Multiple Secret Leaders

Authors here

July 4, 2023

1 Some Notation

1. We will have n parties
2. We will have k leaders elected
3. We will have a "bid" published by a user $i \in [n]$ be denoted as b_i
4. We will denote a commitment as $\texttt{comm}_i$
5. We will denote some generic CRHF as h
6. We will say $\texttt{Enc}_{\text{TFHE}}$ and $\texttt{Dec}_{\text{TFHE}}$ for TFHE encoding and decoding respectively

2 Preliminaries

2.1 Threshold FHE

[JRS17] defines threshold FHE encryption. For the sake of completeness, we will define it here.

Definition 2.1 (TFHE)**.** Let $P = \{ P_1, ..., P_N \}$ be a set of N parties and $\mathbb{S}$ be a class of access structures on P. A TFHE scheme for $\mathbb{S}$ is a tuple of PPT algorithms

$$(\texttt{TFHE.Setup}, \texttt{TFHE.Encrypt}, \texttt{TFHE.Eval}, \texttt{TFHE.PartDec}, \texttt{TFHE.FinDec})$$

such that the following specifications are met

- $(pk, sk_1, ..., sk_N) \leftarrow \texttt{TFHE.Setup}(1^\lambda, 1^s, \mathbb{A})$: Takes as input a security parameter λ, a depth bound on the circuit, and a structure $\mathbb{A} \in \mathbb{S}$. Outputs a public key pk and a secret key sk_i for each party P_i.
- $\texttt{ct} \leftarrow \texttt{TFHE.Encrypt}(pk, \mu)$: Takes as input a public key and a message $\mu \in \{ 0, 1 \}$ and outputs a ciphertext $\texttt{ct}$.
- $\hat{\texttt{ct}} \leftarrow \texttt{TFHE.Eval}(C, \texttt{ct}_1, ..., \texttt{ct}_k)$: Takes as input a circuit C of depth at most d and k ciphertexts $\texttt{ct}_1, ..., \texttt{ct}_k$. Outputs a ciphertext $\hat{\texttt{ct}} = C(\texttt{ct}_1, ..., \texttt{ct}_k)$.
- $p_i \leftarrow \texttt{TFHE.PartDec}(\texttt{ct}, sk_i)$: Takes as input a ciphertext $\texttt{ct}$ and a secret key sk_i and outputs a partial decryption p_i.
- $\hat{\mu} \leftarrow \texttt{TFHE.FinDec}(B)$: Takes as input a set $B = \{ p_i \}_{i \in S}$ for some $S \subseteq [N]$ and deterministically outputs a message $\hat{\mu} \in \{ 0, 1, \perp \}$.

Further we remember the definitions of evaluation correctness and simulation security as outlined in, [BEHG20].

Definition 2.2 (Evaluation Correctness [BEHG20])**.** . We have that $\texttt{TFHE}$ scheme is correct if for all λ, depth bounds d, access structure $\mathbb{A}$, circuit $C : \{0,1\}^k \to \{0,1\}$ of depth at most d, $S \in \mathbb{A}$, and $\mu_i \in \{0,1\}$, we have the following. For $(pk, sk_1, ..., sk_N) \leftarrow \texttt{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$, $\texttt{ct}_i \leftarrow \texttt{TFHE.Encrypt}(pk, \mu_i)$ for $i \in [k]$, $\hat{\texttt{ct}} \leftarrow \texttt{TFHE.Eval}(pk, C, \texttt{ct}_1, ..., \texttt{ct}_k)$,

$$\mathbf{Pr}\left[\texttt{TFHE.FinDec}(pk, \{\texttt{TFHE.PartDec}(pk, \hat{\texttt{ct}}, sk_i)\}_{i\in S}) = C(\mu_1, ..., \mu_k)\right] = 1 - \texttt{negl}(\lambda).$$

Definition 2.3 (Semantic Security)**.** We have that a $\texttt{TFHE}$ scheme satisfies semantic security for for all λ, and depth bound d if the following holds. There is a stateful PPT algorithm $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT adversary $\mathcal{A}$, the following experiment outputs 1 with negligible probability in λ:

1. On input 1^λ and depth 1^d, the adversary outputs $\mathbb{A} \in \mathbb{S}$
2. The challenger runs $(pk, sk_1, ..., sk_N) \leftarrow \texttt{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$ and provides pk to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a set $S \subseteq \{P_1, ..., P_N\}$ such that $\S \notin \mathbb{A}$.
4. The challenger provides $\{sk_i\}_{i\in S}$ and $\texttt{TFHE.Encrypt}(pk, \mu)$ to $\mathcal{A}$ where $\mu \xleftarrow{\$} \{0,1\}$.
5. $\mathcal{A}$ outputs a guess μ'. The experiment outputs 1 if $\mu' = \mu$.

