Efficient Modular NIZK Arguments from Shift and Product

Speaker: Bingsheng Zhang[1]

Joint work with Prastudy Fauzi[2] and Helger Lipmaa[2]

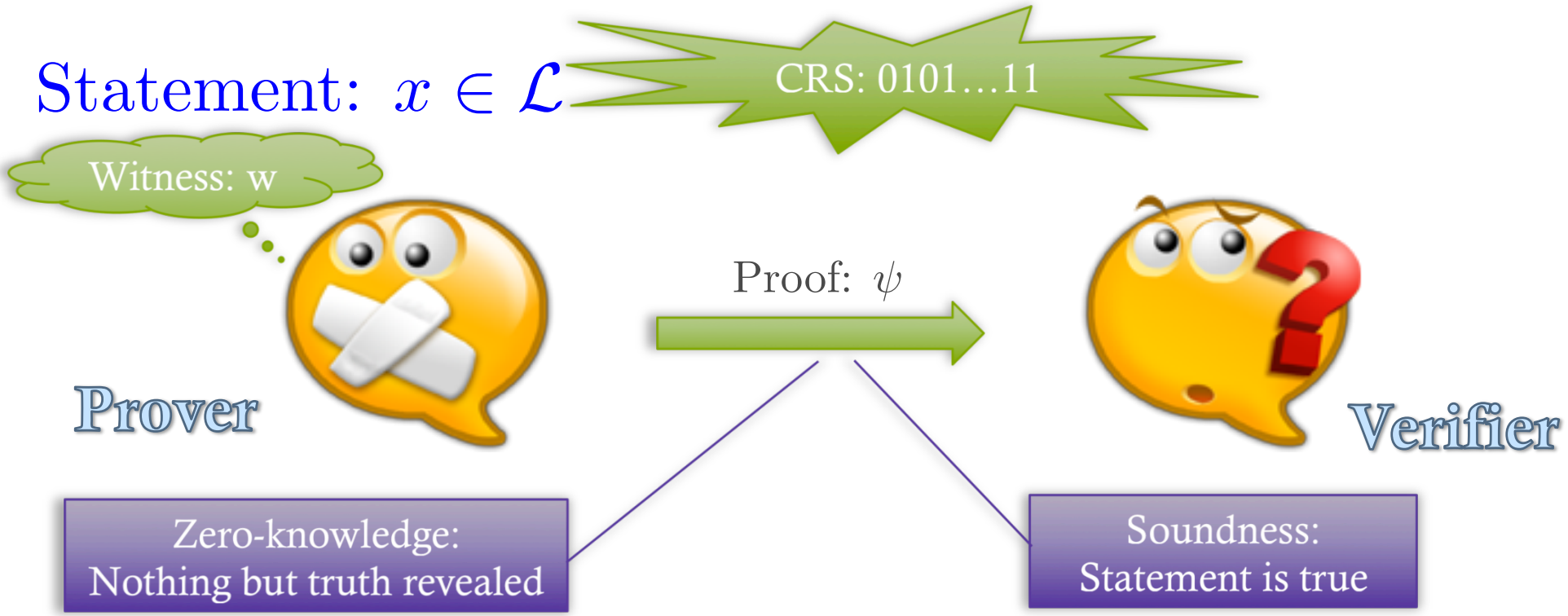
1. National and Kapodistrian University of Athens, Greece
2. University of Tartu, Estonia

CANS 2013, Paraty, Brazil

Outline

- NIZK Background
- The New Succinct Commitment Scheme
- The Improved Hadamard Product Argument
- The Shift and Rotation Arguments
- Applications
- Conclusion

Non-interactive Zero-knowledge (NIZK) Argument



Constant-Size NIZK Arguments

- Constant-size NIZK argument for CIRCUIT-SAT was first proposed by Groth [ASIACRYPT 2010].
  - CRS size is $O(n^2)$.
- Lipmaa then improved Groth's NIZK argument for CIRCUIT-SAT [TCC 2012].
  - CRS size is $O(n^{1+o(1)})$.
- Gennaro et al. proposed another constant-size NIZK argument for CIRCUIT-SAT based on quadratic span programs [EUROCRYPT 2013].
  - Prover's computation complexity is $\Theta(n \log^3 n)$.
- Lipmaa proposed a better span program based NIZK argument with prover's computation $\Theta(n \log^2 n)$ [ASIACRYPT 2013].

