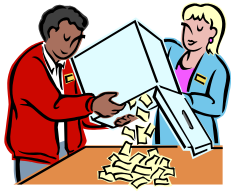
Efficient Zero-Knowledge Argument for Correctness of a Shuffle

Stephanie Bayer
University College London

Jens Groth
University College London

Motivation – e-voting

- Voting:  - Voter casts secret vote

  - Authorities reveal votes in random permuted order 

- E-voting:  - voter casts secret votes on a computer

- The votes are sent to a server who sends all votes to the central authorities
- Authorities reveal votes in random permuted order 

Background - ElGamal encryption

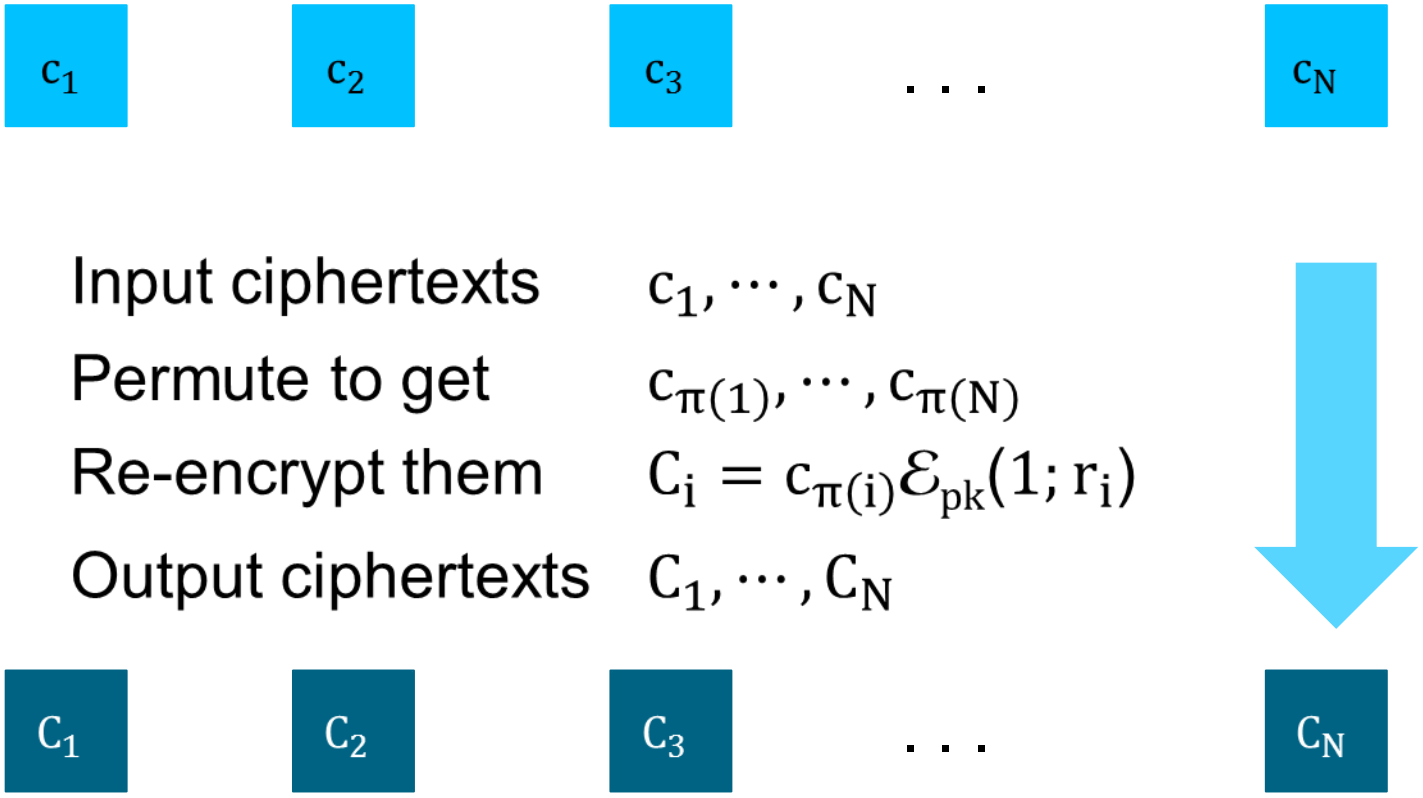
- Setup: Group $\mathcal{G}$ of prime order q with generator g
- Public key: $pk = y = g^x$
- Encryption: $\mathcal{E}_{pk}(m; r) = (g^r, y^r m)$
- Decryption: $\mathcal{D}_x(u, v) = vu^{-x}$
- Homomorphic:

$$\mathcal{E}_{pk}(m; r) \times \mathcal{E}_{pk}(M; R) = \mathcal{E}_{pk}(mM; r + R)$$

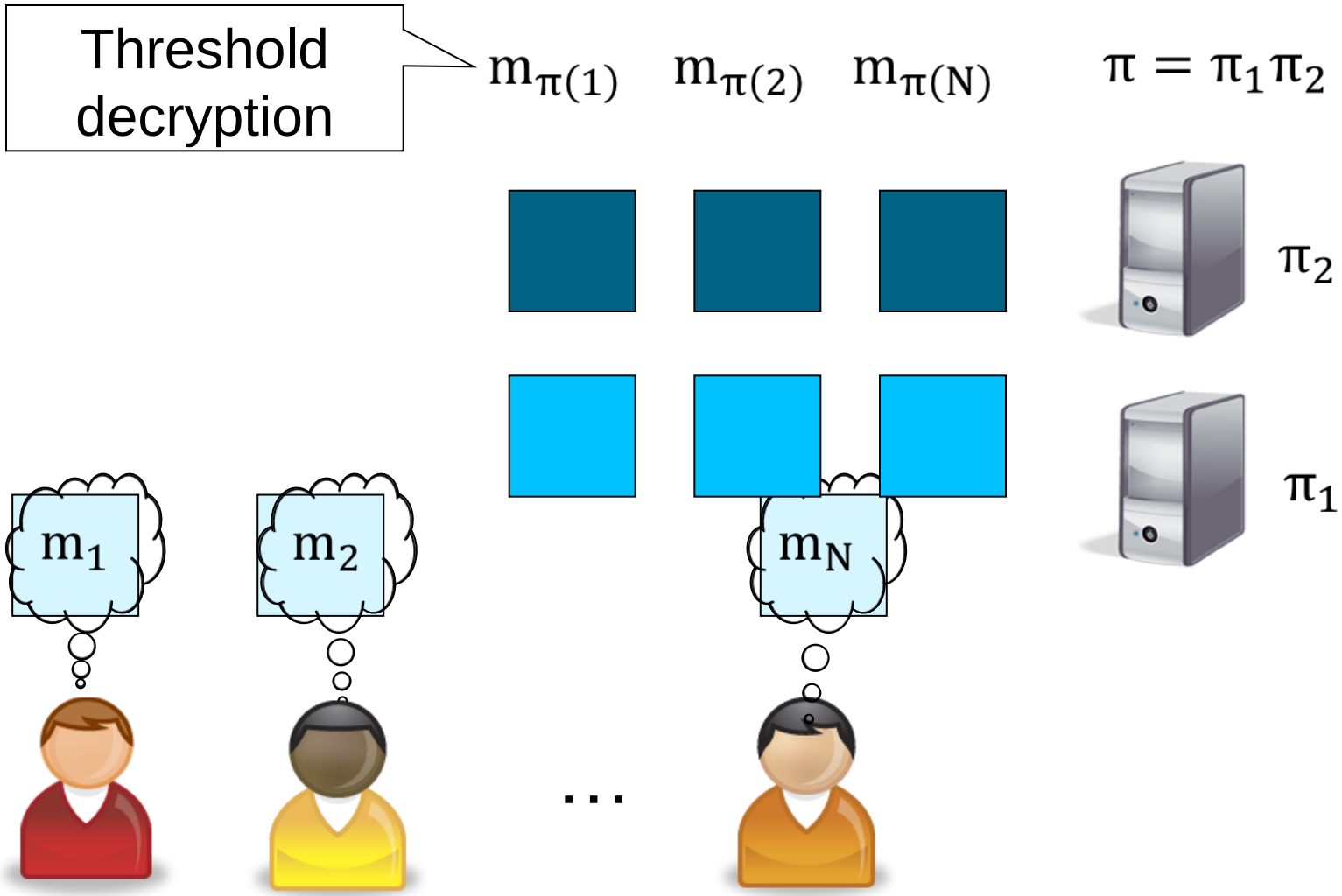
- Re-rencryption:

