Zero-Knowledge Arguments for Subverted RSA Groups

Dimitris Kolonelos

IMDEA Software Institute
& Universidad Politecnica de Madrid

Mary Maller

Ethereum Foundation
& PQShield

Mikhail Volkhov

The University of Edinburgh

PKC 2023, Atlanta
10 May 2023

Hidden Order Groups
(aka Groups of Unknown Order)

Group $\mathbb{G}$ where computing $\mathrm{ord}(\mathbb{G})$ is hard

Hidden Order Groups
(aka Groups of Unknown Order)

Group $\mathbb{G}$ where computing ord($\mathbb{G}$) is hard

wlog
multiplicative

computationally

Group exponentiation: $g^{x \underbrace{\bmod \mathrm{ord}(g)}_{\text{unknown}}} = g^x$

Hidden Order Groups
(aka Groups of Unknown Order)

Group $\mathbb{G}$ where computing $\text{ord}(\mathbb{G})$ is hard

wlog multiplicative

computationally

Group exponentiation: $g^{x \underbrace{\bmod \text{ord}(g)}_{\text{unknown}}} = g^x$

Group exponentiations are (computationally) over $\mathbb{Z}$

Hidden Order Groups - Applications

List not exhaustive

Group exponentiations are (computationally) over $\mathbb{Z}$

- ★ **Accumulators** [BdM94, BP97, CL02, LLX07, L12, BBF19] & **Set Membership (zk-)proofs** [CL02, BCFGK19, CDHKKO22]
- ★ (zk-)**Range proofs** [B00, L03, G05, CPP17, CKLR21]
- ★ **Vector Commitments** [CF13, LM19, BBF19, CFGKN20, TXN20, CFKS22]
- ★ **Polynomial commitments** & **SNARKs** [BFS20, BHRRS21, AGLMS23]
- ★ **Verifiable delay functions** [BBBF18, W19, P19] & **time-lock puzzles** [RSW96]
- ★ **Additively Homomorphic Encryption** [P99, CL15]

Hidden Order Groups - Applications

List not exhaustive

Group exponentiations are (computationally) over $\mathbb{Z}$

★ **Accumulators** [BdM94, BP97, CL02, LLX07, L12, BBF19] & **Set Membership (zk-)proofs** [CL02, BCFGK19, CDHKKO22]

★ (zk-)**Range proofs** [B00, L03, G05, CPP17, CKLR21]

★ **Vector Commitments** [CF13, LM19, BBF19, CFGKN20, TXN20, CFKS22]

[AGLMS23]: next talk!

★ **Polynomial commitments** & **SNARKs** [BFS20, BHRRS21, AGLMS23]

★ **Verifiable delay functions** [BBBF18, W19, P19] & **time-lock puzzles** [RSW96]

★ **Additively Homomorphic Encryption** [P99, CL15]

Hidden Order Groups - Applications

List not exhaustive

Group exponentiations are (computationally) over $\mathbb{Z}$

- ★ **Accumulators** [BdM94, BP97, CL02, LLX07, L12, BBF19] & **Set Membership (zk-)proofs** [CL02, BCFGK19, CDHKKO22]
- ★ (zk-)**Range proofs** [B00, L03, G05, CPP17, CKLR21]
- ★ **Vector Commitments** [CF13, LM19, BBF19, CFGKN20, TXN20, CFKS22]
- ★ **Polynomial commitments** & **SNARKs** [BFS20, BHRRS21, AGLMS23]
- ★ **Verifiable delay functions** [BBBF18, W19, P19] & **time-lock puzzles** [RSW96]
- ★ **Additively Homomorphic Encryption** [P99, CL15]

[AGLMS23]: next talk!

