Proofs of Restricted Shuffles

Björn Terelius and Douglas Wikström

KTH, Stockholm

May 3, 2010

A motivating example: Voting

Consider a voting system where each voter submit an encrypted vote.

A motivating example: Voting

Consider a voting system where each voter submit an encrypted vote.

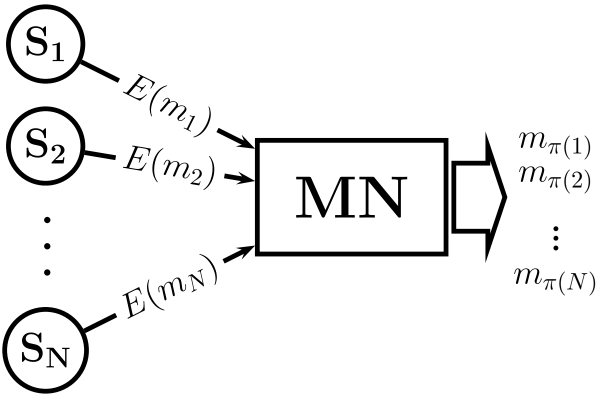
- How can we ensure that the voters remain anonymous when the votes are decrypted?

A motivating example: Voting

Consider a voting system where each voter submit an encrypted vote.

- How can we ensure that the voters remain anonymous when the votes are decrypted?
- There are two main ways to achieve this, homomorphic tallying [CGS97] and mixnets [Cha81].

Mixnets



Mixnets (2)

- How can we implement a mixnet?

Mixnets (2)

- How can we implement a mixnet?
- Chain of mixservers, each permutes and re-encrypts its list of inputs.

$$\xrightarrow{L_0} \boxed{\mathbf{T_1}} \xrightarrow{L_1} \boxed{\mathbf{T_2}} \xrightarrow{L_2} \cdots \xrightarrow{L_{k-1}} \boxed{\mathbf{T_k}} \xrightarrow{L_k}$$

Proof of a shuffle

- How can we verify that a server really permutes and re-encrypts the votes?

Proof of a shuffle

- How can we verify that a server really permutes and re-encrypts the votes?
- Let each server produce an interactive zero-knowledge proof, a *proof of a shuffle* [SK95, Nef01, FS01].

Proof of a shuffle

- How can we verify that a server really permutes and re-encrypts the votes?
- Let each server produce an interactive zero-knowledge proof, a *proof of a shuffle* [SK95, Nef01, FS01].
- Like [FS01], we will construct a proof that a commitment contains a permutation matrix.

Proof of a shuffle

- How can we verify that a server really permutes and re-encrypts the votes?
- Let each server produce an interactive zero-knowledge proof, a *proof of a shuffle* [SK95, Nef01, FS01].
- Like [FS01], we will construct a proof that a commitment contains a permutation matrix.
- One can then prove that the encrypted votes are permuted accordingly.

Test for permutation matrices

M permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

M not permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

Test for permutation matrices

M permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M\overline{x} = \begin{pmatrix} x_2 \\ x_1 \\ x_3 \end{pmatrix}$$

M not permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M\overline{x} = \begin{pmatrix} x_2 \\ 2x_1 - x_3 \\ x_3 \end{pmatrix}$$

Test for permutation matrices

M permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M\overline{x} = \begin{pmatrix} x_2 \\ x_1 \\ x_3 \end{pmatrix}$$

$$\begin{aligned} \prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle &= x_2 x_1 x_3 \\ &= x_1 x_2 x_3 \end{aligned}$$

M not permutation matrix

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$M\overline{x} = \begin{pmatrix} x_2 \\ 2x_1 - x_3 \\ x_3 \end{pmatrix}$$

$$\begin{aligned} \prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle &= x_2(2x_1 - x_3)x_3 \\ &\neq x_1 x_2 x_3 \end{aligned}$$

Test for permutation matrices

Theorem (Permutation Matrix)

Let $M = (m_{i,j})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and $\overline{x} = (x_1, \ldots, x_N)$ be a list of variables. Then M is a permutation matrix if and only if

$$\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle = \prod_{i=1}^{N} x_i \quad \textit{and} \quad M\overline{1} = \overline{1} \ .$$

Test for permutation matrices

Theorem (Permutation Matrix)