$$\mathcal{E}_{pk}(m; r) \times \mathcal{E}_{pk}(1; R) = \mathcal{E}_{pk}(m; r + R)$$

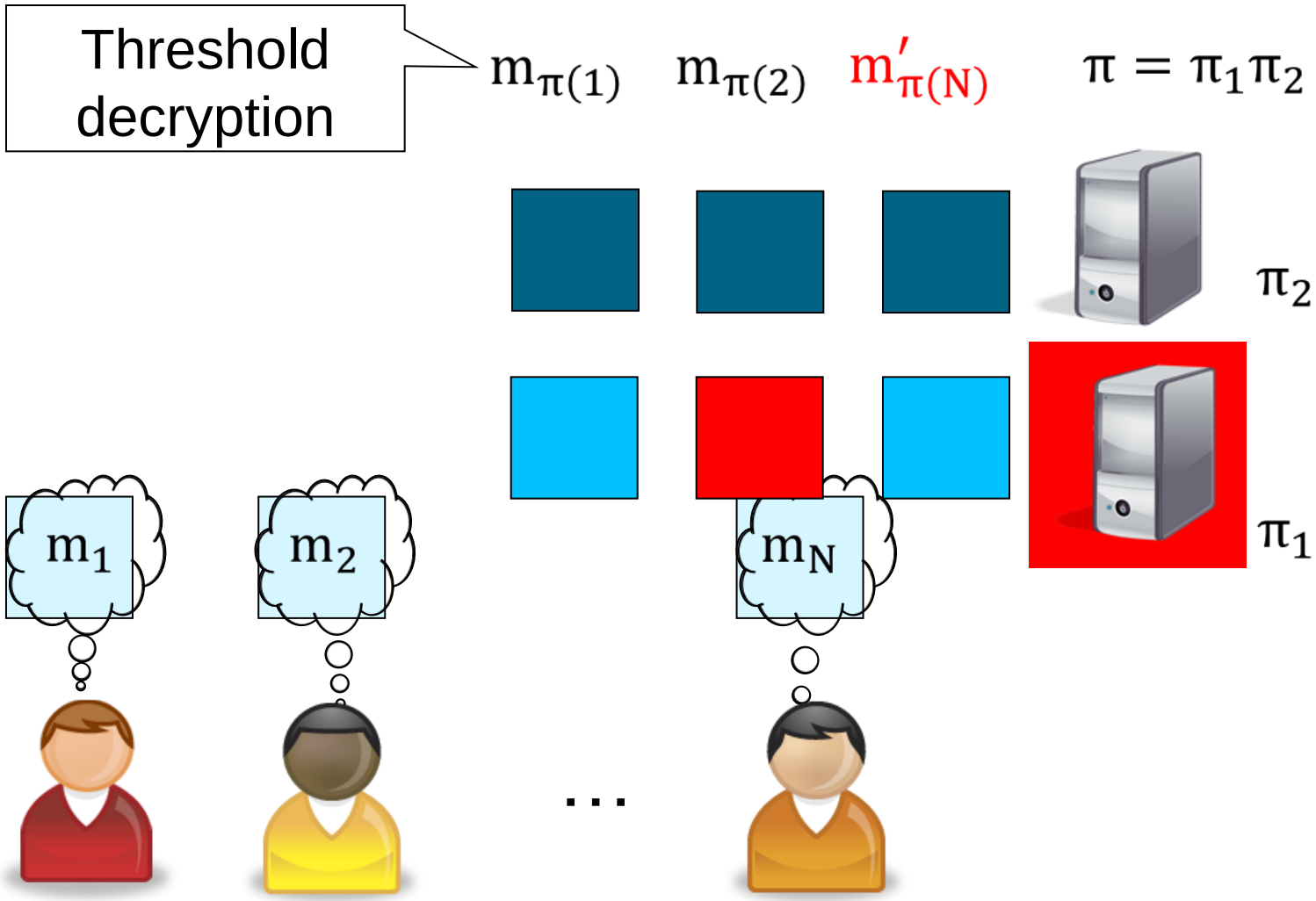
Shuffle


c_1
c_2
c_3
. . .
c_N
Input ciphertexts $c_1, \cdots, c_N$