Hidden Order Groups - Instantiations

RSA Groups [RSA78]

- $\mathbb{Z}_N^*$: $N = p \cdot q$ and p, q 'safe' primes
- $g^x \mod N = g^{x \mod \phi(N)}$

  ord$(\mathbb{G}) = \phi(N) = (p-1)(q-1)$
- Computing ord$(\mathbb{G}) \leftrightarrow$ Factoring 🙂
- Structured setup ☹️

Class Groups [BW88]

- Class Groups of imaginary quadratic order
- Complicated Algebra... (see

  [BuchamannHamdy01, Straka19])
- Computing ord$(\mathbb{G}) \leftrightarrow$ Less cryptanalysis effort ☹️
- Uniformly random setup 🙂

Hidden Order Groups - Instantiations

RSA Groups [RSA78]

- $\mathbb{Z}_N^*$: $N = p \cdot q$ and p, q 'safe' primes
- $g^x \mod N = g^{x \mod \phi(N)}$

  $\mathrm{ord}(\mathbb{G}) = \phi(N) = (p-1)(q-1)$
- Computing $\mathrm{ord}(\mathbb{G}) \leftrightarrow$ Factoring 🙂
- Structured setup ☹️

Class Groups [BW88]

- Class Groups of imaginary quadratic order
- Complicated Algebra… (see

  [BuchamannHamdy01, Straka19])
- Computing $\mathrm{ord}(\mathbb{G}) \leftrightarrow$ Less cryptanalysis effort ☹️
- Uniformly random setup 🙂

Subverted RSA Groups

- Natural setting when the setup is adversarial
- E.g. $N = pq$ but not safe primes, $N = p_1 p_2 \ldots p_k$, $N = p_1^{k_1} p_2^{k_2}$

Hidden Order Groups - Instantiations

RSA Groups [RSA78]

- $\mathbb{Z}_N^*$: $N = p \cdot q$ and p, q 'safe' primes
- $g^x \mod N = g^{x \mod \phi(N)}$

  ord($\mathbb{G}$) $= \phi(N) = (p-1)(q-1)$
- Computing ord($\mathbb{G}$) $\leftrightarrow$ Factoring 🙂
- Structured setup ☹️

Class Groups [BW88]

- Class Groups of imaginary quadratic order
- Complicated Algebra… (see

  [BuchamannHamdy01, Straka19])
- Computing ord($\mathbb{G}$) $\leftrightarrow$ Less cryptanalysis effort ☹️
- Uniformly random setup 🙂

Subverted RSA Groups

- Natural setting when the setup is adversarial
- E.g. $N = pq$ but not safe primes, $N = p_1 p_2 \ldots p_k$, $N = p_1^{k_1} p_2^{k_2}$

This Work:

ZK proofs over Subverted RSA groups

Paillier Encryption & Key Subversion

Fully additive variant [DJ10, L17]

RSA modulus

❖KeyGen(1^λ) : $N = p \cdot q$, $h \leftarrow g^N \in \mathbb{Z}^2_N$, $\mathsf{pk} = (N, h)$, $\mathsf{sk} = (p, q)$

❖Enc(pk, m) : $\mathsf{ct} \leftarrow (N+1)^m h^r \mod N^2$

❖Dec$(\mathsf{sk}, \mathsf{ct})$: $([\mathsf{ct} \cdot \mathsf{ct}^{[N^{-1} \mod \phi(N)] \cdot N} \mod N^2] - 1)/N$

Paillier Encryption & Key Subversion

RSA modulus

Fully additive variant [DJ10, L17]

❖KeyGen(1^λ) : $N = p \cdot q$, $h \leftarrow g^N \in \mathbb{Z}^2_N$, pk = (N, h), sk = (p, q)

❖Enc(pk, m) : ct $\leftarrow (N+1)^m h^r \mod N^2$

❖Dec(sk, ct) : $([\mathsf{ct} \cdot \mathsf{ct}^{[N^{-1} \mod \phi(N)] \cdot N} \mod N^2] - 1)/N$

Why would anyone subvert their own key? 🤔

Encrypt-(under-your-own-key)-and-prove

- MPC Ceremony for RSA modulus generation [HMRT19]
- Threshold ECDSA [CGGMP20, BMP22]
- E-Voting [DJ01]

