of a GAMC

In the CSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathcal{C}\ell(\mathcal{O})$ are groups of ideals (encoding isogenies).
- $\mathcal{C} := \mathsf{SS}_p \times \mathsf{SS}_p$.
- $C_0 := (E_0, E_1)$.

Malleability is given by

$$(\mathfrak{m}, \mathfrak{r}) \star (E, E') := (\mathfrak{r} \cdot E, \mathfrak{m}\mathfrak{r} \cdot E')$$

Enough to define a GAMC for arithmetic circuits !

But there exist alternatives to circuits :

An instance of a GAMC

In the CSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathcal{C}\ell(\mathcal{O})$ are groups of ideals (encoding isogenies).
- $\mathcal{C} := \mathsf{SS}_p \times \mathsf{SS}_p$.
- $C_0 := (E_0, E_1)$.

Malleability is given by

$$(\mathfrak{m}, \mathfrak{r}) \star (E, E') := (\mathfrak{r} \cdot E, \mathfrak{m}\mathfrak{r} \cdot E')$$

Enough to define a GAMC for arithmetic circuits !

But there exist alternatives to circuits :

Rank 1 Constraint Systems (R1CS) and Branching programs

In the CSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathcal{C}\ell(\mathcal{O})$ are groups of ideals (encoding isogenies).
- $\mathcal{C} := \mathsf{SS}_p \times \mathsf{SS}_p$.
- $C_0 := (E_0, E_1)$.

Malleability is given by

$$(\mathfrak{m}, \mathfrak{r}) \star (E, E') := (\mathfrak{r} \cdot E, \mathfrak{m}\mathfrak{r} \cdot E')$$

Enough to define a GAMC for arithmetic circuits !

But there exist alternatives to circuits :

Rank 1 Constraint Systems (R1CS) and **Branching programs** .

Branching programs

A *branching program* is another way to encode an arithmetic circuit.

A *branching program* is another way to encode an arithmetic circuit.

Definition (Branching Program)

Given matrices $\{(A_{1,0}, A_{1,1}), \ldots, (A_{d,0}, A_{d,1})\}$ in $M_2(\mathbb{Z}_n)$, a branching program of length d takes as input :

- An initial state $M_0 \in M_2(\mathbb{Z}_n)$
- $x = (x_1, \ldots, x_d) \in \{0, 1\}^d$

and outputs :

$$A_{d,x_d} \cdot A_{d-1,x_{d-1}} \cdot \ldots \cdot A_{1,x_1} \cdot M_0$$

Branching programs

A *branching program* is another way to encode an arithmetic circuit.

Definition (Branching Program)

Given matrices $\{(A_{1,0}, A_{1,1}), \ldots, (A_{d,0}, A_{d,1})\}$ in $M_2(\mathbb{Z}_n)$, a branching program of length d takes as input :

- An initial state $M_0 \in M_2(\mathbb{Z}_n)$
- $x = (x_1, \ldots, x_d) \in \{0,1\}^d$

and outputs :

$$A_{d,x_d} \cdot A_{d-1,x_{d-1}} \cdot \ldots \cdot A_{1,x_1} \cdot M_0$$

Theorem (Barrington)

Every arithmetic circuit can be encoded as a branching program.

GAMC with branching programs

A GAMC where $\mathcal{M}$ and $\mathcal{R}$ are 2x2 matrices $\rightsquigarrow$ Proof systems for branching programs $\rightsquigarrow$ More efficient than arithmetic circuits

Features :

- Requires more structure than arithmetic circuits.
- $\mathcal{M}$ and $\mathcal{R}$ are rings $\rightsquigarrow$ no restriction on $|\mathcal{M}|$.
- Smaller proofs than arithmetic circuits.

GAMC with branching programs

A GAMC where $\mathcal{M}$ and $\mathcal{R}$ are 2x2 matrices $\rightsquigarrow$ Proof systems for branching programs $\rightsquigarrow$ More efficient than arithmetic circuits

Features :

- Requires more structure than arithmetic circuits.
- $\mathcal{M}$ and $\mathcal{R}$ are rings $\rightsquigarrow$ no restriction on $|\mathcal{M}|$.
- Smaller proofs than arithmetic circuits.

A example of GAMC for branching programs ?

GAMC with branching programs

Two ingredients :

- pSIDH, a key exchange mechanism using isogenies.
- The *malleability oracle* from [KMPW21].