2.2 Data Independent Priority Queue

In this work, we will use data independent queues as studied in [Tof11, MZ14, MDPB23]. Data independent data structures are unique as their control flow and memory access do not depend on input data ([MZ14]).

Definition 2.4 (Word RAM model [MZ14])**.** In the word RAM model, the RAM has a constant number of public and secret registers and can perform arbitrary operations on a constant number of registers in constant time.

Definition 2.5 (Data Independent Data Structure [MZ14])**.** In the word RAM model, a data independent data structure is a collection of algorithms where all the algorithms uses RAM such that the RAM can only set its control flow based on registers that are public.

Data independent queues are especially useful as they allow for efficient computation within MPC and FHE as control flow is not dependent on underlying ciphertexts data. We use a data independent queue as outlined in [MDPB23] which allows for

- `PQ.Insert`: Inserts a tag and value, (p, x) into `PQ` according to the tag's priority.
- `PQ.ExtractFront`: Removes and returns the (p, y) with highest tag priority.
- `PQ.Front`: Returns the (p, y) with highest tag priority without removing the element.

Moreover, we note that the order is stable. I.e. the first inserted among equal tagged elements has a higher priority.

2.3 Resevoir Sampling

Resevoir sampling is an online algorithm which allows for randomly selecting k elements from a stream of n elements while using $\tilde{O}(k)$ space. Algorithm R ([Vit85]) is a simple algorithm which relies on a priority queue:

- $\texttt{Resevoir.Insert}(\mu_i, e_i, \texttt{coin}_i)$ where μ_i is i-th item, e_i is independentally sampled randomness, and $\texttt{coin}_i$ is a random coin with probability $1/m$ of equaling 1.
  - If $i \leq k$, insert the item into the queue along with e_i as its tag via $\texttt{PQ.Insert}(e_i, \mu_i)$.
  - If $i > k$ and $\texttt{coin}_i = 1$, replace the smallest labeled item in the queue with the new item if the coin is 1.
  - If $i > k$ and $\texttt{coin}_i = 0$, do nothing.
- $\mu_{a_1}, \mu_{a_2}, ..., \mu_{a_k} \leftarrow \texttt{Resevoir.Output}()$ where $a_1, ..., a_k$ are a uniformly random ordered subset of $[n]$
  - Call $\texttt{PQ.ExtractFront}$ k times setting μ_{a_ℓ} to the ℓ-th call to $\texttt{PQ.ExtractFront}$ where $\ell \in [k]$.

3 MSLE Protocol

We use a similar notion of ideal functionality for a multi-secret leader election from the ideal functionality of single secret leader election of LS: CITE.

The MSLE functionality $\mathcal{F}_{\text{MSLE}}$: Initialize $E, R \leftarrow \emptyset, \leftarrow 0$. Fix some $k \in \mathbb{N}$ to denote the number of rounds. Upon receiving,

- $\texttt{register}$ from party P_i, set $R \leftarrow R \cup \{(i, n\}$, broadcast $(\texttt{register}, i)$ to all parties and set $n \leftarrow n + 1$
- $\texttt{elect}(eid, S)$ Elect k leaders from $S \subseteq R$ parties. If $|S| \geq k$ and eid was not requested before, randomly sample $W^{eid} \subseteq S$ where $|W^{eid}| = k$. Then, assign a random ordering to W^{eid} to get ordered set E^{eid}. Next, send $(\texttt{outcome}, eid, a)$ to P_j for all $E^{eid}_a = (j, \cdot)$ and $(\texttt{outcome}, eid, \bot)$ to P_i if $(i, \cdot) \notin E^{eid}$. Store $E \leftarrow E \bigcup \{E^{eid}\}$.
- $\texttt{reveal}(eid, \ell)$ from P_i: for $E_{eid} \in E$, if $i = E^{eid}_a$, broadcast $(\texttt{result}, eid, \ell, i)$. Otherwise, broadcast $(\texttt{rejected}, eid, \ell, i)$.

Figure 1: Description of the MSLE functionality, heavily based on the description of SSLE in LS: CITE Dario and CFG21

.

