Universally Composable NIZK from Sigma Protocols via a New Straight-line Compiler

Pousali Dey

Indian Statistical Institute, Kolkata

Joint work with Megan Chen, Chaya Ganesh, Pratyay Mukherjee, Pratik Sarkar, Swagata Sasmal

PKC 2025 | Røros, Norway

Roadmap of today's talk

1. New Straight line extractable compiler for Sigma protocols for algebraic relations.
2. Simulation Extractability of Compiled NIZK for algebraic relations.
3. UC secure NIZK from Sigma protocol for algebraic relations.
4. Instantiation of the compiler from Class groups.

Motivation for our work

- NIZK for algebraic relations are used in the applications such as-
  Multi-party threshold signature protocols [Lin22, KG20], protocols based on PKI, and Distributed Verifiable Random function [GLOW21],[KMMM23] .
- Popular way to achieve NIZK is-
  Fiat-Shamir Compiled Sigma protocols.
- Applications crucially need-
  UC Security
- Issue-
  Fiat-Shamir compiled Sigma protocols are not UC secure.

Sigma Protocol

Prover

Verifier

Wants to prove the knowledge of the witness **w** for **x** to the verifier.

Commit

Challenge

Response

Accept/Reject

**This talk will particularly focus on Schnorr Protocol

Schnorr Protocol

Example of Sigma Protocol for proving the knowledge of Discrete Logarithm of x where $x = g^w$

g be a generator of a group G of order q

Prover

Verifier

$r \leftarrow Z_q \quad a = g^r$

a →

c ←

$c \xleftarrow{\$} Z_q$

$z = r + c.w \mod q$

z →

Checks if $g^z \stackrel{?}{=} a.x^c$

Accept/Reject

Sigma protocols satisfy 3 properties:

1. **Completeness**: Honest Prover P convinces the Verifier V of a true statement x .

2. **Knowledge Soundness:** An efficient knowledge extractor $\mathcal{E}$ with oracle access to P^* can extract a valid witness w for a statement x accepted by the verifier.

3. **Honest Verifier Zero Knowledge:** Prover P convinces correctness of a statement x to the Verifier V without revealing any other information beyond that fact.

Knowledge Extraction

Malicious Prover

Knowledge Extractor

$r \leftarrow Z_q \quad a = g^r$

a

$c \xleftarrow{\$} Z_q$

c

$z = r + c.w \mod q$

z

$c' \xleftarrow{\$} Z_q$ where $c \neq c'$

c'

$z' = r + c'.w \mod q$

z'

$g^{z-z'} = x^{c-c'} = g^{w(c-c')}$

$=> z - z' = w(c - c')$

$=> w = \frac{z - z'}{c - c'}$

Extraction requires rewinding the prover

Roadmap of today's talk

1. New Straight line extractable compiler for Sigma protocols for algebraic relations .
2. Simulation Extractability of Compiled NIZK.
3. **UC secure NIZK from Sigma protocol for algebraic relations.**
4. Instantiation of the compiler from Class groups.

Universally Composable Security Model(UC)

UC: Guarantees that a protocol remains secure even if it is composed with arbitrarily many instances of the same protocol or other protocols which run concurrently.

Rewinding the prover is not UC-compatible.

- In UC framework, there is an environment Z, which is an interactive distinguisher between the real protocol and the ideal protocol.
- A simulator **Sim** in ideal protocol cannot rewind Z.
- Z can distinguish the real protocol and ideal protocol.

Roadmap of today's talk

1. New Straight line extractable compiler for Sigma protocols for algebraic relations.
2. Simulation Extractability of Compiled NIZK.
3. **UC secure NIZK from Sigma protocol for algebraic relations.**
   - A popular way to obtain a NIZK is Fiat-Shamir Transformation.
4. Instantiation of the compiler from Class groups.

Non Interactive Zero Knowledge Proof(NIZK)

NIZK Proof: Used in the situation where parties are not able to interact between themselves.

Fiat-Shamir Transform

Prover **Verifier**

$r \leftarrow Z_q \quad a = g^r$

a

$c = H(x, a)$

c

$c \leftarrow Z_q$

Checks if

$z = r + c.w \mod q$

a, c, z

$c \stackrel{?}{=} H(x, a)$

$g^z \stackrel{?}{=} a.x^c$

** Where H is a hash fuction modeled as Random Oracle.

Accept/Reject

Fiat-Shamir compiled NIZKs from Sigma Protocols are not UC compatible, as they require rewinding the prover like in the interactive version.

