A Lightweight Implementation of a Shuffle Proof for Electronic Voting Systems

Philipp Locher and Rolf Haenni

Bern University of Applied Sciences

Informatik 2014, Stuttgart

September 2014

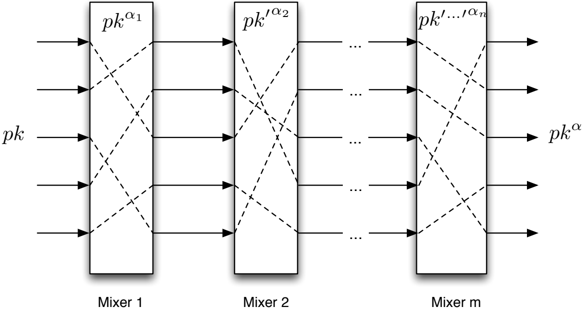
This work is supported by the Swiss National Science foundation, under the grant 200021L-140650/1

UniVote: Verifiable Electronic Voting over the Internet

- Internet voting system for student board elections at Swiss universities
- Project started in 2012
- First elections in spring 2013
- 6 elections were held successfully
- https://www.univote.ch

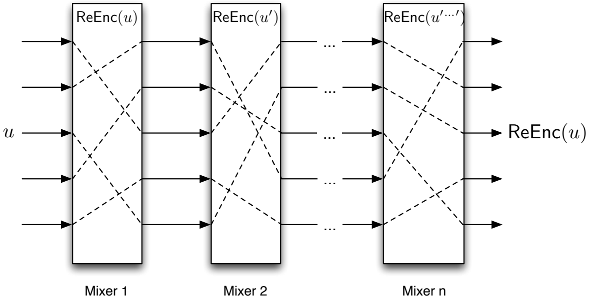
UniVote is (not yet) end-to-end verifiable and offers anonymized vote casting [Neff01, HS11].

Mixing of public keys:



Before the final decryption and tally phase, the ballots are mixed.

- Late registration (students cannot be forced to register before voting phase)
- Anonymous channel cannot always be expected
- No performance issue (only a few thousand ballots)



UniCrypt is a cryptographic Java library:

- Simplifies the implementation of cryptographic voting protocols
- Split into two layers: mathematical fundament and cryptographic primitives
- Type safety on a mathematical level
- https://github.com/bfh-evg/unicrypt

Mix-Net Implementations

- Verificatum: An implementation of a full-featured mix-net by Wikström
- A number of prototype implementations of shuffle proofs

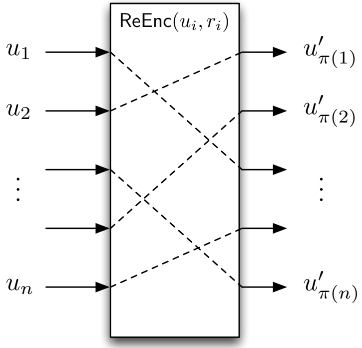
Contribution

A new implementation of a shuffle proof

- Based on the findings of Wikström and Terelius [Wik09, TW10]
- Embedded in a cryptographic library with a clean and intuitive application programming interface
- Full flexibility with respect to the encryption system and the algebraic groups
- Support for different types of mix-nets
- Portable to any device running a Java Virtual Machine

Proof of a (Re-Encryption) Shuffle

Proof that each ciphertext of a list of ciphertexts has been re-encrypted and permuted



Wikström/Terelius's Shuffle Proof

[**Wik09**] A Commitment-Consistent Proof of a Shuffle

- Offline part: Commit to a permutation matrix and proof that it is indeed a permutation matrix.
- Online part: Shuffle the input batch and give a commitment-consistent proof of a shuffle.

[**TW10**] Proofs of Restricted Shuffles

- Restricting the set of permutations.
- A new proof of a shuffle based on a permutation matrix.

