NTNU | Norwegian University of Science and Technology

LATTICE-BASED VERIFIABLE SHUFFLE AND DECRYPTION

Diego Aranha, Carsten Baum, Kristan Gjøsteen, Thomas Haines, Johannes Muller, Peter Rønne, **Tjerand Silde** and Thor Tunge

November 26, 2020

Introduction

Preliminaries

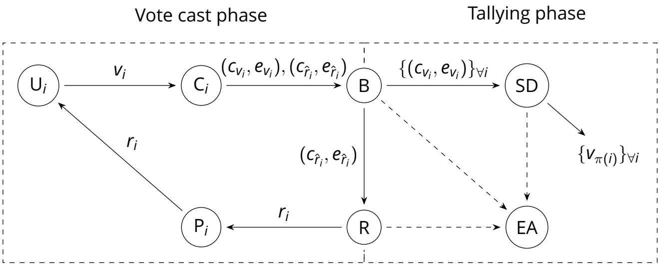
Proof of Shuffle

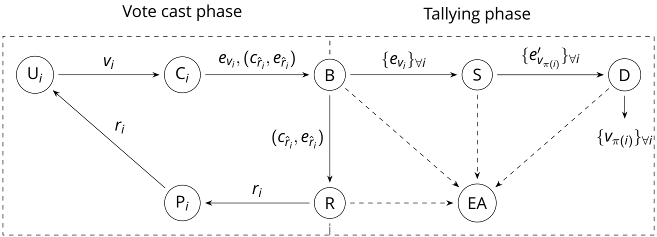
Mixing Network

Verifiable Key-Shifting

Verifiable Decryption

Electronic Voting





Introduction - Goals

1. Build a zero-knowledge protocol to prove correct shuffle of messages
2. Extend the shuffle to handle ciphertexts instead of messages
3. Build a mixing network from the extended shuffle
4. Construct a return-code protocol to achieve voter verifiability
5. Combine everything to construct systems for electronic voting
6. Use primitives based on lattices to achieve post-quantum security

Preliminaries - Commitment

`KeyGen` outputs a public matrix $\boldsymbol{B}$ of the form

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} = \begin{bmatrix} 1 & b_1 & b_2 \\ 0 & 1 & b_3 \end{bmatrix}, \qquad b_1, b_2, b_3 \xleftarrow{\$} R_q = \mathbb{Z}[X]/\langle X^N + 1 \rangle.$$

`Com` commits to messages $m \in R_q$ by sampling an $\boldsymbol{r}_m \xleftarrow{\$} S_1^3$ as

$$\mathtt{Com}(m; \boldsymbol{r}_m) = \boldsymbol{B} \cdot \boldsymbol{r}_m + \begin{bmatrix} 0 \\ m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = [m] \, .$$

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

`Open` verifies whether an opening $(m, \boldsymbol{r}_m, f)$ checking if

$$f \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \stackrel{?}{=} \boldsymbol{B} \cdot \boldsymbol{r}_m + f \cdot \begin{bmatrix} 0 \\ m \end{bmatrix}, \qquad \|r_i\| \stackrel{?}{\leq} 4\sigma_C \sqrt{N}$$

Preliminaries - Proof of Linearity

Let

$$[x] = \mathtt{Com}(x; \boldsymbol{r}) \quad \text{and} \quad \left[x'\right] = [\alpha x + \beta] = \mathtt{Com}(x'; \boldsymbol{r}').$$

T hen the protocol Π_{Lin} is a sigma-protocol to prove the relation $x' = \alpha x + \beta$, given the commitments $[x]$, $[x']$ and the scalars α, β.

For more details about the commitment and proof see Baum et al. [BDL+18].

Preliminaries - Amortized Proof of Shortness

Let

$$[x_1] = \texttt{Com}(x_1; \boldsymbol{r}_1), \quad [x_2] = \texttt{Com}(x_2; \boldsymbol{r}_2), \quad \ldots, \quad [x_n] = \texttt{Com}(x_n; \boldsymbol{r}_n),$$

where all are commitments to short values. Then the protocol Π_A is a sigma-protocol to prove that the underlying messages of $[x_1], [x_2], \ldots, [x_n]$ are bounded.

For more details about the amortized proof see Baum et al. [BBC+18].

Preliminaries - BGV Encryption

`KeyGen` samples random $a \xleftarrow{\$} R_q$, short $s \leftarrow R_q$ and noise $e \leftarrow \mathcal{N}_\sigma$.
The algorithm outputs $\mathtt{pk} = (a, b) = (a, as + pe)$ and $\mathtt{sk} = s$.