- Examples of other compilers which are **straight line extractable** UC compatible NIZK in ROM:

Pass Compiler(Crypto'03) , Fischlin Compiler(Crypto'05).

01

Pass compiler requires repetition of the underlying Sigma protocol for security parameter (k) where k is as large as 128.

128 times overhead over FS compiled NIZK.

02

Fischlin's compiler requires a proof-of-work from Prover's side.
It also requires repetition of underlying Sigma protocol.

15 times overhead over FS compiled NIZK
when applied to Schnorr protocol **[Chen, Lindell CiC 2024]**

Roadmap of today's talk

1

New Straight line extractable compiler for Sigma protocols for algebraic relations.

- Use Additively Homomorphic Encryption Scheme as a building block.

2

Simulation Extractability of Compiled NIZK.

3

UC secure NIZK from Sigma protocol for algebraic relations.

4

Instantiation of the compiler from Class groups.

Additively Homomorphic Encryption Scheme(AHE)

An AHE =(KGen, Enc, Dec) satisfies:

Additive Homomorphism: Define $c^{\times} = c_1 + sc_2$ for $c_1, c_2 \in C$ and for scalar s, if $c_1 = Enc(m_1), c_2 = Enc(m_2)$, then $c^{\times} = Enc(m_1 + sm_2)$.

Additionally, we need two other properties from the AHE

1 **Homomorphic Well formedness:** Let $c_1, c_2 \in C$ and $c^{\times} = c_1 + sc_2$ for scalar s , if $m^{\times} = Dec(c^{\times})$, then $m^{\times} = m_1 + sm_2$.

2 **Oblivious Sampleability of pk:** There exists a poly time hash function H_{pk} such that public key pk can be sampled obliviously as $pk = H_{pk}\left(1^k, x\right)$ on an uniform random input x and the following distributions are computationally indistingushable.

$$\{pk : (sk, pk) \leftarrow KGen\} \quad \text{and} \quad \{pk \leftarrow H_{pk}\left(1^k,\right)\}$$

Compiler

g be a generator of a group G of order q

crs=$\{pk\}$, td=$\{sk\}$

knows x, w, pk

Prover

knows x

Verifier

- $s \leftarrow Z_q$,$S = g^s$,
  $r_s; r_w \leftarrow Z_q$, are encryption randomnesses,
  $C_s = Enc(pk, s, r_s)$, $C_w = Enc(pk, w, r_w)$,
- $a = (S, C_s, C_w)$,

a →

c ←

z, r_z →

- $z = s + c.w \ mod \ q$
  $r_z = r_s + c.r_w \ mod \ q$

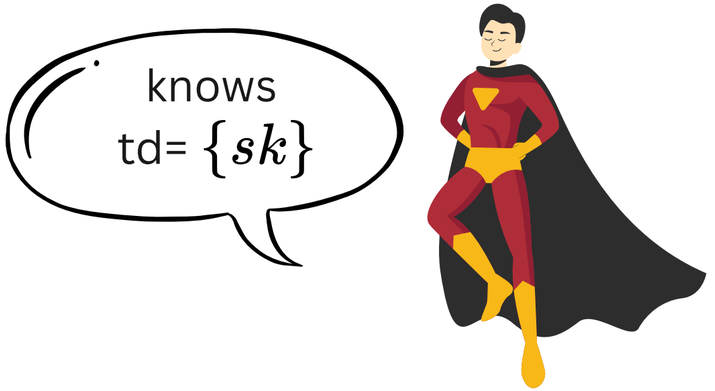
$c \xleftarrow{\$} Z_q$

- $g^z \stackrel{?}{=} S.x^c$
- check if C_s, C_w are valid ciphertexts.
- $Enc(pk, z, r_z) \stackrel{?}{=} C_s + c.C_w$

Accept/Reject

Straight line extractability of our protocol

Malicious Prover



Knowledge Extractor

a

c

z, r_z

Parse a as (S, C_s, C_w).

Decrypt C_w with sk and receive w.

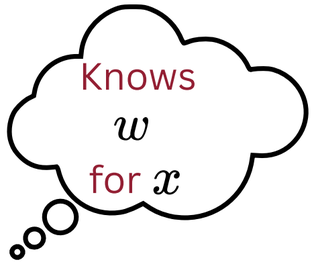
01 **Our compiled protocol is interactive.**

Achieve Non-Interactivity by Fiat-Shamir transformation using a hash function H in ROM .