Implementation

Shuffling and the shuffle proof are implemented in UniCrypt inside the following cryptographic components:

Mixer Covers the shuffle functionality without proving its correctness. Two implementations: *re-encryption mixer* and *identity mixer*

Proof System Holds all types of zero-knowledge proofs, including the proof of a shuffle

Challenge Generator Creates challenges in an interactive or non-interactive manner

Example of Usage

```
// Select cyclic group for safe prime p=2q+1 (1024 bit)
CyclicGroup group = GStarModSafePrime.getRandomInstance(1024);

// Create ElGamal encryption scheme and key pair
ElGamalEncryptionScheme elGamal =
    ElGamalEncryptionScheme.getInstance(group);

Pair keys = elGamal.getKeyPairGenerator().generateKeyPair();
Element pk = keys.getSecond();

// Set shuffle size and create random ElGamal ciphertexts
int n = 100;
Tuple ciphertexts = Tuple.getInstance();
for (int i = 0; i < n; i++) {
    Element m = group.getRandomElement();
    Pair c = elGamal.encrypt(pk, m);
    ciphertexts = ciphertexts.add(c);
}
```

Listing 1: Setup

Example of Usage

```
// Create mixer, random permutation pi, and randomizations r
ReEncryptionMixer mixer =
    ReEncryptionMixer.getInstance(elGamal, pk, n);

PermutationElement pi =
    mixer.getPermutationGroup().getRandomElement();

Tuple r = mixer.generateRandomizations();

// Shuffle ciphertexts using pi and r
Tuple shuffledCiphertexts = mixer.shuffle(ciphertexts, pi, r);
```

Listing 2: Shuffle

Example of Usage

```
// Create permutation commitment c_pi based on pi
// and randomizations s
PermutationCommitmentScheme pcs =
    PermutationCommitmentScheme.getInstance(group, n);
Tuple s = pcs.getRandomizationSpace().getRandomElement();
Tuple c_pi = pcs.commit(pi, s);

// Create permutation commitment proof system
PermutationCommitmentProofSystem pcps =
    PermutationCommitmentProofSystem.getInstance(group, n);

// Define private and public input
Pair offlinePrivateInput = Pair.getInstance(pi, s);
Element offlinePublicInput = c_pi;

// Generate permutation commitment proof
Pair offlineProof =
    pcps.generate(offlinePrivateInput, offlinePublicInput);
```

Listing 3: Online Phase (Proof of Knowledge of Permutation Matrix)

Example of Usage

```
// Create shuffle proof system
ReEncryptionShuffleProofSystem rsps =
    ReEncryptionShuffleProofSystem
        .getInstance(group, n, elGamal, pk);

// Define private and public input
Triple onlinePrivateInput = Triple.getInstance(pi, s, r);
Triple onlinePublicInput =
    Triple.getInstance(c_pi, ciphertexts, shuffledCiphertexts);

// Generate shuffle proof
Triple onlineProof =
    rsps.generate(onlinePrivateInput, onlinePublicInput);
```

Listing 4: Online Phase (Commitment Consistent Proof of a Shuffle)

Example of Usage

```
// Verify permutation commitment proof
boolean v1 = pcps.verify(offlineProof, offlinePublicInput);

// Verify shuffle proof
boolean v2 = rsps.verify(onlineProof, onlinePublicInput);

// Verify equality of permutation commitments
boolean v3 =
  offlinePublicInput.isEquivalent(onlinePublicInput.getFirst());

if (v1 && v2 && v3) success();
```

Listing 5: Proof Verification

Thank you!

Wikström/Terelius's Shuffle Proof

An $N \times N$ - matrix M is a permutation matrix if there is exactly one non-zero element in each row and column and if this non-zero element is equal to one.

Example:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_1 \\ x_2 \end{pmatrix}$$

If M_π is a permutation matrix for the permutation π then

$$M_\pi \cdot \bar{x} = \bar{x}' = (x_{\pi(1)}, \ldots, x_{\pi(N)})$$

Wikström/Terelius's Shuffle Proof

Theorem (Permutation Matrix) [TW10]
Let $M = (m_{i,j})$ be an $N \times N$ - matrix over $\mathbb{Z}_q$ and $\bar{x} = (x_1, \ldots, x_N)$ be a list of variables. Then M is a permutation matrix if and only if

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

m_i denotes the i-th row vector of M and $\langle \bar{m}_i, \bar{x} \rangle = \sum_{j=1}^{N} m_{i,j}\, x_j$