`Enc` samples a short $r \leftarrow R_q$ and noise $e_1, e_2 \leftarrow \mathcal{N}_\sigma$, and outputs
$(u, v) = (ar + pe_1, br + pe_2 + m)$.

`Dec` outputs $m \equiv v - su \mod q \mod p$ when noise is bounded by $\lfloor q/2 \rfloor$.

For more details about the encryption scheme see Brakerski et al. [BGV12].

Proof of Shuffle - Setting

- Public information: sets of commitments $\{[m_i]\}_{i=1}^{\tau}$ and messages $\{\hat{m}_i\}_{i=1}^{\tau}$.
- P knows the openings $\{(m_i, \boldsymbol{r}_{m_i}, f_i)\}_{i=1}^{\tau}$ of the commitments $\{[m_i]\}_{i=1}^{\tau}$,
- and P knows a permutation π such that $\hat{m}_i = m_{\pi^{-1}(i)}$ for all $i = 1, \ldots, \tau$.
- We construct a $4 + 3\tau$-move ZKPoK protocol to prove this statement.
- This extends Neff's construction [Nef01] to the realm of PQ assumptions.

Proof of Shuffle - Linear System

As a first step, P draws $\theta_i \xleftarrow{\$} R_q$ uniformly at random for each $i \in \{1, \ldots, \tau\}$, and computes the commitments:

$$\begin{aligned} [D_1] &= \left[\theta_1 \hat{M}_1\right] \\ \forall j \in \{2, \ldots, \tau - 1\} : \ [D_j] &= \left[\theta_{j-1} M_j + \theta_j \hat{M}_j\right] \\ [D_\tau] &= [\theta_{\tau-1} M_\tau] \,. \end{aligned} \tag{1}$$

Proof of Shuffle - Linear System

P receives a challenge $\beta \in R_q$ and computes $s_i \in R_q$ such that the following equations are satisfied:

$$\begin{aligned} \beta M_1 + s_1 \hat{M}_1 &= \theta_1 \hat{M}_1 \\ \forall j \in \{2, \ldots, \tau - 1\} : \; s_{j-1} M_j + s_j \hat{M}_j &= \theta_{j-1} M_j + \theta_j \hat{M}_j \\ s_{\tau-1} M_\tau + (-1)^\tau \beta \hat{M}_\tau &= \theta_{\tau-1} M_\tau . \end{aligned} \tag{2}$$

Proof of Shuffle - Linear System

P uses the protocol Π_{Lin} to prove that each commitment $[D_i]$ satisfies the equations (2). In order to compute the s_i values, we can use the following fact:

Lemma

Choosing

$$s_j = (-1)^j \cdot \beta \prod_{i=1}^{j} \frac{M_i}{\hat{M}_i} + \theta_j \tag{3}$$

for all $j \in 1, \ldots, \tau - 1$ yields a valid assignment for Equation (2).

Proof of Shuffle - Protocol

Zero-Knowledge Proof Π_{Shuffle} of Correct Shuffle

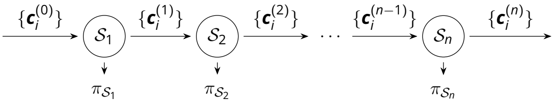
Prover, P		Verifier, V
	$\xleftarrow{\rho}$	$\rho \xleftarrow{\$} R_q \setminus \{\hat{m}_i\}_{i=1}^{\tau}$
$\hat{M}_i = \hat{m}_i - \rho$		$\hat{M}_i = \hat{m}_i - \rho$
$M_i = m_i - \rho$		$[M_i] = [m_i] - \rho$
$\theta_i \xleftarrow{\$} R_q, \forall i \in [\tau - 1]$		
Compute $[D_i]$ as in Eq. (1), i.e.		
$[D_1] = [\theta_1 \hat{M}_1], [D_\tau] = [\theta_{\tau-1} M_\tau]$,		
$[D_i] = [\theta_{i-1} M_i + \theta_i \hat{M}_i]$ for $i \in [\tau - 1] \setminus \{1\}$	$\xrightarrow{\{[D_i]\}_{i=1}^{\tau}}$	
	$\xleftarrow{\beta}$	$\beta \xleftarrow{\$} R_q$
Compute $s_i, \forall i \in [\tau - 1]$ as in (3).	$\xrightarrow{\{s_i\}_{i=1}^{\tau-1}}$	
		Use Π_{Lin} to prove that
		(1) $\beta[M_1] + s_1 \hat{M}_1 = [D_1]$
		(2) $\forall i \in [\tau - 1] \setminus \{1\}: \ s_{i-1}[M_i] + s_i \hat{M}_i = [D_i]$
		(3) $s_{\tau-1}[M_\tau] + (-1)^\tau \beta \hat{M}_\tau = [D_\tau]$
		i.e. all equations from (2)