Zero-Knowledge Proofs [GMR89]

$R(x, w) = 1$

$\mathscr{P}(\mathsf{vpk}, x, w)$

π

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

$\mathscr{V}(\mathsf{vsk}, x)$

$\mathsf{Verify}(\mathsf{vsk}, x, \pi) = 1$

- (Proof/)Argument of Knowledge ((P/)AoKs): if $\mathsf{Verify}(x, \pi) = 1$ then $\mathscr{P}$ knows a w' s.t. $R(x, w') = 1$
- Zero-Knowledge (ZK): $\mathscr{V}$ learns nothing about w
- Non-interactive (NI): $\mathscr{P}$ generates π without any interaction with $\mathscr{V}$
- (Public/)Designated-Verifier (DV) [DFN06, PsV06]: (every $\mathscr{V}$/)only one $\mathscr{V}$ holding vsk can verify π

Zero-Knowledge Proofs [GMR89]

$R(x, w) = 1$

$\mathscr{P}(\mathsf{vpk}, x, w)$

$\mathscr{V}(\mathsf{vsk}, x)$

$\xrightarrow{\pi}$

$\mathscr{P}$ Computationally Unbounded/Bounded

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

$\mathsf{Verify}(\mathsf{vsk}, x, \pi) = 1$

- (Proof/)Argument of Knowledge ((P/)AoKs): if $\mathsf{Verify}(x, \pi) = 1$ then $\mathscr{P}$ knows a w' s.t. $R(x, w') = 1$
- Zero-Knowledge (ZK): $\mathscr{V}$ learns nothing about w
- Non-interactive (NI): $\mathscr{P}$ generates π without any interaction with $\mathscr{V}$
- (Public/)Designated-Verifier (DV) [DFN06, PsV06]: (every $\mathscr{V}$/)only one $\mathscr{V}$ holding vsk can verify π

Zero-Knowledge Proofs [GMR89]

Public verifiability: $vsk = \perp$

$R(x, w) = 1$

$\mathscr{P}(\mathsf{vpk}, x, w)$

$\mathscr{V}(\mathsf{vsk}, x)$

$\xrightarrow{\quad\pi\quad}$

$\mathscr{P}$ Computationally Unbounded/Bounded

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

$\mathsf{Verify}(\mathsf{vsk}, x, \pi) = 1$

- (Proof/)Argument of Knowledge ((P/)AoKs): if $\mathsf{Verify}(x, \pi) = 1$ then $\mathscr{P}$ knows a w' s.t. $R(x, w') = 1$
- Zero-Knowledge (ZK): $\mathscr{V}$ learns nothing about w
- Non-interactive (NI): $\mathscr{P}$ generates π without any interaction with $\mathscr{V}$
- (Public/)Designated-Verifier (DV) [DFN06, PsV06]: (every $\mathscr{V}$/)only one $\mathscr{V}$ holding vsk can verify π

Zero-Knowledge Proofs [GMR89]

Public verifiability: $vsk = \perp$

$R(x, w) = 1$

$\mathscr{P}(\mathsf{vpk}, x, w)$

π

$\mathscr{V}(\mathsf{vsk}, x)$

$\mathscr{P}$ Computationally Unbounded/Bounded

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

$\mathsf{Verify}(\mathsf{vsk}, x, \pi) = 1$

- (Proof/)Argument of Knowledge ((P/)AoKs): if $\mathsf{Verify}(x, \pi) = 1$ then $\mathscr{P}$ knows a w' s.t. $R(x, w') = 1$
- Zero-Knowledge (ZK): $\mathscr{V}$ learns nothing about w
- Non-interactive (NI): $\mathscr{P}$ generates π without any interaction with $\mathscr{V}$
- (Public/)Designated-Verifier (DV) [DFN06, PsV06]: (every $\mathscr{V}$/)only one $\mathscr{V}$ holding vsk can verify π

This work: DV-NIZK-AoK protocols

Our contributions

★A DV-zk-AoK of a pre-image of any (additive) homomorphism under subverted RSA groups.

e.g. $x : y = g^x \pmod{N}$, $x : \mathsf{ct} = \mathsf{Paillier}.\mathsf{Enc}(x)$ for **any** N

★A DV-zk range argument for any (additive) homomorphism under subverted RSA groups.

e.g. $x : \mathsf{ct} = \mathsf{Paillier}.\mathsf{Enc}(x) \wedge x \in [A, B]$ for **any** N

★Technically: A new extraction technique for proving knowledge-soundness.