Modular NIZK Arguments

- Hadamard Product Arguments
  - Show that the given commitments of vectors:

$\mathbf{a} = (a_1, a_2, \ldots, a_n)$ $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ $\mathbf{c} = (c_1, c_2, \ldots, c_n)$

satisfies that $\mathbf{a} \circ \mathbf{b} = \mathbf{c} := (c_1 = a_1 b_1, \ldots, c_n = a_n b_n)$

- (Public) Permutation Arguments
  - Show that the given commitments of vectors:

$\mathbf{a} = (a_1, a_2, \ldots, a_n)$ $\mathbf{b} = (b_1, b_2, \ldots, b_n)$

satisfies that $\mathbf{b} = \pi(\mathbf{a}) := (b_1 = a_{\pi(1)}, \ldots, b_n = a_{\pi(n)})$

where π is a public permutation

Modular NIZK Arguments

- Shift Arguments (this work)
  - Show that the given commitments of vectors:
    $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ $\mathbf{b} = (b_1, b_2, \ldots, b_n)$
    satisfies that $(a_1, \ldots, a_n) = (b_{\xi+1}, \ldots, b_n, 0, \ldots, 0)$
- Rotation Arguments (this work)
  - Show that the given commitments of vectors:
    $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ $\mathbf{b} = (b_1, b_2, \ldots, b_n)$
    satisfies that $(a_1, \ldots, a_n) = (b_{\xi+1}, \ldots, b_n, b_1, \ldots, b_\xi)$

Comparison of NIZK Arguments

Scheme	|CRS|	|Argument|	Prover's computation	Verifier's computation
[Gro10]	$\Theta(n^2)$	$\Theta(1)$	$\Theta(n^2)\ exp$	$\Theta(n)\ mul+$ $\Theta(1)\ pairing$
[Lip12]	$\Theta(n^{1+o(1)})$	$\Theta(1)$	$\Theta(n^2)\ add+$ $\Theta(n^{1+o(1)})\ exp$	$\Theta(n)\ exp+$ $\Theta(1)\ pairing$
This work	$\Theta(n^{1+o(1)})$	$\Theta(1)$	$\Theta(n^{1+o(1)}\log n)\ mul$	$\Theta(n)\ mul+$ $\Theta(1)\ pairing$

Power Knowledge of Exponent Assumption

- Gentry and Wichs showed that succinct NIZK arguments cannot be based on falsifiable assumptions [STOC 2011].
- Knowledge of Exponent Assumption [Dam91]
  - Given $(g, h := g^s)$, if $\mathcal{A}$ outputs (C, D) such that $D = C^s$ then there exists an extractor $\mathcal{X}$ that can access the random tape of $\mathcal{A}$ and output (c, d) such that $C = g^c, D = h^d$.
- Power Knowledge of Exponent Assumption
  - Given $(g_i := g^{\sigma^i}, h_i := g^{s\sigma^i})_{i\in[n]}$, if $\mathcal{A}$ outputs (C, D) s.t. $D = C^s$ then there exists an extractor $\mathcal{X}$ that can access the random tape of $\mathcal{A}$ and output $(c_i, d_i)_{i\in[n]}$ s.t. $C = \prod_{i=1}^{n} g_i^{c_i}, D = \prod_{i=1}^{n} h_i^{d_i}$.

The New Succinct Vector Commitment Scheme

- System parameters: $\Lambda = \{\lambda_1, \ldots, \lambda_n\}$ and $v > \max_i \lambda_i$
- Key generation: set $(g_{\lambda_i}, \hat{g}_{\lambda_i}) \leftarrow (g, g^{\hat{\alpha}})^{\sigma^{\lambda_i}}$ and $(h, \hat{h}) \leftarrow (g, g^{\hat{\alpha}})^{\sigma^{v}}$
  - Return $ck := ((g_{\lambda_i}, \hat{g}_{\lambda_i})_{i \in [n]}, h, \hat{h})$ and $td := \sigma$
- Commit $\mathbf{a} = (a_1, \ldots, a_n)$: pick $r \leftarrow \mathbb{Z}_p$
  - Return $(c, \hat{c}) := (h, \hat{h})^r \cdot \prod_{i=1}^{n} (g_{\lambda_i}, \hat{g}_{\lambda_i})^{a_i}$
- Trapdoor commit: pick $r \leftarrow \mathbb{Z}_p$; return $(c, \hat{c}) := (h, \hat{h})^r$
- Trapdoor open to $\mathbf{a} = (a_1, \ldots, a_n)$: set $r_{td} \leftarrow r - \sum_{i=1}^{n} a_i \sigma^{\lambda_i - v}$
  - Return $(\mathbf{a}, r_{td})$

The Improved Hadamard Product Argument

- Main idea:
  - Let $A := Com(\mathbf{a}; r_a) = g_1^{r_a \sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i}}$