Let $M = (m_{i,j})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and $\overline{x} = (x_1, \ldots, x_N)$ be a list of variables. Then M is a permutation matrix if and only if

$$\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle = \prod_{i=1}^{N} x_i \quad \text{and} \quad M\overline{1} = \overline{1} \enspace .$$

Lemma (Schwartz-Zippel)

Let $f \in \mathbb{Z}_q[x_1, \ldots, x_N]$ be a non-zero polynomial of total degree d and let $e_1, \ldots, e_N$ be chosen randomly from $\mathbb{Z}_q$. Then

$$\Pr[f(e_1, \ldots, e_N) = 0] \leq \frac{d}{q} \enspace .$$

Recall Pedersen commitments

Let g, g_1 be randomly chosen generators in a group of prime order q. The Pedersen commitment of $m \in \mathbb{Z}_q$ is

$$\mathcal{C}(m, s) = g^s g_1^m$$

where s is chosen randomly from $\mathbb{Z}_q$.

Recall Pedersen commitments

Let g, g_1 be randomly chosen generators in a group of prime order q. The Pedersen commitment of $m \in \mathbb{Z}_q$ is

$$\mathcal{C}(m, s) = g^s g_1^m$$

where s is chosen randomly from $\mathbb{Z}_q$.

- perfectly hiding
- computationally binding
- homomorphic, $\mathcal{C}(m, s)\,\mathcal{C}(m', s') = \mathcal{C}(m + m', s + s')$
  $\mathcal{C}(m, s)^e = \mathcal{C}(em, es)$

Generalized Pedersen commitments [FS01]

Let $g, g_1, \ldots, g_N$ be randomly chosen generators in a group of prime order q. We commit to a vector $\overline{m} = (m_1, \ldots, m_N)^{\mathrm{T}}$ by

$$\mathcal{C}(\overline{m}, s) = g^s \prod_{i=1}^{N} g_i^{m_i}$$

where s is chosen randomly from $\mathbb{Z}_q$.

Generalized Pedersen commitments [FS01]

Let $g, g_1, \ldots, g_N$ be randomly chosen generators in a group of prime order q. We commit to a vector $\overline{m} = (m_1, \ldots, m_N)^{\mathrm{T}}$ by

$$\mathcal{C}\left(\overline{m}, s\right) = g^s \prod_{i=1}^{N} g_i^{m_i}$$

where s is chosen randomly from $\mathbb{Z}_q$.

- perfectly hiding
- computationally binding
- homomorphic, $\mathcal{C}\left(\overline{m}, s\right) \mathcal{C}\left(\overline{m}', s'\right) = \mathcal{C}\left(\overline{m} + \overline{m}', s + s'\right)$

$$\mathcal{C}\left(\overline{m}, s\right)^e = \mathcal{C}\left(e\overline{m}, es\right)$$

Generalized Pedersen commitments

We commit column-wise to an $N \times N$-matrix $M = (m_{i,j})$, so $a = \mathcal{C}(M, \overline{s})$ is a list of N commitments satisfying

$$\mathcal{C}(M, \overline{s})^{\overline{e}} = \mathcal{C}(M\overline{e}, \langle \overline{s}, \overline{e} \rangle)$$

where we use the convention

$$a^{\overline{e}} = \prod_{i=1}^{N} a_i^{e_i} \quad .$$

A review of sigma proofs

A sigma proof is a three-message protocol such that

1. the view of the verifier can be simulated for any given challenge

A review of sigma proofs

A sigma proof is a three-message protocol such that

1. the view of the verifier can be simulated for any given challenge
2. a witness can be computed from any pair of accepting transcripts with the same random tape and distinct challenges

Example: Proof of knowledge of discrete logarithm

$\mathcal{P}$ wants to prove knowledge of x such that $y = g^x$

1. $\mathcal{P}$ chooses r at random and sends $\alpha = g^r$
2. $\mathcal{V}$ sends a random challenge c
3. $\mathcal{P}$ responds with $d = cx + r$

$\mathcal{V}$ accepts the proof iff $y^c \alpha = g^d$

Example: Proof of knowledge of discrete logarithm

$\mathcal{P}$ wants to prove knowledge of x such that $y = g^x$

1. $\mathcal{P}$ chooses r at random and sends $\alpha = g^r$
2. $\mathcal{V}$ sends a random challenge c
3. $\mathcal{P}$ responds with $d = cx + r$

$\mathcal{V}$ accepts the proof iff $y^c\alpha = g^d$

There are similar protocols for proving any polynomial relation!