Possible Approaches

N: arbitrary chosen by $\mathscr{P}$ $y = g^x \pmod{N}$

❖ Σ-protocols

➜ λ repetitions (xλ efficiency overhead) [BCK10, TW12]

❖ General purpose NIZK (e.g. SNARK)

➜ Very expensive to encode RSA operations (~80million gates for ar. circuits) [OWWB20]

❖ Prove correctness of N [CM99, …] & proof for non-subverted RSA groups:

➜ Proofs of correct moduli very expensive

❖ More elaborate approaches

➜ See the paper for discussion…

Possible Approaches

N: arbitrary chosen by $\mathscr{P}$

$y = g^x \pmod N$

❖ Σ-protocols

➜ λ repetitions (xλ efficiency overhead) [BCK10, TW12]

This Work: avoid repetitions

❖ General purpose NIZK (e.g. SNARK)

➜ Very expensive to encode RSA operations (~80million gates for ar. circuits) [OWWB20]

❖ Prove correctness of N [CM99, …] & proof for non-subverted RSA groups:

➜ Proofs of correct moduli very expensive

❖ More elaborate approaches

➜ See the paper for discussion…

Sigma-Protocols over HoGs (and pitfails)

N: arbitrary chosen by $\mathscr{P}$

$y = g^x \pmod N$

$\mathscr{P}(N, h, \mathrm{ct}, m)$

$r \leftarrow \$$

$a = g^r$ →

← c

$s = r + cm$ →

$\mathscr{V}(N, h, y)$

$c \leftarrow \mathscr{C}$

$g^s =_? a \cdot y^c \pmod N$

Sigma-Protocols over HoGs (and pitfails)

N: arbitrary chosen by $\mathscr{P}$ $\qquad$ $y = g^x \pmod{N}$

$\mathscr{P}(N, h, \mathrm{ct}, m)$ $\qquad\qquad$ $\mathscr{V}(N, h, y)$

$r \leftarrow \$$

$\xrightarrow{a = g^r}$

$\xleftarrow{c}$

$\xrightarrow{s = r + cm}$

$c \leftarrow \mathscr{C}$

$g^s =_? a \cdot y^c \pmod{N}$

Non-interactive via Fiat-Shamir

Sigma-Protocols over HoGs (and pitfalls)

N: arbitrary chosen by $\mathscr{P}$ $\qquad$ $y = g^x \pmod{N}$

(Knowledge) Soundness. Rewind to obtain 2 accepting transcripts on a:

$$a = g^r \longrightarrow$$

$$\xleftarrow{\quad c \quad}$$

$$s = r + cm \longrightarrow$$

$$g^s =_? a \cdot y^c \pmod{N}$$

Sigma-Protocols over HoGs (and pitfalls)

N: arbitrary chosen by $\mathscr{P}$ $\qquad$ $y = g^x \pmod N$

(Knowledge) Soundness. Rewind to obtain 2 accepting transcripts on a:

$a = g^r$ $\longrightarrow$

c $\longleftarrow$

$s = r + cm$ $\longrightarrow$

$g^s =_? a \cdot y^c \pmod N$

Rewind

$a = g^r$ $\longrightarrow$

c' $\longleftarrow$

$s = r + c'm$ $\longrightarrow$

$g^{s'} =_? a \cdot y^{c'} \pmod N$

Sigma-Protocols over HoGs (and pitfalls)