    $B_2 := Com(\mathbf{b}; r_b) = g_2^{r_b \sigma^v + \sum_{i=1}^n b_i \sigma^{\lambda_i}}$

    $C := Com(\mathbf{c}; r_c) = g_1^{r_c \sigma^v + \sum_{i=1}^n c_i \sigma^{\lambda_i}}$

    $D := Com(\mathbf{1}; 0) = g_2^{\sum_{i=1}^n \sigma^{\lambda_i}}$
  - Goal: to enable $e(A, B_2)/e(C, D) = e(g_1, \psi)$
  - From left side we have: $\log_{e(g_1, g_2)}(e(A, B_2)/e(C, D)) =$

$$(r_a \sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i})(r_b \sigma^v + \sum_{i=1}^n b_i \sigma^{\lambda_i}) - (r_c \sigma^v + \sum_{i=1}^n c_i \sigma^{\lambda_i})(\sum_{i=1}^n \sigma^{\lambda_i})$$

  - So the CRS is designed to allow the prover to compute all the monomials except the ones associated with $a_i b_i - c_i$.

The Improved Hadamard Product Argument

- Speed up the prover's computation:
  - FFT-based polynomial multiplication techniques.
  - Pippenger's multi-exponentiation algorithms.

Let $u(x)$ and $v(x)$ each has n monomials.
Computing $u(x) \cdot v(x)$ is much faster than $O(n^2)$ operations.

Computing $\prod_{i=1}^{n} g_i^{x_i}$ is much faster than n exp.

The Shift-by- ξ Argument

- Main idea:
  - Let $A := Com(\mathbf{a}; r_a) = g_1^{r_a \sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i}}$

    $B := Com(\mathbf{b}; r_b) = g_1^{r_b \sigma^v + \sum_{i=1}^n b_i \sigma^{\lambda_i}}$
  - Goal: to enable $e(A, g_2^{\sigma^\xi})/e(B, g_2) = e(g_1, \psi)$
  - We have: $F(\sigma) := \log_{e(g_1,g_2)}(e(A, g_2^{\sigma^\xi})/e(B, g_2))$

    $$= r_a \sigma^{v+\xi} + \sum_{i=1}^{n} a_i \sigma^{\lambda_i + \xi} - r_b \sigma^v - \sum_{i=1}^{n} b_i \sigma^{\lambda_i}$$
  - If $(a_1, \ldots, a_n) \underset{\xi}{=} (b_{\xi+1}, \ldots, b_n, 0, \ldots, 0)$
  - then $F(\sigma) = -\sum_{i=1}^{\xi} b_i \sigma^{\lambda_i} + \sum_{i=\xi+1}^{n} b_i (\sigma^{\lambda_{i-\xi}+\xi} - \sigma^{\lambda_i}) + r_a \sigma^{v+\xi} - r_b \sigma^v$
  - So the CRS is designed to allow the prover to compute them.

The Rotation-by- ξ Argument

- Main idea:
  - Let $A := Com(\mathbf{a}; r_a) = g_1^{r_a \sigma^v + \sum_{i=1}^n a_i \sigma^{\lambda_i}}$

    $B := Com(\mathbf{b}; r_b) = g_1^{r_b \sigma^v + \sum_{i=1}^n b_i \sigma^{\lambda_i}}$
  - Goal: to enable $e(A, g_2^{\sigma^\xi})/e(B, g_2) = e(g_1, \psi)$
  - We have: $F(\sigma) := \log_{e(g_1,g_2)}(e(A, g_2^{\sigma^\xi})/e(B, g_2))$

    $$= r_a \sigma^{v+\xi} + \sum_{i=1}^{n} a_i \sigma^{\lambda_i + \xi} - r_b \sigma^v - \sum_{i=1}^{n} b_i \sigma^{\lambda_i}$$
  - If $(a_1, \ldots, a_n) = (b_{\xi+1}, \ldots, b_n, b_1, \ldots, b_\xi)$
  - then $F(\sigma) := \sum_{i=1}^{\xi} b_i(\sigma^{\lambda_{n-\xi+i}+\xi} - \sigma^{\lambda_i}) + \sum_{i=\xi+1}^{n} b_i(\sigma^{\lambda_{i-\xi}+\xi} - \sigma^{\lambda_i}) + r_a\sigma^{v+\xi} - r_b\sigma^v$
  - So the CRS is designed to allow the prover to compute them.