3.1 Simulation Security

To show simulation security, we will prove that, given a party's input and output in the ideal model, a simulator can simulate the distribution of the view in the protocol for the party. Note that because the protocol is deterministic, it suffices to prove simulation for only the party's output.

The MSLE Protocol π_{MSLE}: Initialize $n = 0$. Fix some $k \in \mathbb{N}$ to denote the number of rounds. Also, sample ; this will be the key FHE to the PRF.

- `initialize`
  - Set $n = 0$
  - Sample some secret $s_{\texttt{TFHE}}$ and publicly publicly publish $\texttt{Enc}_{\mathrm{TFHE}}(s_{\texttt{TFHE}})$. This will be the key to the PRF
- $\texttt{register}(\texttt{Enc}_{\mathrm{TFHE}}(c_i))$
  - If party P does not already have a TFHE share, create a TFHE share for P_i.
- $\texttt{elect}(eid, S, \texttt{Enc}_{\mathrm{TFHE}}(c_{S_1}), ..., \texttt{Enc}_{\mathrm{TFHE}}(c_{S_{|S|}}))$ where $c_i = \texttt{comm}(s_i)$ for secret s_i to party P_i
  - An encrypted streaming sampler will be publicly initialized
  - the CRS will be used to run a PRF to get $\texttt{Enc}_{\mathrm{TFHE}}(e_i) = \texttt{Enc}_{\mathrm{TFHE}}(\texttt{PRF}(c_i, s_{\texttt{TFHE}}))$ for $i \in S$
  - All $\texttt{Enc}_{\mathrm{TFHE}}(c_{S_i})$ will be fed to the encrypted streaming sampler along with randomness $\texttt{Enc}_{\mathrm{TFHE}}(e_i)$
  - The encrypted streaming sampler will output a list of k messages: $\texttt{Enc}_{\mathrm{TFHE}}(c_{a_1}), \texttt{Enc}_{\mathrm{TFHE}}(c_{a_2}), ..., \texttt{Enc}_{\mathrm{TFHE}}(c_{a_k})$.
  - Then, at least t parties will submit decryption shares to get $c_{a_1}, ..., c_{a_k}$.
  - Each party will then check if they won an election by seeing if their commitment is in the list of decrypted messages.
- $\texttt{reveal}(eid, \ell, \texttt{Enc}_{\mathrm{TFHE}}(c'_i))$ from P_i
  - P_i submits a proof that they know the opening to c_{a_ℓ}. If this proof verifies, send out $(\texttt{result}, eid, \ell, i)$ to all parties. Otherwise, send out $(\texttt{rejected}, eid, \ell, i)$ to all parties.

Figure 2: Description of the MSLE protocol

More formally we will show that for party i,

$$\mathrm{Sim}_i\left(s_i, r_i, \mathcal{F}_{\mathrm{MSLE}}.\mathtt{register}_i\right) \stackrel{c}{\equiv} \mathtt{view}_{\mathtt{register}_i}((s_0, r_0), ..., (s_n, r_n)), \tag{1}$$

$$\mathrm{Sim}_i\left(s_i, b_i, r_i, \mathcal{F}_{\mathrm{MSLE}}.\mathtt{elect}(eid, S)_i\right) \stackrel{c}{\equiv} \mathtt{view}_{\mathtt{elect}(eid,S)_i}((s_0, b_0), ..., (s_n, b_n)), \tag{2}$$

and

$$\mathrm{Sim}_i\left(b_{a,1}, \dots, b_{a,k}, s_i, r_i, \mathcal{F}_{\mathrm{MSLE}}.\mathtt{reveal}(eid, \ell)_i\right) \stackrel{c}{\equiv} \mathtt{view}_{\mathtt{reveal}(eid,\ell)_i}(), \tag{3}$$

Lemma 3.1. *We will first show that eq. (1) is simulation secure.*

Proof. The view of each party i for `register` can be expressed as

$$(r_i, s_i, c_i, C, \mathtt{Enc}_{\mathrm{TFHE}}(c_i), n)$$

We can create a simulator Sim_i that takes as input s_i, r_i, n and outputs an indistinguishable view