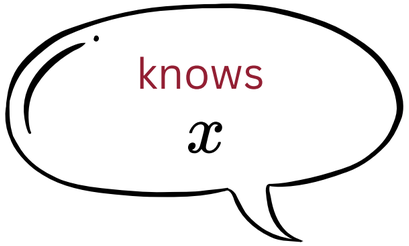
02 **Even in the presence of RO, the protocol requires a trusted CRS setup for the AHE key generation.**

Instantiate the AHE with a scheme in which public keys are obliviously sampleable using a hash function H_{pk} .

Transforming our protocol to NIZK in ROM



Prover



Verifier

- Parse RO as H_{pk}, H.
- Compute $pk = H_{pk}\left(1^k, x\right)$.
- Compute $a = (S, C_s, C_w)$, as earlier.
- $c = H\left(x, a\right)$
- Compute z, r_z as earlier.

a, c, z →

- Parse RO as H_{pk}, H
- Compute $pk = H_{pk}\left(1^k, x\right)$
- Check $c \stackrel{?}{=} H\left(x, a\right)$
- Check $g^z \stackrel{?}{=} S.x^c$
- Check if C_s, C_w are valid ciphertexts.
- $Enc\left(pk, z, r_z\right) \stackrel{?}{=} C_s + c.C_w$

Accept/Reject

Protocols	Straight-line extractable	Transparent Setup
Basic Sigma Protocol	No	Yes
Pass 03	Yes	Yes
Fischlin 05	Yes	Yes
Our work with any AHE	Yes	No
Our work with class group based AHE	Yes	Yes

Comparing our protocol with other compilers in terms of Straight line extractability and Setup.

Roadmap of today's talk

1. New Straight line extractable compiler for Sigma protocols algebraic relations.

2. Simulation Extractability of Compiled NIZK.

3. UC secure NIZK from Sigma protocol algebraic relations.

4. **Instantiation of the compiler from Class groups.**

Concrete Instantiation

Concrete instantiation of AHE using Class Groups

We show that Class Group based instantiation satisfies:

- **Additive Homomorphism.**
- **Homomorphic Well-formedness.**
- **Oblivious Sampleability of public key.**

AHE	Oblivious Sampleability of pk	Homomorphic well-formedness
Paillier	No	Yes
Regev	Yes	Yes
Class Group	Yes	Yes

Popular instantiations of AHE and their support for the two desired properties

Roadmap of today's talk

1. New Straight line extractable compiler for Sigma protocols algebraic relations.
2. **Simulation Extractability of Compiled NIZK.**
3. UC secure NIZK from Sigma protocol for algebraic relations.
4. Instantiation of the complier from Class groups.

Simulation Extractability of FS compiled NIZK

1

Simulation Extractability: With access to the Simulation oracle, whenever adversary P* outputs a valid statement-proof pair (x, π), it is possible to extract a valid witness w from that pair .

Standard technique to show Simulation Extractability is: **Unique Response Property.**

2

Unique Response Property: Adversary P* with access to the Simulation oracle, can't find two accepting transcripts (a, c, z) and (a, c, z') such that $z \neq z'$.

Our compiled protocol satisfies Unique Response Property.

Our compiled protocol satisfies Straight line and Simulation Extractability.

Straight line and Simulation Extractability together gives us UC Security.

Our NIZK protocol is UC secure

Applications

- Our compiler is convenient to compile a sigma protocol for an algebraic relation with m algebraic statements and n witnesses where $m > n$. Example: **Chaum-Pedersen Protocol**.

- All applications of Schnorr and Chaum-Pedersen such as **multi-party threshold signature protocols, protocols based on PKI, and Distributed Verifiable Random function** achieve UC security without repetition of the underlying sigma protocol.

Summary

1. Compile Sigma protocol into a straight line extractable protocol using an Additively Homomorphic Encryption scheme in transparent CRS model.

2. Apply Fiat-Shamir transform on this compiled protocol to obtain a NIZK which is straight line extractable.

3. Show that resulting NIZK is UC secure.

4. Unlike prior works, our compiler requires no repetitions of the underlying Sigma protocol.

Thank you!

https://eprint.iacr.org/2024/1713