Proof of Shuffle - Performance

- Optimal parameters for the commitment scheme is $q \approx 2^{32}$ and $N = 2^{10}$.
- The proof of linearity use Gaussian noise of standard deviation $\sigma \approx 2^{15}$.
- The prover sends 1 commitment, 1 ring-element and 1 proof per message.
- The shuffle proof is of total size $\approx 21\tau$ KB for τ messages.
- The shuffle proof takes time $\approx 18\tau$ ms to compute for τ messages.

Mixing Network - Extending the Shuffle

- We extend the shuffle to ciphertexts instead of messages
- We create a mixing network that does the following:
  1. Randomize the ciphertexts
  2. Commit to the randomness
  3. Permute the ciphertexts
  4. Prove that shuffle is correct
  5. Prove that the noise is short
- Integrity holds because of the proofs
- Privacy if at least one server is honest



Verifiable Key-Shifting - Protocol

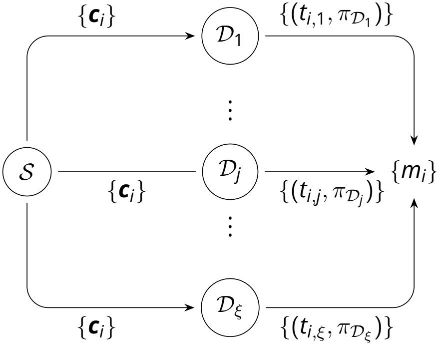
- We're given a ciphertext (u, v) under s_1.
- We want the ciphertext (u', v') under $s = s_1 + s_2$.
- The protocol works as following:
  1. Compute $(u', v') = (u + ar' + pE_1, v + us_2 + br' + pE_2)$
  2. We need s_1 and s_2 to be short to achieve correctness
  3. We need E_1 and E_2 to be 2^{sec} larger than s for privacy
  4. We use Π_{Lin} to prove correctness of each computation
  5. We use Π_{A} to prove that E_1 and E_2 are bounded
- Distributed protocol for $s_2 = \sum_j \hat{s}_j$ where $\hat{s}_j$ are random.

Verifiable Decryption - Distributed Decryption

Actively secure distributed decryption protocol from SPDZ [DPSZ12]:

- On input key s_j and ciphertext (u, v), sample large noise E_j, output $t_j = s_j u + pE_j$.
- We use Π_{Lin} to prove correct computation.
- We use Π_{A} to prove that E_j is bounded.

We obtain the plaintext as $m \equiv (v - t \mod q) \mod p$, where $t = t_1 + t_2 + \ldots + t_\xi$.



Verifiable Decryption - One-Party Decryption

We can decrypt directly as following:

- Public commitment to secret key s.
- Compute $m_i \equiv (v_i - su_i \mod q) \mod p$.
- Commit to $d_i = v_i - su_i - m_i$.
- Use Π_{Lin} to prove correct computation.
- We use Π_{A} to prove that d_i is bounded.

Verifiable Decryption - MPC in the Head

1. Deal splits the key into two parts and prove correctness.
2. Play compute a decryption share $t_{i,j}$ based on key share s_i.
3. P commits to the shares, and V challenges half of them.
4. V checks correctness of shares.
5. V reconstructs to check the message from the shares.

Π_{ZKPCD}

Prover$((\mathsf{pk}, \{c_j\}_{j=1}^{\tau}, \{m_j\}_{j=1}^{\tau}), (\mathsf{sk}))$

$k = 1, \dots, \lambda:$

$\quad (\mathsf{sk}_{0,k}, \mathsf{sk}_{1,k}, \mathsf{aux}_k) \leftarrow \mathsf{Deal}(\mathsf{pk}, \mathsf{sk})$

$\quad i = 0, 1, j = 1, \dots, \tau:$

$\qquad t_{i,j,k} \leftarrow \mathsf{Play}(\mathsf{sk}_{i,k}, c_j; \rho_{i,k,j})$