1. Sample something? Does the simulator have access to C?

□

Lemma 3.2. *We will now show that eq. (2) is simulation secure.*

Proof. The view of each party i for $\mathtt{elect}(eid, S)$ can be expressed as

$$(r_i, s_i, \mathtt{Enc}_{\mathrm{TFHE}}(b_1), \dots \mathtt{Enc}_{\mathrm{TFHE}}(b_n), \mathtt{Enc}_{\mathrm{TFHE}}(r'_1), \dots, \mathtt{Enc}_{\mathrm{TFHE}}(r'_n),$$
$$\mathtt{Enc}_{\mathrm{TFHE}}(b_{a_1}), \dots, \mathtt{Enc}_{\mathrm{TFHE}}(b_{a_k}), \sigma_1, \dots \sigma_t, b_1, \dots, b_n, y)$$

where r'_i is the randomness from the streaming sampler, $\sigma_1, \dots \sigma_t$ are the decryption shares, and $y \in \{\perp, 1, ..., k\}$ representing whether a party won election 1 through k or not ($\perp$). Then, we have the simulator proceed in the following manner:

1. The simulator sets a random, local tape
2. The simulator samples c'_j for all $j \neq i$ where c'_j is a commitment to a random value
3. The simulator samples some randomness r and computes $\mathtt{Enc}_{\mathrm{TFHE}}(e_1), \dots, \mathtt{Enc}_{\mathrm{TFHE}}(e_n) = \mathtt{Enc}_{\mathrm{TFHE}}(\mathtt{PRF}(r, s_{\mathtt{TFHE}}))$
4. The simulator creates an encrypted priority queue `EncPQ` and simulates the encrypted streaming sampler for all inputs $\mathtt{Enc}_{\mathrm{TFHE}}(c'_j)$ for $j \in [i]$.
5. The simulator runs the encrypted streaming sampler to get the outputs $\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_1}), ...\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_k})$.
6. The simulator then chooses a random, ordered subset $S \subseteq [n]$ where $|S| = k$. If there is some w such that $S_w = i$, then set $y' = w$. Otherwise, set $y' = \perp$.
7. The simulator then sets the decryptions of $\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_\ell})$ to c'_{S_ℓ}. The simulator also creates shares σ_j such that $\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_\ell})$ decrypts to c'_{S_ℓ}.
8. The simulator then outputs y'

We now use a sequence of hybrids to show that the view of the real protocol is indistinguishable from that of the simulated one

- $\mathbf{Hyb}_0$: The real protocol
- $\mathbf{Hyb}_1$: As $\mathbf{Hyb}_0$ but, for all $j \neq i$, $\mathtt{Enc}_{\mathrm{TFHE}}(c_j)$ are replaced with $\mathtt{Enc}_{\mathrm{TFHE}}(c'_j)$, where c'_j is a commitment to a random value. We can see that $\mathbf{Hyb}_0 \equiv \mathbf{Hyb}_1$ by the hiding property of commitments.
- $\mathbf{Hyb}_2$: As $\mathbf{Hyb}_1$ but replace $\mathtt{Enc}_{\mathrm{TFHE}}(c_{a_1}), ...\mathtt{Enc}_{\mathrm{TFHE}}(c_{a_k})$ with $\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_1}), ...\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_k})$, the output of sampling the encrypted streaming sampler with $\mathtt{Enc}_{\mathrm{TFHE}}(c'_j)$.
- $\mathbf{Hyb}_3$: As $\mathbf{Hyb}_2$ but replace $\mathtt{Dec}_{\mathrm{TFHE}}(\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_1})), ...\mathtt{Dec}_{\mathrm{TFHE}}(\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_k}))$ with $c'_{S_1}, ...c'_{S_k}$, where S is the random subset chosen by the simulator. Replace σ_j with shares σ'_j such that $\mathtt{Enc}_{\mathrm{TFHE}}(c'_{a_\ell})$ decrypts to c'_{S_ℓ}.
- $\mathbf{Hyb}_4$: As $\mathbf{Hyb}_3$ but replace y with y'.
- $\mathbf{Hyb}_5$: The simulated protocol

□

Lemma 3.3. *We will now show that eq. (3) is simulation secure.*

Proof. □

References

[BEHG20] Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 12–24, 2020. 2.1, 2.2

[JRS17] Aayush Jain, Peter MR Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *Cryptology ePrint Archive*, 2017. 2.1

[MDPB23] Sahar Mazloom, Benjamin E Diamond, Antigoni Polychroniadou, and Tucker Balch. An efficient data-independent priority queue and its application to dark pools. *Proceedings on Privacy Enhancing Technologies*, 2:5–22, 2023. 2.2, 2.2

[MZ14] John C Mitchell and Joe Zimmerman. Data-oblivious data structures. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014. 2.2, 2.4, 2.5

[Tof11] Tomas Toft. Secure data structures based on multi-party computation. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 291–292, 2011. 2.2

[Vit85] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 2.3