Permute to get $c_{\pi(1)}, \cdots, c_{\pi(N)}$
Re-encrypt them $C_i = c_{\pi(i)}\mathcal{E}_{pk}(1; r_i)$
Output ciphertexts $C_1, \cdots, C_N$
C_1
C_2
C_3
. . .
C_N

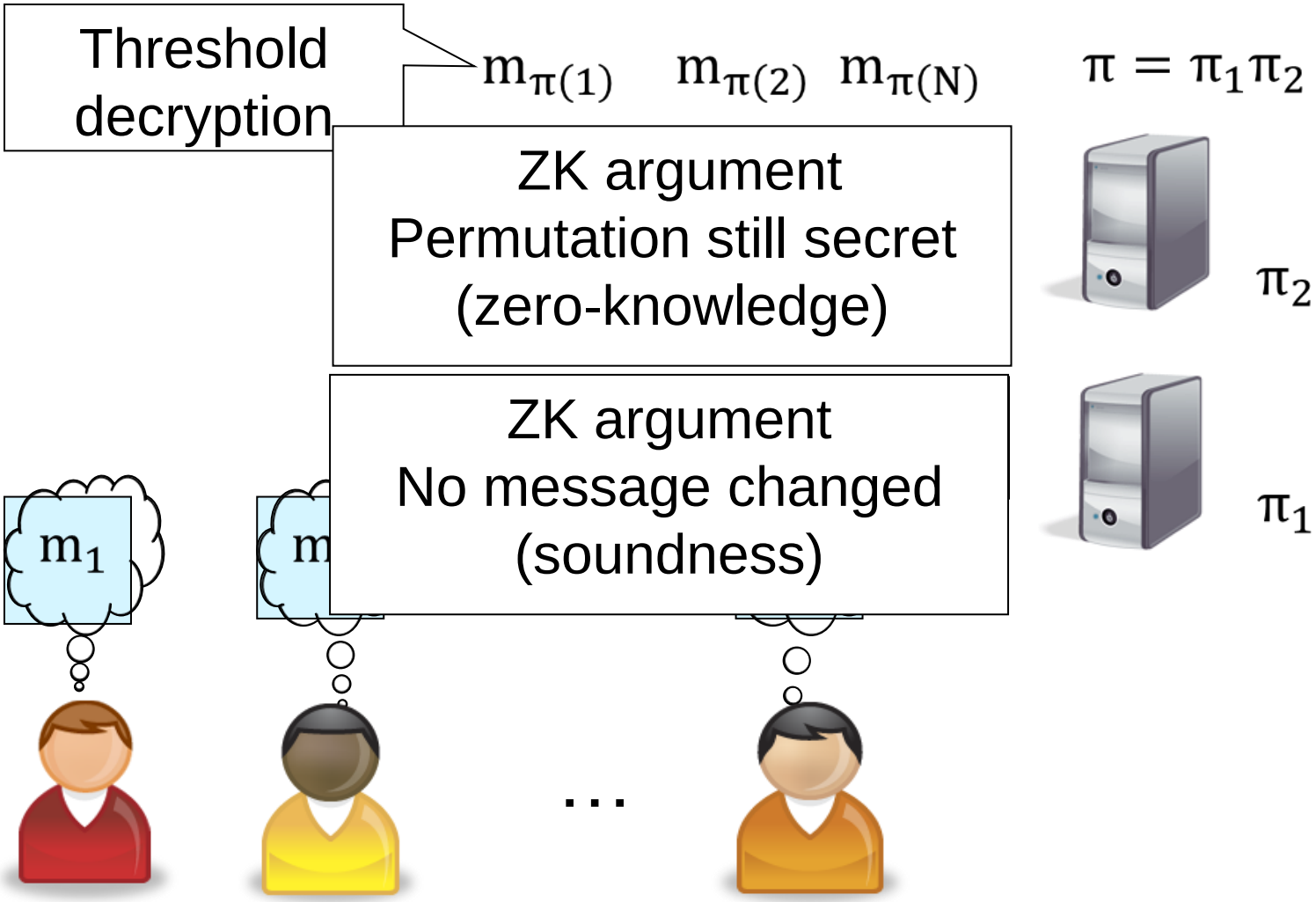
Mix-net:


Threshold decryption
$m_{\pi(1)}$ $m_{\pi(2)}$ $m_{\pi(N)}$
$\pi = \pi_1\pi_2$
π_2
π_1
m_1
m_2
m_N
...

Problem: Corrupt mix-server

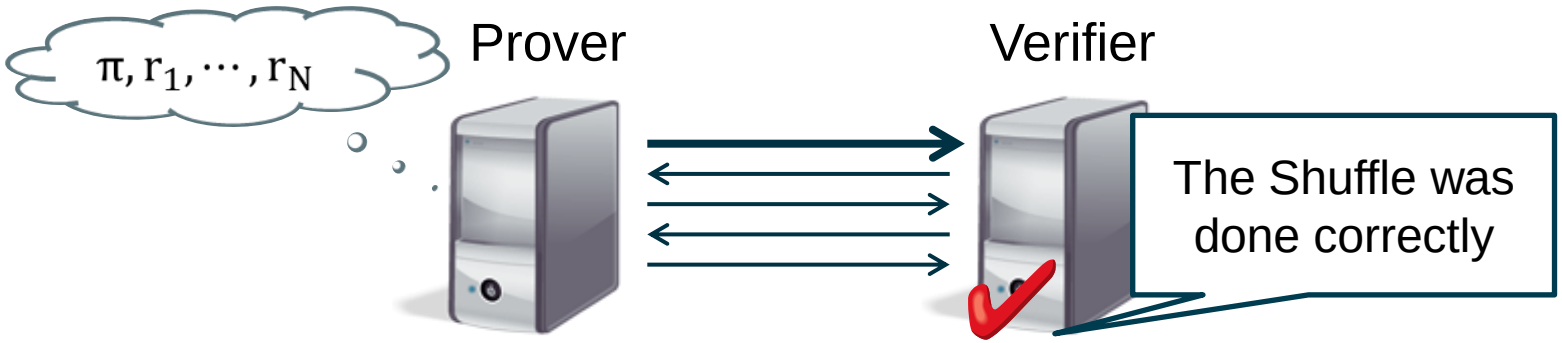


Solution: Zero-knowledge argument



Zero-Knowledge Argument

Statement: $(\mathrm{pk}, c_1, \cdots, c_N, C_1, \cdots, C_N)$



Requested Properties:

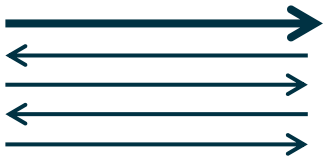
- Soundness: The Verifier reject with overwhelming probability if the Prover tries to cheat
- Zero-Knowledge: Nothing but the truth is revealed; permutation is secret
- Efficient: Small computation and small communication complexity

Public coin honest verifier zero-knowledge

Setup: $(\mathcal{G}, q, g)$ and common reference string

Statement: $(pk, c_1, \cdots, c_N, C_1, \cdots, C_N)$

Honest verifier zero-knowledge
Nothing but truth revealed;
permutation secret



Public coin:

Random challenges from $\mathbb{Z}_q$

Can convert to standard zero-knowledge argument

Our contribution

- 9-move public coin honest verifier zero-knowledge argument for correctness of shuffle in common reference string model
- For $\mathrm{N} = \mathrm{m} \times \mathrm{n}$ ciphertexts

  Communication: $\mathcal{O}(\mathrm{m} + \mathrm{n})$k bits

  Prover's computation: $\mathcal{O}(\log(\mathrm{m})\ \mathrm{N})$ expos

  Verifier's computation: $\mathcal{O}(\mathrm{N})$ expos

Comparison of ElGamal shuffles ($N = mn$)

$\|p\| = 1024$ $\|q\| = 160$	Rounds	Prover in expos	Verifier in expos	Size in kbits
Furukawa-Sako 01	3	$8N$	$10N$	$5.3N$
FMMOS 02	5	$9N$	$10N$	$5.3N$
Furukawa 05 (GL07)	3	$7N$	$8N$	$1.5N$
Terelius-Wikström 10	5	$9N$	$11N$	$3.7N$
Neff 01,04	7	$8N$	$12N$	$7.7N$
Groth 03,10	7	$6N$	$6N$	$0.6N$
Groth-Ishai 08	7	$3mN$	$4N$	$3m^2 + 0.5n$
Bayer-Groth 11	9	$2\log(m)N$	$4N$	$11m + 0.8n$
Bayer-Groth 11	$\log(m)$	$O(N)$	$4N$	$11m + 0.8n$

Commitments

- Commit to a column vector $(a_1, \cdots, a_n)^T \in \mathbb{Z}_q{}^n$ as $A=com_{ck}((a_1, \cdots, a_n)^T; r)$
  - Length reducing
  - Computational binding
  - Perfectly hiding
  - Homomorphic

    $com_{ck}(\vec{a};r)*com_{ck}(\vec{b}; s) = com_{ck}(\vec{a} + \vec{b}; r + s)$

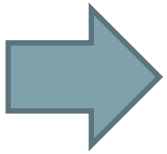
- Pedersen Commitment:

  $com_{ck}(\vec{a}; r) = h^r \prod_{i=1}^{n} g_i{}^{a_i}$

Techniques - Sublinear cost

- Length reducing commitments
- Batch verification
- Structured Vandermonde challenges

$$1 \; x \; x^2 \; \cdots \; x^N$$



Sublinear communication cost

Shuffle argument

- Given public keys pk and ck
- Given ciphertexts $c_1, \cdots, c_N$ and $C_1, \cdots, C_N$
- Prover knows permutation π and randomizers $r_1, \cdots, r_N$ and wants to convince the verifier

$$C_1 = c_{\pi(1)}\mathcal{E}_{\mathrm{pk}}(1; r_1) \quad \cdots \quad C_N = c_{\pi(N)}\mathcal{E}_{\mathrm{pk}}(1; r_N)$$

Shuffle argument

1. The prover commits to a permutation π by committing to $\pi(1), \cdots, \pi(N)$
   - Verifier sends challenge $x \in \mathbb{Z}_q$
2. The prover commits to $x^{\pi(1)}, \cdots, x^{\pi(N)}$
3. The prover gives an argument that both commitments are constructed using the same permutation
4. The prover demonstrates that the input ciphertexts are permuted using the same permutation and knowledge of the randomizers used in the re-encryption.