N: arbitrary chosen by $\mathscr{P}$ $y = g^x \pmod N$

(Knowledge) Soundness. Rewind to obtain 2 accepting transcripts on a:

$a = g^r$ $\longrightarrow$

c $\longleftarrow$

$s = r + cm$ $\longrightarrow$

$g^s =_? a \cdot y^c \pmod N$

Rewind

$a = g^r$ $\longrightarrow$

c' $\longleftarrow$

$s = r + c'm$ $\longrightarrow$

$g^{s'} =_? a \cdot y^{c'} \pmod N$

$$g^{s-s'} = y^{c-c'} \pmod N$$

Sigma-Protocols over HoGs (and pitfails)

N: arbitrary chosen by $\mathscr{P}$ $\qquad$ $y = g^x \pmod N$

(Knowledge) Soundness. Rewind to obtain 2 accepting transcripts on a:

$a = g^r$ $\longrightarrow$	Rewind	$a = g^r$ $\longrightarrow$
$\longleftarrow$ c		$\longleftarrow$ c'
$s = r + cm$ $\longrightarrow$		$s = r + c'm$ $\longrightarrow$
$g^s =_? a \cdot y^c \pmod N$		$g^{s'} =_? a \cdot y^{c'} \pmod N$

$$g^{s-s'} = y^{c-c'} \pmod N$$

[BCK10, TW12]

Cannot divide with $c - c'$ in the exponent: $\phi(N^2)$ is secret

➜ Unable to extract unless $c \in \mathscr{C} = \{0,1\}$ ➜ soundess error = 1/2 ➜ requires λ repetitions

Sigma-Protocols over HoGs (and pitfails)

N: arbitrary chosen by $\mathscr{P}$ $\quad$ $y = g^x \pmod N$

(Knowledge) Soundness. Rewind to obtain 2 accepting transcripts on a:

$a = g^r$ $\longrightarrow$

c $\longleftarrow$

$s = r + cm$ $\longrightarrow$

$g^s =_? a \cdot y^c \pmod N$

Rewind

$a = g^r$ $\longrightarrow$

c' $\longleftarrow$

$s = r + c'm$ $\longrightarrow$

$g^{s'} =_? a \cdot y^{c'} \pmod N$

$$g^{s-s'} = y^{c-c'} \pmod N$$

[FO97, DF01] circumvents it but taking an assumption over $\mathbb{G}$

➜ not applicable for arbitrary N

[BCK10, TW12]

Cannot divide with $c - c'$ in the exponent: $\phi(N^2)$ is secret

➜ Unable to extract unless $c \in \mathscr{C} = \{0,1\}$ ➜ soundess error = 1/2 ➜ requires λ repetitions

Our Protocol (1): A new extraction approach

$\beta_i := s^{(i)} - s^{(1)}$ $\delta_i := c^{(i)} - c^{(1)}$

Assume M accepting transcripts on a we get:

$$\left.\begin{array}{l} 1.\ \{a, c^{(1)}, s^{(1)}\} : g^{s^{(1)}} = a \cdot y^{c^{(1)}} \\ 2.\ \{a, c^{(2)}, s^{(2)}\} : g^{s^{(2)}} = a \cdot y^{c^{(2)}} \\ \quad\vdots \\ M.\ \{a, c^{(M)}, s^{(M)}\} : g^{s^{(M)}} = a \cdot y^{c^{(M)}} \end{array}\right\} \Rightarrow \left.\begin{array}{c} g^{\beta_2} = y^{\delta_2} \\ g^{\beta_3} = y^{\delta_3} \\ \vdots \\ g^{\beta_M} = y^{\delta_M} \end{array}\right\} \Rightarrow$$