$w \leftarrow (\{\mathsf{aux}_k, \{t_{i,j,k}\}\})$

$\xrightarrow{w}$

Verifier$(\mathsf{pk}, \{c_j\}_{j=1}^{\tau}, \{m_j\}_{j=1}^{\tau})$

$\beta \xleftarrow{\$} \{0,1\}^{\lambda}$

$\xleftarrow{\beta}$

$z \leftarrow (\{\mathsf{sk}_{\beta[k],k}\}_k, \{\rho_{\beta[k],k,j}\}_{k,j})$

$\xrightarrow{z}$

$k = 1, \dots, \lambda:$

$\quad \mathsf{Verify}(\mathsf{pk}, \mathsf{aux}_k, \beta[k], \mathsf{sk}_{\beta[k],k}) \stackrel{?}{=} 1$

$\quad j = 1, \dots, \tau:$

$\qquad \mathsf{Play}(\mathsf{sk}_{\beta[k],k}, c_j; \rho_{\beta[k],k,j}) \stackrel{?}{=} t_{\beta[k],j,k}$

$\qquad \mathsf{Reconstruct}(c_j, t_{0,j,k}, t_{1,j,k}) \stackrel{?}{=} m_j$

Verifiable Decryption - MPC in the Head

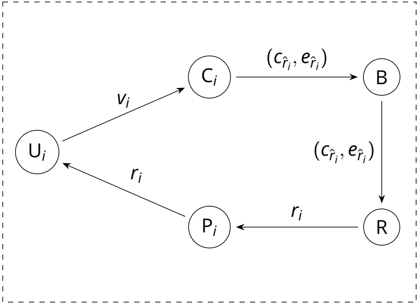
- Can run the protocol λ times for soundness $2^{-\lambda}$.
- Can choose security parameter κ such that $\kappa > \lambda$.
- Deal is dependent on λ not number of messages τ.
- The decryption proof is of total size $\approx 8\lambda\tau$ KB for τ messages.
- The decryption proof takes time $\approx 34\lambda\tau$ μs to compute for τ messages.

Electronic Voting - Setting

- We use a trusted printer to give users return codes.
- Each user have their own return-code-key $\hat{k}$.
- The return code server has a secret PRF-key k.
- We encrypt openings of commitments using verifiable encryption.
- Trusted election authorities EA verifies proofs and views.

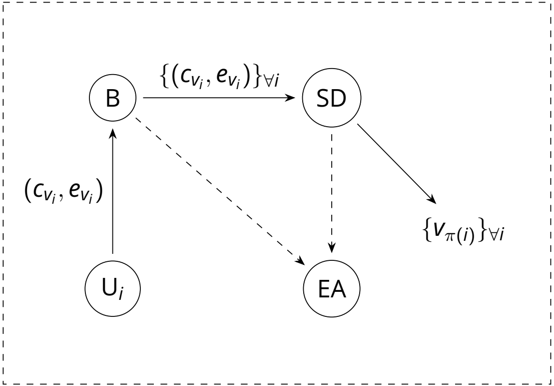
Electronic Voting - Return Codes

- $\hat{r}$ is a pre-code based on v_i and $\hat{k}$.
- r is the return code of k applied to $\hat{r}$.
- Integrity if C_i or P_i does not collude with R.
- Privacy if C_i, $\hat{r}$ and r does not leak the vote.



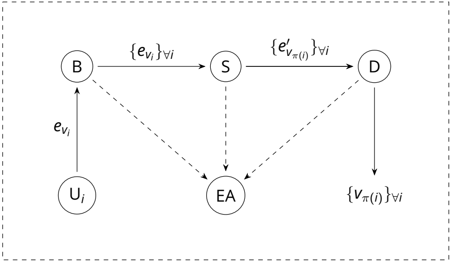
Electronic Voting - Verifiable Shuffle-Decryption

- SD both shuffle and decrypt the votes.
- Integrity follows from the ZK-proof.
- Privacy if B and SD does not collude.



Electronic Voting - Verifiable Mix-Net

- S may consist of many shuffle-servers.
- D may consist of many decryption-servers, or many key-shifting servers and only one decryption server.
- Integrity follows from the ZK-proofs.
- Privacy holds if either is true:
  1. at least one shuffle-server is honest, or
  2. at least one decryption-server is honest.



Thank you! Any questions?

NTNU | Norwegian University of Science and Technology

Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky.
Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits.
In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699. Springer, Heidelberg, August 2018.

Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert.
More efficient commitments from structured lattice assumptions.
In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Heidelberg, September 2018.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan.
(Leveled) fully homomorphic encryption without bootstrapping.
In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias.

Multiparty computation from somewhat homomorphic encryption.
In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.

C. Andrew Neff.
A verifiable secret shuffle and its application to e-voting.
In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.