NTNU

Norwegian University of Science and Technology

# Electronic Voting Using Lattice-Based Commitments and Verifiable Encryption

Aarhus Crypto Seminar

Carsten Baum, Kristian Gjøsteen, **Tjerand Silde** and Thor Tunge

**Department of Mathematical Sciences,**
**NTNU Trondheim**

NTNU

# Takeaway I

We propose a protocol for remote electronic voting using lattice-based commitments and verifiable encryption. It is the first practical construction for electronic voting that supports complex ballots and is built from post-quantum assumptions. Our scheme is also the first to defend against compromise of the voter's computer using return codes.

# Takeaway II

The core of our protocol is a verifiable shuffle of known values inspired by Neff's construction (ACM CCS 2001). Our shuffle uses the lattice- based commitments from Baum et al. (SCN 2018).

# Takeaway III

To prevent malformed encryptions, we use the verifiable encryption scheme of Lyubashevsky and Neven (EUROCRYPT 2017). We reuse this trick of verifiable encryption of commitment openings to create a practical return code mechanism.

# Overview

- Security Definitions
- The Protocol Architecture
- Lattice-Based Commitments
- The Shuffle Protocol
- Proof of Encrypted Opening
- Return Codes
- The Voting Scheme
- Parameters and Efficiency
- Improvements and Future Work

# Security Definitions I

If a voting system gives the correct answer, relative to some ideally determined collection of ballots and some counting function, we have *integrity*.

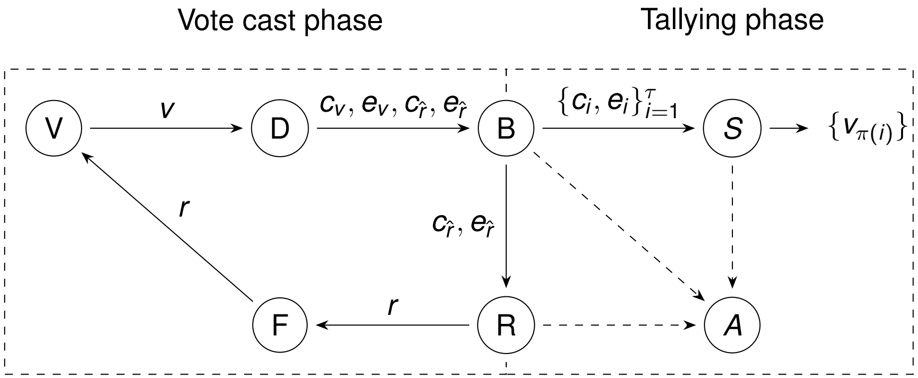
If it is hard to determine what ballot a given voter cast, up to what can be deduced from the election outcome, we have *privacy*.

# Security Definitions II

If voters can ensure that the ballot of their choice is counted, even when an adversary can control the voter during ballot casting, we have *coercion resistance*.

If the ballot casting and the talley phase produce transcripts that allow voters to verify that the count included their ballots, we have *verifiability*.

# The Protocol Architecture



# Lattice-Based Commitments from Baum et al.

$\texttt{Com}$ commits to messages $m \in R_p$ by sampling an $\boldsymbol{r}_m \xleftarrow{\$} S_{\beta_\infty}^k$ and computing

$$\texttt{Com}(m; \boldsymbol{r}_m) = \boldsymbol{B} \cdot \boldsymbol{r}_m + \begin{bmatrix} \boldsymbol{0}^n \\ m \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 \\ c_2 \end{bmatrix} = [m] \,.$$

$\texttt{Com}$ outputs $c = [m]$ and $d = (m; \boldsymbol{r}_m, 1)$.

$\texttt{Open}$ verifies whether an opening $(m, \boldsymbol{r}_m, f)$ with $f \in \bar{\mathcal{C}}$ is a valid opening by checking if

$$f \cdot \begin{bmatrix} \boldsymbol{c}_1 \\ c_2 \end{bmatrix} \stackrel{?}{=} \boldsymbol{B} \cdot \boldsymbol{r}_m + f \cdot \begin{bmatrix} \boldsymbol{0}^n \\ m \end{bmatrix},$$

and that $\|r_i\| \leq 4\sigma\sqrt{N}$ for $\boldsymbol{r}_m = (r_0, \ldots r_{k-1})$ with $\sigma = 11 \cdot \nu \cdot \beta_\infty \cdot \sqrt{kN}$.

$\texttt{Open}$ outputs 1 if all these conditions holds, and 0 otherwise.