Our Protocol (1): A new extraction approach

$\beta_i := s^{(i)} - s^{(1)}$ $\delta_i := c^{(i)} - c^{(1)}$

Assume M accepting transcripts on a we get:

$$\left.\begin{array}{l} 1.\ \{a, c^{(1)}, s^{(1)}\} : g^{s^{(1)}} = a \cdot y^{c^{(1)}} \\ 2.\ \{a, c^{(2)}, s^{(2)}\} : g^{s^{(2)}} = a \cdot y^{c^{(2)}} \\ \vdots \\ M.\ \{a, c^{(M)}, s^{(M)}\} : g^{s^{(M)}} = a \cdot y^{c^{(M)}} \end{array}\right\} \Rightarrow \left.\begin{array}{c} g^{\beta_2} = y^{\delta_2} \\ g^{\beta_3} = y^{\delta_3} \\ \vdots \\ g^{\beta_M} = y^{\delta_M} \end{array}\right\} \Rightarrow$$

Bezout's theorem (Informal):

If $\gcd(\delta_2, \ldots, \delta_M) = 1$ then there exist $k_2, \ldots, k_M$ such that $k_2\delta_2 + \ldots + k_M\delta_M = 1$ (over $\mathbb{Z}$)

Our Protocol (1): A new extraction approach

$\beta_i := s^{(i)} - s^{(1)}$ $\delta_i := c^{(i)} - c^{(1)}$

Assume M accepting transcripts on a we get:

$$\left.\begin{array}{l} 1.\ \{a, c^{(1)}, s^{(1)}\} : g^{s^{(1)}} = a \cdot y^{c^{(1)}} \\ 2.\ \{a, c^{(2)}, s^{(2)}\} : g^{s^{(2)}} = a \cdot y^{c^{(2)}} \\ \vdots \\ M.\ \{a, c^{(M)}, s^{(M)}\} : g^{s^{(M)}} = a \cdot y^{c^{(M)}} \end{array}\right\} \Rightarrow \left.\begin{array}{c} g^{\beta_2} = y^{\delta_2} \\ g^{\beta_3} = y^{\delta_3} \\ \vdots \\ g^{\beta_M} = y^{\delta_M} \end{array}\right\} \Rightarrow g^{\sum_{i=2}^{M} k_i\beta_i} = y^{\sum_{i=2}^{M} k_i\delta_i} = y \pmod{N}$$

Bezout's theorem (Informal):

If $\gcd(\delta_2, \ldots, \delta_M) = 1$ then there exist $k_2, \ldots, k_M$ such that $k_2\delta_2 + \ldots + k_M\delta_M = 1$ (over $\mathbb{Z}$)

Our Protocol (1): A new extraction approach

$\beta_i := s^{(i)} - s^{(1)}$

$\delta_i := c^{(i)} - c^{(1)}$

Assume M accepting transcripts on a we get:

1. $\{a, c^{(1)}, s^{(1)}\} : g^{s^{(1)}} = a \cdot y^{c^{(1)}}$
2. $\{a, c^{(2)}, s^{(2)}\} : g^{s^{(2)}} = a \cdot y^{c^{(2)}}$

$\vdots$

M. $\{a, c^{(M)}, s^{(M)}\} : g^{s^{(M)}} = a \cdot y^{c^{(M)}}$

$$\Rightarrow \quad \begin{matrix} g^{\beta_2} = y^{\delta_2} \\ g^{\beta_3} = y^{\delta_3} \\ \vdots \\ g^{\beta_M} = y^{\delta_M} \end{matrix} \quad \Rightarrow \quad g^{\sum_{i=2}^{M} k_i \beta_i} = y^{\sum_{i=2}^{M} k_i \delta_i} = y \pmod N$$

Bezout's theorem (Informal):

If $\gcd(\delta_2, \ldots, \delta_M) = 1$ then there exist $k_2, \ldots, k_M$ such that $k_2\delta_2 + \ldots + k_M\delta_M = 1$ (over $\mathbb{Z}$)

How can we guarantee that $\gcd(\delta_2, \ldots, \delta_M) = 1$?

Our Protocol (2): Our core technical Lemma

How can we guarantee that $\gcd(\delta_2, \ldots, \delta_M) = 1$?