Shuffle argument

- Prover commits to π as

  $A=com_{ck}(\pi(1),\cdots,\pi(N);r)=com_{ck}(a_1,\cdots,a_N;r)$

  and after receiving challenge $x \in \mathbb{Z}_q$ to

  $b=com_{ck}(x^{\pi(1)},\cdots,x^{\pi(N)};s)=com_{ck}(b_1,\cdots,b_N;s)$

- Prover gives product argument for A, B such that

  $\prod_{i=1}^{N}(a_i y + b_i - z) = \prod_{i=1}^{N}(iy + x^i - z)$

- Multi-exponentiation argument such that

  $\prod_{i=1}^{N} C_i^{b_i} = \prod_{i=1}^{N} c_i^{x^i}$

Inexpensive
See full paper

Both polynomials are equal, only the roots are permuted

Expensive
Will sketch idea

- Sketch idea focusing on soundness
- Ignore ZK (easy and cheap to add)
- Will also for simplicity assume randomness $\rho = 0$

Notation

- B contains commitments $B_1, \cdots, B_m$ where

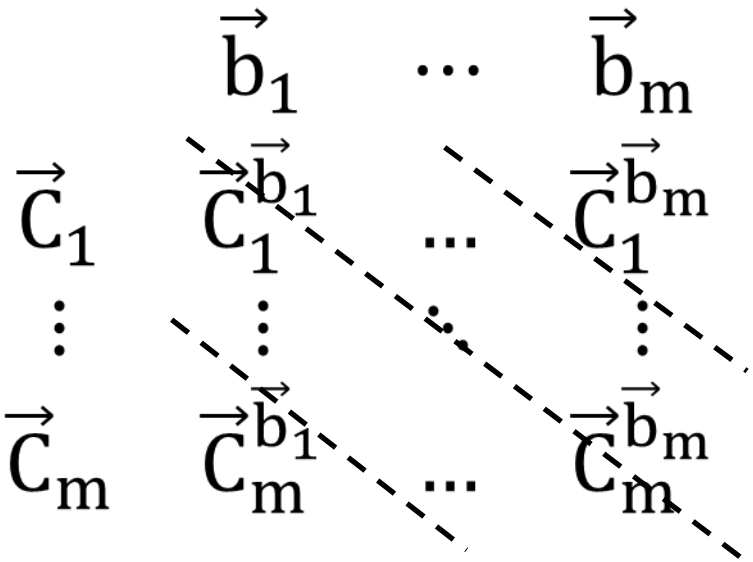
$$B_1 = \mathrm{com}_{ck}\begin{pmatrix} x^{\pi(1)} \\ \vdots \\ x^{\pi(n)} \end{pmatrix}; s_1 = \mathrm{com}_{ck}(\vec{b}_1; s_1), \cdots, B_m = \mathrm{com}_{ck}(\vec{b}_m; s_m)$$

- Arrange ciphertexts $C_1, \cdots, C_N$ in $m \times n$ matrix

$$\begin{bmatrix} \vec{C}_1 \\ \vdots \\ \vec{C}_m \end{bmatrix} = \begin{bmatrix} C_1 & \cdots & C_n \\ \vdots & \ddots & \vdots \\ C_{N-n+1} & \cdots & C_N \end{bmatrix}$$

- Define inner product $\vec{C}_i^{\vec{b}_j} = \prod_{k=1}^{n} C_{il}^{b_{jk}}$ to simplify the statement as $\prod_{i=1}^{N} C_i^{b_i} = \prod_{i=1}^{m} \vec{C}_i^{\vec{b}_i} = C$

Multi-exponentiation argument idea



$$E_{-k} = \prod_{i=1-k}^{m} \vec{C}_{i-k}^{\vec{b}_i}$$

$$E_k = \prod_{i=1}^{m-k} \vec{C}_{i+k}^{\vec{b}_i} \qquad C = \prod_{i=1}^{m} \vec{C}_i^{\vec{b}_i}$$