Proof of knowledge of permutation matrix

Given a matrix commitment a, $\mathcal{P}$ wants to prove knowledge of a **permutation matrix** M and randomness $\overline{s}$ such that $a = \mathcal{C}\left(M, \overline{s}\right)$.

1. $\mathcal{V}$ chooses a vector $\overline{e}$ randomly and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ uses a sigma proof to prove knowledge of t, k and a vector $\overline{e}'$ such that

$$\mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}}$$
$$\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}}$$
$$\prod_{i=1}^{N} e_i' = \prod_{i=1}^{N} e_i$$

Proof of knowledge of permutation matrix

Given a matrix commitment a, $\mathcal{P}$ wants to prove knowledge of a **permutation matrix** M and randomness $\overline{s}$ such that $a = \mathcal{C}(M, \overline{s})$.

1. $\mathcal{V}$ chooses a vector $\overline{e}$ randomly and sends it to $\mathcal{P}$.
2. $\mathcal{P}$ uses a sigma proof to prove knowledge of t, k and a vector $\overline{e}'$ such that

$$\mathcal{C}(\overline{e}', k) = a^{\overline{e}}$$
$$\mathcal{C}(\overline{1}, t) = a^{\overline{1}}$$
$$\prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i$$

$$\overline{e}' = M\overline{e}$$
$$\overline{1} = M\overline{1}$$
$$\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle = \prod_{i=1}^{N} e_i$$

Properties of the protocol

Theorem

The protocol is a honest verifier zero knowledge proof of knowledge of a permutation matrix M *such that* $a = \mathcal{C}(M, \bar{s})$*, assuming the commitment scheme is binding.*

Properties of the protocol

Theorem

The protocol is a honest verifier zero knowledge proof of knowledge of a permutation matrix M such that $a = \mathcal{C}(M, \bar{s})$, assuming the commitment scheme is binding.

- The zero-knowledge property is easy.

Properties of the protocol

Theorem

The protocol is a honest verifier zero knowledge proof of knowledge of a permutation matrix M such that $a = \mathcal{C}(M, \bar{s})$, assuming the commitment scheme is binding.

- The zero-knowledge property is easy.
- We must construct an extractor which computes a permutation matrix from accepting transcripts.

Sketch of proof

1. Run the extractor of the sigma proof N times with $\overline{e}_1, \ldots, \overline{e}_N$, each time extracting $\overline{e}'_i$ and k_i such that $\mathcal{C}\left(\overline{e}'_i, k_i\right) = a^{\overline{e}_i}$.

Sketch of proof

1. Run the extractor of the sigma proof N times with $\overline{e}_1, \ldots, \overline{e}_N$, each time extracting $\overline{e}'_i$ and k_i such that $\mathcal{C}\left(\overline{e}'_i, k_i\right) = a^{\overline{e}_i}$.
2. The random vectors are linearly independent with probability at least $1 - N/q$.

Sketch of proof

1. Run the extractor of the sigma proof N times with $\overline{e}_1, \ldots, \overline{e}_N$, each time extracting $\overline{e}'_i$ and k_i such that $\mathcal{C}\left(\overline{e}'_i, k_i\right) = a^{\overline{e}_i}$.
2. The random vectors are linearly independent with probability at least $1 - N/q$.
3. Linear independence implies existence of $\alpha_{\ell,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^{N} \alpha_{\ell,j}\overline{e}_j$ is the ℓth standard unit vector in $\mathbb{Z}_q^N$.

Sketch of proof

1. Run the extractor of the sigma proof N times with $\overline{e}_1, \ldots, \overline{e}_N$, each time extracting $\overline{e}'_i$ and k_i such that $\mathcal{C}\left(\overline{e}'_i, k_i\right) = a^{\overline{e}_i}$.
2. The random vectors are linearly independent with probability at least $1 - N/q$.
3. Linear independence implies existence of $\alpha_{\ell,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^{N} \alpha_{\ell,j} \overline{e}_j$ is the ℓth standard unit vector in $\mathbb{Z}_q^N$.
4. Then $\sum_{j=1}^{N} \alpha_{\ell,j} \overline{e}'_j$ is the ℓth column in M