GAMC with branching programs

Two ingredients :

- pSIDH, a key exchange mechanism using isogenies.
- The *malleability oracle* from [KMPW21].

For a give curve E_0, we define two sets :

- $\mathrm{End}(E_0)_N$: Ring of endomorphisms of E_0 with norm coprime with N.
- $\mathrm{SS}(E_0)_N$: Curves at distance N from E_0.

Two ingredients :

- pSIDH, a key exchange mechanism using isogenies.
- The *malleability oracle* from [KMPW21].

For a give curve E_0, we define two sets :

- $\mathrm{End}(E_0)_N$: Ring of endomorphisms of E_0 with norm coprime with N.
- $\mathrm{SS}(E_0)_N$: Curves at distance N from E_0.

Malleability oracle f : allows the computation of a group action
$\mathrm{End}(E_0)_N \times \mathrm{SS}(E_0)_N \to \mathrm{SS}(E_0)_N$.
This is the underlying group action of the GAMC.

In the pSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathrm{End}(E_0)_N$ are **rings** of endomorphisms (encoding isogenies).
- $\mathcal{C} := \mathrm{SS}(E_0)_N$.
- $C_0 := E_1 \in \mathrm{SS}(E_0)_N$

Malleability is given by

$$(\theta_m, \theta_r) \star E := f(\theta_m \theta_r, E)$$

GAMC with branching programs

In the pSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathrm{End}(E_0)_N$ are **rings** of endomorphisms (encoding isogenies).
- $\mathcal{C} := \mathrm{SS}(E_0)_N$.
- $C_0 := E_1 \in \mathrm{SS}(E_0)_N$

Malleability is given by

$$(\theta_m, \theta_r) \star E := f(\theta_m \theta_r, E)$$

There is an explicit isomorphism :

$$\mathrm{End}(E_0)_N \simeq \mathrm{GL}_2(\mathbb{Z}_N)$$

GAMC with branching programs

In the pSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathsf{End}(E_0)_N$ are **rings** of endomorphisms (encoding isogenies).
- $\mathcal{C} := \mathsf{SS}(E_0)_N$.
- $C_0 := E_1 \in \mathsf{SS}(E_0)_N$

Malleability is given by

$$(\theta_m, \theta_r) \star E := f(\theta_m \theta_r, E)$$

There is an explicit isomorphism :

$$\mathsf{End}(E_0)_N \simeq \mathsf{GL}_2(\mathbb{Z}_N)$$

$\rightsquigarrow$ Proof system for branching programs !

GAMC with branching programs

In the pSIDH setting :

- $\mathcal{M} = \mathcal{R} := \mathrm{End}(E_0)_N$ are **rings** of endomorphisms (encoding isogenies).
- $\mathcal{C} := \mathrm{SS}(E_0)_N$.
- $C_0 := E_1 \in \mathrm{SS}(E_0)_N$

Malleability is given by

$$(\theta_m, \theta_r) \star E := f(\theta_m \theta_r, E)$$

There is an explicit isomorphism :

$$\mathrm{End}(E_0)_N \simeq \mathrm{GL}_2(\mathbb{Z}_N)$$

$\rightsquigarrow$ Proof system for branching programs !

But... pSIDH is no longer post-quantum secure. [CII+23]

Conclusion

Contributions :

- New framework for generic NP statements ZK proofs.
- Highly versatile.
- Proof-of-concept construction.

Performances :

- Strong security assumptions and no trusted setup.
- Proof system for an arithmetic circuit $= O(|\mathcal{M}|)$ malleability computations.
- Size of the proof $= O(\lambda|\mathcal{M}|)$ bits.

Future work :

- Proof-of-concept cannot use higher security parameters than CSIDH-512.
- Post-quantum constructions for R1CS and branching programs

Mingjie Chen, Muhammad Imran, Gábor Ivanyos, Péter Kutas, Antonin Leroux, and Christophe Petit.
Hidden stabilizers, the isogeny to endomorphism ring problem and the cryptanalysis of psidh.
Cryptology ePrint Archive, Paper 2023/779, 2023.
https://eprint.iacr.org/2023/779.

Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Charlotte Weitkämper.
One-way functions and malleability oracles: Hidden shift attacks on isogeny-based protocols.
In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 242–271, Cham, 2021. Springer International Publishing.