# The Shuffle Protocol I

1. The prover $\mathcal{P}$ and verifier $\mathcal{V}$ receives a set of commitments $\{[m_i]\}_{i=1}^{\tau}$,
2. $\mathcal{P}$ also receives the set of openings $\{(m_i, r_m)\}_{i=1}^{\tau}$ of the commitments,
3. $\mathcal{V}$ picks a random $\rho \xleftarrow{\$} \Sigma_{\beta_\infty}$ and sends $\rho$ to $\mathcal{P}$,
4. $\mathcal{P}$ and $\mathcal{V}$ shifts the commitments to get $M_i = m_i - \rho$,
5. $\mathcal{P}$ picks a random permutation $\pi \in S_\tau$,
6. $\mathcal{P}$ shuffles the messages by defining $\hat{M}_i := M_{\pi^{-1}(i)}$ for all $i \in \{1, \dots, \tau\}$,
7. $\mathcal{P}$ sends the set of shuffled messages $\{\hat{M}_i\}_{i=1}^{\tau}$ to the verifier $\mathcal{V}$,
8. $\mathcal{P}$ proves to the verifier $\mathcal{V}$ in a public coin 6-move protocol $\Pi_S$ that the set of shuffled messages is indeed the openings of the received commitments.

# The Shuffle Protocol II

$\mathcal{P}$ picks a random permutation $\pi \in S_\tau$ and defines $\hat{M}_i = M_{\pi^{-1}(i)}$ for all $i$.

$\mathcal{P}$ draws $\theta_i \xleftarrow{\$} R_p$ uniformly at random for each $i$ and computes

$$
\begin{aligned}
D_1 &= \left[\theta_1 \hat{M}_1\right] \\
D_2 &= \left[\theta_1 M_2 + \theta_2 \hat{M}_2\right] \\
&\vdots \\
D_{\tau-1} &= \left[\theta_{\tau-2} M_{\tau-1} + \theta_{\tau-1} \hat{M}_{\tau-1}\right] \\
D_\tau &= \left[\theta_{\tau-1} M_\tau\right].
\end{aligned}
$$

# The Shuffle Protocol III

$\mathcal{V}$ chooses a challenge $\beta \in R_p$.

$\mathcal{P}$ computes $s_i \in R_q$ such that the following equations are satisfied:

$$
\begin{aligned}
\beta M_1 + s_1 \hat{M}_1 &= \theta_1 \hat{M}_1 \\
s_1 M_2 + s_2 \hat{M}_2 &= \theta_1 M_2 + \theta_2 \hat{M}_2 \\
&\vdots \\
s_{\tau-2} M_{\tau-1} + s_{\tau-1} \hat{M}_{\tau-1} &= \theta_{\tau-2} M_{\tau-2} + \theta_{\tau-1} \hat{M}_{\tau-1} \\
(-1)^{\tau} \beta \hat{M}_{\tau} + s_{\tau-1} M_{\tau-1} &= \theta_{\tau-1} M_{\tau}.
\end{aligned}
$$

# The Shuffle Protocol IV

This can be written as a matrix-equation:

$$\underbrace{\begin{bmatrix} \hat{M}_1 & 0 & \dots & 0 & 0 \\ M_2 & \hat{M}_2 & \dots & 0 & 0 \\ 0 & M_3 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & M_{\tau-1} & \hat{M}_{\tau-1} \\ 0 & 0 & \dots & 0 & M_\tau \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} s_1 - \theta_1 \\ s_2 - \theta_2 \\ s_3 - \theta_3 \\ \vdots \\ s_{\tau-2} - \theta_{\tau-2} \\ s_{\tau-1} - \theta_{\tau-1} \end{bmatrix}}_{s-\theta} = \underbrace{\begin{bmatrix} -\beta M_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ (-1)^{\tau-1}\beta \hat{M}_\tau \end{bmatrix}}_{b}.$$

# The Shuffle Protocol V

**Theorem**

*The shuffle protocol is complete.*

**Theorem**

*The shuffle protocol in is sound for any PPT $\mathcal{P}^*$ that wins with probability $> \frac{\tau^\delta}{p^N}$.*

# The Shuffle Protocol VI

Simulation to prove HVZK:

**Real**

$$\begin{aligned}
1:&\quad \theta_i \xleftarrow{\$} R_p \\
2:&\quad D_i \leftarrow (2) \\
3:&\quad \text{Send } D_i \\
4:&\quad s_i \leftarrow (3) \\
5:&\quad \text{Send } s_i \\
6:&\quad \text{SHVZK}
\end{aligned}$$

**Game 1**

$$\begin{aligned}
1:&\quad \theta_i \xleftarrow{\$} R_p \\
2:&\quad D_i \leftarrow (2) \\
3:&\quad \text{Send } D_i \\
4:&\quad s_i \leftarrow (3) \\
5:&\quad \text{Send } s_i \\
6:&\quad \boxed{\textbf{Sim}}
\end{aligned}$$

**Game 2**

$$\begin{aligned}
1:&\quad \boxed{s_i \xleftarrow{\$} R_p} \\
2:&\quad \theta_i \xleftarrow{\$} R_p \\
3:&\quad \boxed{D_i \leftarrow (6)} \\
4:&\quad \text{Send } D_i \\
5:&\quad \text{Send } s_i \\
6:&\quad \text{Sim}
\end{aligned}$$

**Simulator**

$$\begin{aligned}
1:&\quad \boxed{D_i \leftarrow [0]} \\
2:&\quad s_i \xleftarrow{\$} R_p \\
3:&\quad \text{Send } D_i \\
4:&\quad \text{Send } s_i \\
5:&\quad \text{Sim}
\end{aligned}$$

# Proof of Encrypted Opening I

We want to prove that we know $\mu$ such that $T\mu = u$ for public $T$ and $u$.

