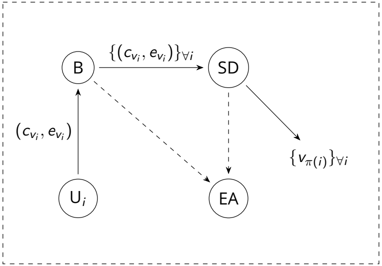
NTNU | Norwegian University of Science and Technology

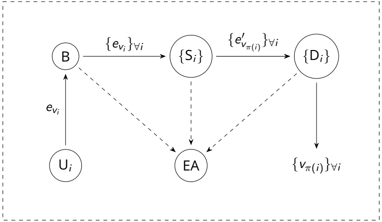
LATTICE-BASED VERIFIABLE SHUFFLE AND DECRYPTION

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

November 26, 2020



B
SD
$\{(c_{v_i}, e_{v_i})\}_{\forall i}$
(c_{v_i}, e_{v_i})
$\{v_{\pi(i)}\}_{\forall i}$
U_i
EA



B
$\{e_{v_i}\}_{\forall i}$
$\{S_i\}$
$\{e'_{v_{\pi(i)}}\}_{\forall i}$
$\{D_i\}$
e_{v_i}
U_i
EA
$\{v_{\pi(i)}\}_{\forall i}$

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. Combine everything to construct systems for electronic voting
5. Use primitives based on lattices to achieve post-quantum security

Preliminaries - Commitment

Algorithms:

$\texttt{Com}$: samples randomness $\boldsymbol{r}_m$ and commits to m as $[m] = \texttt{Com}(m; \boldsymbol{r}_m)$.

$\texttt{Open}$: takes as input $([m], m, \boldsymbol{r}_m)$ and verifies that $[m] \stackrel{?}{=} \texttt{Com}(m; \boldsymbol{r}_m)$.

Properties:

$\texttt{Binding}$: it is hard to find $m \neq \hat{m}$ and $\boldsymbol{r}_m \neq \hat{\boldsymbol{r}}_{\hat{m}}$ s.t. $\texttt{Com}(m; \boldsymbol{r}_m) = \texttt{Com}(\hat{m}; \hat{\boldsymbol{r}}_{\hat{m}})$.

$\texttt{Hiding}$: it is hard to distinguish $\texttt{Com}(m; \boldsymbol{r}_m)$ from $\texttt{Com}(0; \boldsymbol{r}_0)$ when given m.

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

Preliminaries - Proof of Linearity

Let

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

Then 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 proof of linearity see Baum et al. [BDL+18].

Preliminaries - Amortized Proof of Shortness

Let

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

where all are commitments to short values. Then the protocol Π_{A} is a sigma-protocol to prove that the underlying messages of $[x_1]$, $[x_2]$, ..., $[x_n]$ 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 $e \leftarrow \mathcal{N}_{\sigma_E}$.
The algorithm outputs $\mathtt{pk} = (a, b) = (a, as + pe)$ and $\mathtt{sk} = s$.

`Enc` samples a short $r \leftarrow R_q$ and noise $e_1, e_2 \leftarrow \mathcal{N}_{\sigma_E}$, 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 $i = 1, ..., \tau$.
- 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

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

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

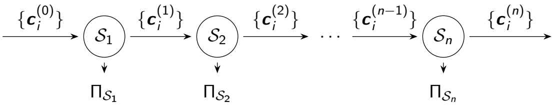
Prover, P		Verifier, V
	$\xleftarrow{\rho}$	$\rho \xleftarrow{\$} R_q \setminus \{\hat{m}_i\}_{i=1}^{\tau}$
$\hat{M}_i = \hat{m}_i - \rho$ $M_i = m_i - \rho$		$\hat{M}_i = \hat{m}_i - \rho$ $[M_i] = [m_i] - \rho$
$\theta_i \xleftarrow{\$} R_q, \forall i \in [\tau - 1]$ Compute $[D_i]$ as in Eq. (1), 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 $\sigma_C \approx 2^{15}$.
- 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 $\approx 18\tau$ 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 randomness is short
- Integrity holds because of the proofs
- Privacy if at least one server is honest



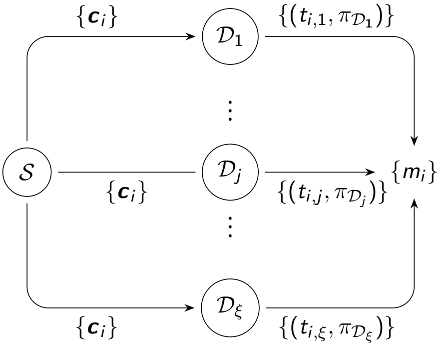
Verifiable Key-Shifting - Protocol

- We're given a ciphertext (u, v) under key s_1.
- We want the ciphertext (u', v') under key $s = s_1 + s_2$.
- 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 [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 - 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 verifies all shares.
5. V reconstructs to check the message from the shares.

Π_{ZKPCD}

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

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

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

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

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

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

$\xrightarrow{\quad w \quad}$

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

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

$\xleftarrow{\quad \beta \quad}$

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

$\xrightarrow{\quad z \quad}$

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

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

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

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

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

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 the 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.

Verifiable Decryption - One-Party Decryption

New: We can decrypt directly as following:

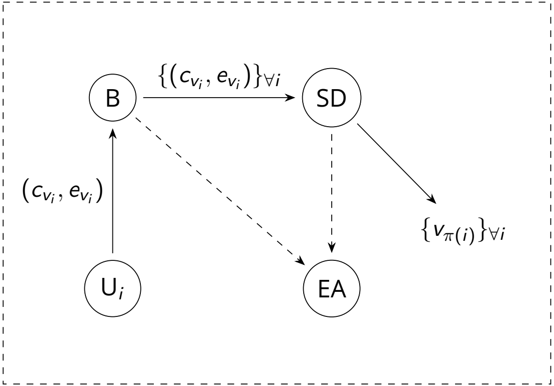
- Public commitment $[s]$ to secret key s.
- Compute $m_i \equiv (v_i - su_i \mod q) \mod p$.
- Commit to $d_i = v_i - su_i - m_i$ as $[d_i]$.
- Use Π_{Lin} to prove correct computation.
- Use Π_{A} to prove that each d_i is bounded.

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.

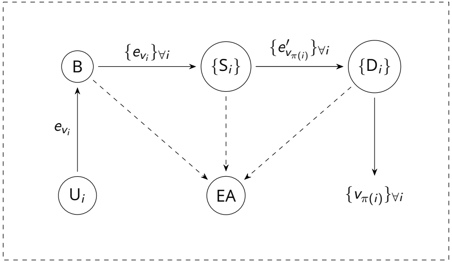
Electronic Voting - Verifiable Shuffle-Decryption

- 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 the following is true:
  1. at least one shuffle-server is honest, and
  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.