Multi-exponentiation argument

Communicaton:
$\mathcal{O}(m+n)$ elements

Verifier computation:
$4N + \mathcal{O}(m+n)$ expos

$\cdots, E_{-1}, E_1, \cdots E_m$

ge $y \in \mathbb{Z}_q$

2m ciphertexts

$$\prod_{j=1}^{m} B_j^{y^{-j}} = com_{ck}\left(\sum_{j=1}^{m} y^{-j}\vec{b}_j ; \sum_{j=1}^{m} y^{-j} s_j\right)$$

n elements in $\mathbb{Z}_q$

N ciphertext expos

to $\vec{b} = \sum_{j=1}^{m} y^{-j}\vec{b}_j$

N ciphertext expos

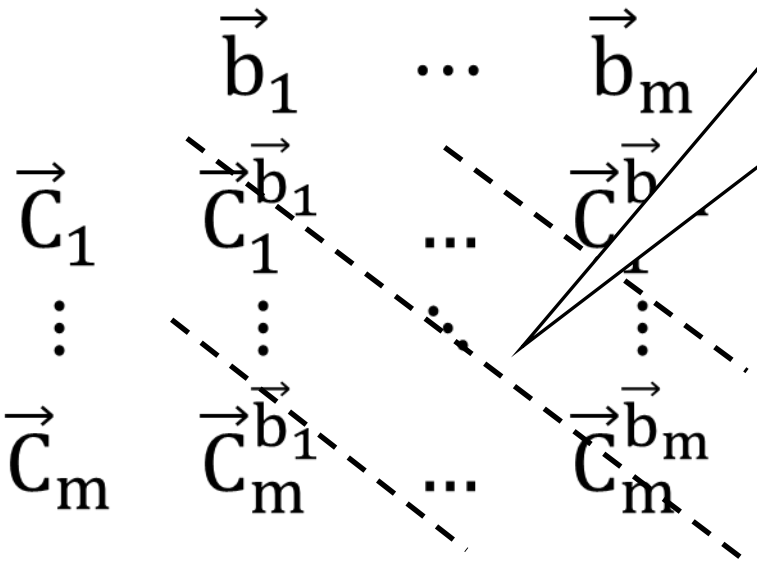
les $\vec{C} = \prod_{i=1}^{m} C_i^{y^{-i}}$ and checks

2m ciphertext expos

$$\vec{C}^{\vec{b}} = C \prod_{k=1}^{m} E_{-k}^{\cdots} E_k$$

Prover's computation

Computing this matrix costs $m^2n = mN$ ciphertext expos



$$E_{-k} = \prod_{i=1-k}^{m} \vec{C}_{i-k}^{\vec{b}_i}$$

$$E_k = \prod_{i=1}^{m-k} \vec{C}_{i+k}^{\vec{b}_i} \qquad C = \prod_{i=1}^{m} \vec{C}_i^{\vec{b}_i}$$

Reducing the prover's computation

- Do not compute entire matrix
- Instead use techniques for multiplication of polynomials "in the exponent" of ciphertexts
- Fast Fourier Transform
  - O(N log m) exponentiations o (1) rounds
- Interaction
  - o (N) exponentiations o (log m) rounds

Implementation

- Implementation in C++ using the NTL library and the GMP library
- Different levels of optimization
  - Multi-exponentiation techniques
  - Fast Fourier Transform
  - Extra Interaction and Toom-Cook

Comparison

$N = 100,000$	Single argument	Argument Size
Verificatum	5 min	37.7 MB
Toom-Cook, $m = 64$	2 min	0.7 MB

- Runtime comparison of Verificatum (Wikström) to our shuffle argument
- MacBook Pro; CPU: 2.54 GHZ, RAM: 4GB
- $|p| = 1024$, $|q| = 160$
- $N = 100{,}000$ ciphertexts, $m = 64, n = 1563$

Thank You