Using the verifiable encryption scheme by Lyubashevsky and Neven, we can prove that we know a small $\bar{\mu}$ and $\bar{c}$ such that

$$T\bar{\mu} = \bar{c}u.$$

# Proof of Encrypted Opening II

We can re-write the commitment as a single matrix-vector multiplication:

$$\mathtt{Com}(m; r_m) = \begin{bmatrix} B_1 & 0^n \\ b_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_m \\ m \end{bmatrix}.$$

Denote this matrix by $C$.

# Proof of Encrypted Opening III

We can use the verifiable encryption scheme to encrypt the opening $\mu = (r_m, m)$, and prove that the voter knows a witness for the relation

$$C\mu = C \cdot \begin{bmatrix} r_m \\ m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$$

# Return Codes I

Assume that the voters have $\omega$ different options in the election.

Further, let

- $v_1, v_2, \ldots, v_\omega \in S_{\beta_\infty}$ be possible ballots,
- $a_0 \in R_p$ be a blinding-key for a voter $V$,
- $\mathtt{PRF}_k : \{0,1\}^* \times R_p \to \{0,1\}^n$ be a pseudo-random function,

# Return Codes II

The *pre-code* $\hat{r}_j$ corresponding to the ballot $v_j$ is $\hat{r}_j = v_j + a_0$.

The *return code* $r_j$ corresponding to the ballot $v_j$ is $r_j = \mathtt{PRF}_k(V, \hat{r}_j)$.

Let $c_{a_0}$, $c_{v_j}$ and $c_{\hat{r}}$ be commitments to the blinding key, the ballot and the pre-code.

We can prove in zero-knowledge that $\hat{r}_j = a_0 + v_j$ is correct.

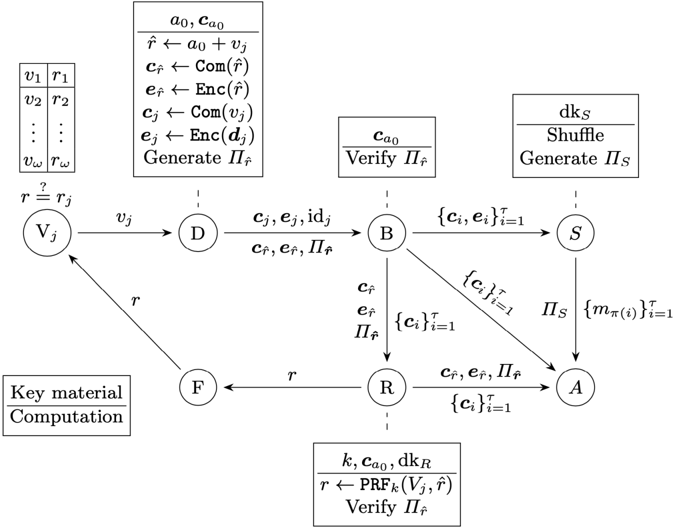
# Return Codes III

A commitment to voter $V$'s voter-unique blinding key $a_0$ is public.

The voter $V$'s *computer* computes the pre-code $\hat{r}_j = v_j + a_0$, a commitment $c_{\hat{r}_j}$ to $\hat{r}_j$ and a proof $\Pi_{\hat{r}_j}$. Finally, it will verifiably encrypt as $e_{\hat{r}_j}$ the opening of $c_{\hat{r}_j}$ with the return code generator's public key.

The *return code generator* receives $V$, $c_{\hat{r}_j}$, $c_{v_j}$, $e_{\hat{r}_j}$ and $\Pi_{\hat{r}_j}$. It verifies the proof and the encryption, and then decrypts the ciphertext to get $\hat{r}_j$. It computes the return code as $r_j = \mathtt{PRF}_k(V, \hat{r}_j)$.

# The Voting Scheme



# Parameters and Efficiency

A vote $(c_j, e_j, c_{\hat{r}}, e_{\hat{r}}, \Pi_{\hat{r}})$ is of total size $\approx$ 400 KB.

For $\tau$ voters, the ballot box $\mathcal{B}$ receives $\approx 400\tau$ KB of data.

The shuffle proof is of total size $\approx 21\tau$ KB.

| Commitments | Encryption | Verification | Shuffle Proof |
|---|---|---|---|
| 0.12ms | 60ms | 4ms | $12\tau$ms |

# Improvements and Future Work

Can we...

— extend the shuffle to handle arbitrary ring elements?

— extend this into a mix-net with more than one shuffle-server?

— extend the return-code mechanism to handle re-voting?

— aggregate zero-knowledge proofs for all equations in the shuffle?

# *Thank You! Questions?*

Email: tjerand.silde@ntnu.no
Slides: `www.tjerandsilde.no/talks`