Wikström/Terelius's Shuffle Proof

A matrix commitment based on the generalized Pedersen commitment has the property:

$$\langle Com(M, \bar{s}), \bar{e} \rangle = Com(M\bar{e}, \langle \bar{s}, \bar{e} \rangle)$$

It follows that if M is a permutation matrix then $M\bar{e} = \bar{e}' = (e_{\pi(1)}, \ldots, e_{\pi(N)})$ and $\langle Com(M, \bar{s}), \bar{e} \rangle$ is a publicly computed commitment to the permuted $\bar{e}$ - vector based on the commitment to M.

Proof of Knowledge of Permutation Matrix (offline) 1/2

Common Input: Matrix commitment c_π
Private Input: Permutation matrix M_π and $\bar{s}$ such that $c_\pi = Com(M_\pi, \bar{s})$.

1. $\mathcal{V}$ chooses $\bar{e} \in \mathbb{Z}_q^N$ randomly and hands $\bar{e}$ to $\mathcal{P}$
2. $\mathcal{P}$ computes $v = \langle \bar{s}, \bar{1} \rangle$, $w = \langle \bar{s}, \bar{e} \rangle$ and $\bar{e}' = M_\pi \bar{e}$.
3. $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{matrix} v, w \in \mathbb{Z}_q \\ \bar{e}' \in \mathbb{Z}_q^N \end{matrix} \;\middle|\; Com(\bar{1}, v) = \langle c_\pi, \bar{1} \rangle \wedge Com(\bar{e}', w) = \langle c_\pi, \bar{e} \rangle \wedge \prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i \right]$$

Proof of Knowledge of Permutation Matrix (offline) 2/2

The Σ-proof of the proof of knowledge of permutation matrix can be transformed into a generic preimage proof by the homomorphic one-way function:

$$\phi_{offline}(v, w, \bar{t}, d, \bar{e}') = \Big(Com(\bar{1}, v), Com(\bar{e}', w), g^{t_1} c_0^{e'_1}, \ldots, g^{t_N} c_{N-1}^{e'_N}, Com(0, d)\Big)$$

With additional private input: Randomness $\bar{t} \in \mathbb{Z}_q^N$ and $d = d_N$ and $d_i = t_i + e'_i d_{i-1}$ for $i > 2, \ldots, N$ with $d_1 = t_1$. $c_i = g^{t_i} c_{i-1}^{e'_i}$ and $c_0 = h$.

Commitment-Consistent Proof of a Shuffle (online) 1/2

Common Input: Permutation matrix commitment c_π and ciphertexts (ElGamal) $u_1, \ldots, u_N, u'_1, \ldots, u'_N \in (G_q \times G_q)$.
Private Input: Permutation π and randomness $\bar{r} \in \mathbb{Z}_q^N$ such that $u'_i = ReEnc(u_{\pi(i)}, r_{\pi(i)})$.

1. $\mathcal{V}$ chooses $\bar{e} \in \mathbb{Z}_q^N$ randomly and hands $\bar{e}$ to $\mathcal{P}$
2. $\mathcal{P}$ computes $w = \langle \bar{s}, \bar{e} \rangle$, $r = \langle \bar{r}, \bar{e} \rangle$ and $\bar{e}' = M_\pi \bar{e}$.
3. $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{matrix} r, w \in \mathbb{Z}_q \\ \bar{e}' \in \mathbb{Z}_q^N \end{matrix} \;\middle|\; Com(\bar{e}', w) = \langle c_\pi, \bar{e} \rangle \wedge \prod_{i=1}^{N} (u'_i)^{e'_i} = ReEnc(\prod_{i=1}^{N} (u_i)^{e_i}, r)\right]$$

Commitment-Consistent Proof of a Shuffle (online) 2/2

The Σ-proof of the proof of knowledge of permutation matrix can be transformed into a generic preimage proof by the homomorphic one-way function:

$$\phi_{online}(r, w, \bar{e}') = \left(Com(\bar{e}', w), \prod_{i=1}^{N} (u_i')^{e_i'} Enc(1, -r) \right)$$