Sketch of proof

1. Run the extractor of the sigma proof N times with $\overline{e}_1, \ldots, \overline{e}_N$, each time extracting $\overline{e}'_i$ and k_i such that $\mathcal{C}\left(\overline{e}'_i, k_i\right) = a^{\overline{e}_i}$.
2. The random vectors are linearly independent with probability at least $1 - N/q$.
3. Linear independence implies existence of $\alpha_{\ell,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^{N} \alpha_{\ell,j}\overline{e}_j$ is the ℓth standard unit vector in $\mathbb{Z}_q^N$.
4. Then $\sum_{j=1}^{N} \alpha_{\ell,j}\overline{e}'_j$ is the ℓth column in M since

$$a_l = \prod_{j=1}^{N} a^{\alpha_{\ell,j}\overline{e}_j} = \prod_{j=1}^{N} \mathcal{C}\left(\overline{e}'_j, k_j\right)^{\alpha_{\ell,j}} = \mathcal{C}\left(\sum_{j=1}^{N} \alpha_{\ell,j}\overline{e}'_j, \sum_{j=1}^{N} \alpha_{\ell,j}k_j\right)$$

Sketch of proof (2)

What if the extracted matrix M isn't a permutation matrix?

Sketch of proof (2)

What if the extracted matrix M isn't a permutation matrix?

1. If $M\overline{1} \neq \overline{1}$ then

$$\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} = \mathcal{C}\left(M\overline{1}, \langle \overline{s}, \overline{1} \rangle\right)$$

Sketch of proof (2)

What if the extracted matrix M isn't a permutation matrix?

1. If $M\overline{1} \neq \overline{1}$ then

$$\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} = \mathcal{C}\left(M\overline{1}, \langle \overline{s}, \overline{1} \rangle\right)$$

2. If $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle \neq \prod_{i=1}^{N} x_i$

Sketch of proof (2)

What if the extracted matrix M isn't a permutation matrix?

1. If $M\overline{1} \neq \overline{1}$ then

$$\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} = \mathcal{C}\left(M\overline{1}, \langle \overline{s}, \overline{1} \rangle\right)$$

2. If $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle \neq \prod_{i=1}^{N} x_i$ then we invoke the extractor to get $\overline{e}, \overline{e}'$ and k satisfying $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle \neq \prod_{i=1}^{N} e_i$.

Sketch of proof (2)

What if the extracted matrix M isn't a permutation matrix?

1. If $M\overline{1} \neq \overline{1}$ then

$$\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} = \mathcal{C}\left(M\overline{1}, \langle \overline{s}, \overline{1} \rangle\right)$$

2. If $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle \neq \prod_{i=1}^{N} x_i$ then we invoke the extractor to get $\overline{e}, \overline{e}'$ and k satisfying $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle \neq \prod_{i=1}^{N} e_i$. Observe that

$$\mathcal{C}\left(\overline{e}', k\right) = a^{\overline{e}} = \mathcal{C}\left(M\overline{e}, \langle \overline{s}, \overline{e} \rangle\right)$$

but $\overline{e}' \neq M\overline{e}$.

Restricting the permutation

Given that we can prove that a committed matrix is a permutation matrix, what other properties can we prove about the permutation?

Restricting the permutation

Given that we can prove that a committed matrix is a permutation matrix, what other properties can we prove about the permutation?

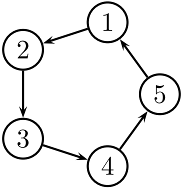
For example, can we prove that the permutation is a rotation [RW04, dHSSV09]?

Restricting the permutation

Given that we can prove that a committed matrix is a permutation matrix, what other properties can we prove about the permutation?

For example, can we prove that the permutation is a rotation [RW04, dHSSV09]?

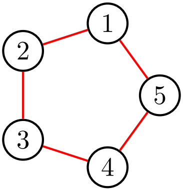
A rotation is precisely an automorphism of the directed cycle graph!



Restricting the permutation

Given that we can prove that a committed matrix is a permutation matrix, what other properties can we prove about the permutation?

For example, can we prove that the permutation is a rotation [RW04, dHSSV09]?



Let us look at the undirected cycle instead.