Our Protocol (2): Our core technical Lemma

How can we guarantee that $\gcd(\delta_2, \ldots, \delta_M) = 1$?

Caveat: Prover can choose the 'type' of c to answer so that **never** $\gcd\left((c^{(i)} - c^{(1)})_{i=2}^{M}\right) = 1$ (e.g. only even c)

Our Protocol (2): Our core technical Lemma

How can we guarantee that $\boxed{\gcd(\delta_2, \ldots, \delta_M) = 1}$?

Caveat: Prover can choose the 'type' of c to answer so that **never** $\gcd\left((c^{(i)} - c^{(1)})_{i=2}^{M}\right) = 1$ (e.g. only even c)

Our approach: **Partially** hide c from the prover ➜ $\boxed{c = \langle \vec{d}, \vec{b} \rangle}$ where $\vec{d}$ hidden, $\vec{b}$ sampled during the protocol

Our Protocol (2): Our core technical Lemma

How can we guarantee that $\gcd(\delta_2, \ldots, \delta_M) = 1$?

Caveat: Prover can choose the 'type' of c to answer so that **never** $\gcd\left((c^{(i)} - c^{(1)})_{i=2}^{M}\right) = 1$ (e.g. only even c)

Our approach: **Partially** hide c from the prover ➜ $c = \langle \vec{d}, \vec{b} \rangle$ where $\vec{d}$ hidden, $\vec{b}$ sampled during the protocol

Our Information-Theoretical Lemma (Informal):

Let $\vec{d} = (d_1, \ldots, d_\lambda) \in \left(\{0,1\}^\lambda\right)^\lambda$ uniformly random, $\vec{b} = (b_1, \ldots, b_n) \in \left(\{0,1\}\right)^\lambda$ and $c = \langle \vec{d}, \vec{b} \rangle$

then for any distribution of $\vec{b}$ one can obtain $M = \text{poly}(\lambda)$ transcripts such that:

$$Pr[\gcd\left((c^{(i)} - c^{(1)})_{i=2}^{M}\right) = 1] = 1 - \text{negl}(\lambda)$$

Our Protocol (3): Bootstraping via DV

Our approach:

- **Partially** hide c from the prover: $c = \langle \vec{d}, \vec{b} \rangle$ where $\vec{d}$ hidden, $\vec{b}$ sampled during the protocol
- Then Core-Lemma + gcd-extraction technique ➜ Knowledge-Sound Protocol

Our Protocol (3): Bootstraping via DV

Our approach:

- **Partially** hide c from the prover: $c = \langle \vec{d}, \vec{b} \rangle$ where $\vec{d}$ hidden, $\vec{b}$ sampled during the protocol
- Then Core-Lemma + gcd-extraction technique ➜ Knowledge-Sound Protocol

Question: How to hide $\vec{d} = (d_1, \ldots, d_\lambda)$ from $\mathscr{P}$?

➜ $\mathscr{V}$ encrypts it ➜ DV-model

Performance-Extensions-Limitations

More on the paper:

- ★ Range proofs over Subverted RSA groups
- ★ Malicious and reusable DV keys

Implementation and Performance:

- ★ Paillier Range proof (with malicious-verifier security):

  $T(\mathscr{P})$ = 192ms, $T(\mathscr{V})$ = 125ms, $|\pi|$ = 11.05KB

Limitations:

- ❖ Designated-Verifier model
- ❖ Relatively expensive DV KeyGen for malicious zk
- ❖ Polynomial-reusability of DV keys (if verification oracles queries are assumed)

Conclusions and summary

Summary:

- ★A new general extraction method for Σ-protocols.
- ★DV-AoK and range proof-protocols for subverted RSA groups.

Open questions:

- ❖Efficient Public-Verifier protocols for Subverted RSA groups?
- ❖Apply the extraction technique to other contexts (e.g. lattice-based zk-proofs)?

Thank you!

Full version: https://eprint.iacr.org/2023/364
Implementation: https://github.com/volhovm/rsa-zkps-impl