Applications

- Improved range argument
- Set partition argument
- Subset-sum argument
- Decision-knapsack argument

Improved range argument

- Simplified Version
  - Basic idea: show $a \in [0, 2^{\ell+1})$ by showing $a = \sum_{i=0}^{\ell} b_i 2^i, b_i \in \{0,1\}$
  - Steps:
    - 1. Commit $A = Com(a; r_a), B = Com((b_0, \ldots, b_\ell); r_b)$
    - 2. Show that $[b_0, \ldots, b_\ell] \circ [b_0, \ldots, b_\ell] = [b_0, \ldots, b_\ell]$
    - 3. Set and prove that $[b_0, \ldots, b_\ell] \circ [2^0, \ldots, 2^\ell] = [c_0, \ldots, c_\ell]$
    - 4. Set $[d_0, d_1, \ldots, d_\ell] := [\sum_{j=0}^{0} c_j, \sum_{j=0}^{1} c_j, \ldots, \sum_{j=0}^{\ell} c_j]$
    - 5. Set and prove $[e_0, e_1, \ldots, e_\ell] := [0, d_0, \ldots, d_{\ell-1}]$
    - 6. Show that $[e_0, e_1, \ldots, e_\ell] + [c_0, c_1, \ldots, c_\ell] := [d_0, d_1, \ldots, d_\ell]$
    - 7. Show that $[d_0, d_1, \ldots, d_\ell] \circ [0, 0, \ldots, 1] = [0, 0, \ldots, a]$

Set Partition Argument

- Set partition problem
  - Given $S = (s_1, \ldots, s_n), s_i \in \mathbb{Z}_p$
  - Find a set $V \subset S$ such that $\sum_{x \in V} x = \sum_{y \in S \setminus V} y$
- Argument steps: Define $b_i = 1$ for $s_i \in V$ and $b_j = -1$ for $s_j \in S \setminus V$
  - 1. Commit and show that $[b_1, \ldots, b_n] \circ [b_1, \ldots, b_n] = [1, \ldots, 1]$
  - 2. Commit and show that $[s_1, \ldots, s_n] \circ [b_1, \ldots, b_n] = [c_1, \ldots, c_n]$
  - 4. Set $[d_0, d_1, \ldots, d_n] := [\sum_{j=0}^{0} c_j, \sum_{j=0}^{1} c_j, \ldots, \sum_{j=0}^{n} c_j]$
  - 5. Set and prove $[e_0, e_1, \ldots, e_n] := [0, d_0, \ldots, d_{n-1}]$
  - 6. Show that $[e_0, e_1, \ldots, e_n] + [c_0, c_1, \ldots, c_n] = [d_0, d_1, \ldots, d_n]$
  - 7. Show that $[d_1, \ldots, d_n] \circ [0, \ldots, 0, 1] = [0, \ldots, 0]$

Subset-sum Argument

- Subset-sum problem
  - Given $S = (s_1, \ldots, s_n), s_i \in \mathbb{Z}_p$ and the target $t \in \mathbb{Z}_p$
  - Find a set $V \subset S$ such that $\sum_{x \in V} x = t$
- Argument steps: Define $b_i = 1$ for $s_i \in V$ and $b_j = 0$ for $s_j \in S \setminus V$
  - 1. Commit and show that $[b_0, \ldots, b_n] \circ [b_0, \ldots, b_n] = [b_0, \ldots, b_n]$
  - 2. Commit and show that $[s_1, \ldots, s_n] \circ [b_1, \ldots, b_n] = [c_1, \ldots, c_n]$
  - 4. Set $[d_0, d_1, \ldots, d_n] := [\sum_{j=0}^{0} c_j, \sum_{j=0}^{1} c_j, \ldots, \sum_{j=0}^{n} c_j]$
  - 5. Set and prove $[e_0, e_1, \ldots, e_n] := [0, d_0, \ldots, d_{n-1}]$
  - 6. Show that $[e_0, e_1, \ldots, e_n] + [c_0, c_1, \ldots, c_n] = [d_0, d_1, \ldots, d_n]$
  - 7. Show that $[d_1, \ldots, d_n] \circ [0, \ldots, 0, 1] = [0, \ldots, 0, t]$

Decision-knapsack Argument

- Decision-knapsack problem
  - Given a set S, integers W and B, and benefit value $\{b_i\}_{i \in S}$ and weight $\{w_i\}_{i \in S}$ of every item in S.
  - Decide whether there exists a subset $T \subseteq S$ such that
    - 1. $\sum_{i \in T} w_i \leq W$
    - 2. $\sum_{i \in T} b_i \geq B$

- Argument steps: range NIZK argument is involved. (See full version for details).

Conclusion

- We improved the Hadamard product argument
- We proposed shift and rotation arguments
- We constructed many NP-complete languages, such as set partition and subset-sum, etc.
- We believe that the presented modular NIZK arguments can be used to build many other complex NIZK arguments for some concrete languages.

Thank You!

Q & A