Restricting the permutation (graphs)

- Let $\mathscr{G}$ be a graph with vertices $V = \{1, 2, 3, \ldots, N\}$. Encode the edge set as

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j) \in E} x_i x_j \enspace .$$

Restricting the permutation (graphs)

- Let $\mathscr{G}$ be a graph with vertices $V = \{1, 2, 3, \ldots, N\}$. Encode the edge set as

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j)\in E} x_i x_j \quad .$$

- A permutation π is an automorphism of $\mathscr{G}$ if and only if

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = F_{\mathscr{G}}\left(x_{\pi(1)}, \ldots, x_{\pi(N)}\right) \quad .$$

Restricting the permutation (graphs)

- Let $\mathscr{G}$ be a graph with vertices $V = \{1, 2, 3, \ldots, N\}$. Encode the edge set as

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j) \in E} x_i x_j \ .$$

- A permutation π is an automorphism of $\mathscr{G}$ if and only if

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = F_{\mathscr{G}}\left(x_{\pi(1)}, \ldots, x_{\pi(N)}\right) \ .$$

- Apply Schwartz-Zippel ...

Restricting the permutation (directed graphs)

We can encode not only graphs, but also

- directed graphs
- labeled graphs
- hypergraphs
- etc.

Restricting the permutation (directed graphs)

We can encode not only graphs, but also

- directed graphs
- labeled graphs
- hypergraphs
- etc.

Returning to the rotation example, use the encoding polynomial

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j)\in E} x_i x_j^2$$

Restricting the permutation (directed graphs)

We can encode not only graphs, but also

- directed graphs
- labeled graphs
- hypergraphs
- etc.

Returning to the rotation example, use the encoding polynomial

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j) \in E} x_i x_j^2 = x_1 x_2^2 + x_2 x_3^2 + x_3 x_4^2 + x_4 x_5^2 + x_5 x_1^2$$

Restricting the permutation (directed graphs)

We can encode not only graphs, but also

- directed graphs
- labeled graphs
- hypergraphs
- etc.

Returning to the rotation example, use the encoding polynomial

$$F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j)\in E} x_i x_j^2 = x_1 x_2^2 + x_2 x_3^2 + x_3 x_4^2 + x_4 x_5^2 + x_5 x_1^2$$

Testing $F_{\mathscr{G}}(x_1, \ldots, x_N) = F_{\mathscr{G}}(x_{\pi(1)}, \ldots, x_{\pi(N)})$ determines whether π is a rotation.

Restricting the permutation (polynomials)

Theorem

Let F be any polynomial in $\mathbb{Z}_q[x_1, \ldots, x_N]$ and let S_F be the group of permutations π such that

$$F(x_1, \ldots, x_N) = F(x_{\pi(1)}, \ldots, x_{\pi(N)}) \ .$$

Then we can prove that the permutation is chosen from S_F.

Summary

We have demonstrated

Summary

We have demonstrated

- an efficient proof of a shuffle with a simple analysis

Summary

We have demonstrated

- an efficient proof of a shuffle with a simple analysis
- a general method for restricting the permutation to certain groups

Summary

We have demonstrated

- an efficient proof of a shuffle with a simple analysis
- a general method for restricting the permutation to certain groups

Problem Are there applications for other restrictions than rotations, e.g. automorphisms of a complete binary tree?

Questions?

References I

R. Cramer, R. Gennaro, and B. Schoenmakers.
A secure and optimally efficient multi-authority election scheme.
In *Advances in Cryptology – Eurocrypt '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer Verlag, 1997.

D. Chaum.
Untraceable electronic mail, return addresses and digital pseudo-nyms.
Communications of the ACM, 24(2):84–88, 1981.

References II

S. de Hoogh, B. Schoenmakers, B. Skoric, and J. Villegas.
Verifiable rotation of homomorphic encryptions.
In *Public Key Cryptography – PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 393–410. Springer Verlag, 2009.

J. Furukawa and K. Sako.
An efficient scheme for proving a shuffle.
In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 368–387. Springer Verlag, 2001.

References III

A. Neff.
A verifiable secret shuffle and its application to e-voting.
In *8th ACM Conference on Computer and Communications Security (CCS)*, pages 116–125. ACM Press, 2001.

M. K. Reiter and X. Wang.
Fragile mixing.
In *11th ACM Conference on Computer and Communications Security (CCS)*, pages 227–235. ACM Press, 2004.

K. Sako and J. Killian.
Reciept-free mix-type voting scheme.
In *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer Verlag, 1995.

References IV