CSC 2419: Lattice-based Cryptography **Fall 2025**

Lecture 5: Rate-1 FHE

Instructor: Akshayaram Srinivasan *Scribe: Nikolay Avramov*

Date: 2025-10-06

5.1 Recap

In the previous lecture, we discussed digital signatures and how to construct them from the Learning With Errors (LWE) assumption. This illustrated how LWE serves as a fundamental building block for both encryption and authentication primitives.

We then examined the GSW construction of a bounded Fully Homomorphic Encryption (FHE) scheme, which supports evaluation of circuits up to a bounded multiplicative depth of n^ϵ. To overcome these limitations, we showed how bootstrapping can be used to convert a bounded FHE scheme into an unbounded one.

5.2 Rate-1 FHE

Previously, we saw how to construct an FHE scheme that can evaluate arbitrary Boolean functions $f : \{0,1\}^\ell \to \{0,1\}$. A natural next step is to extend this to multi-bit functions $f : \{0,1\}^\ell \to \{0,1\}^m$, but doing so naively leads to prohibitively large communication costs.

Suppose a client wishes to offload computation on encrypted data to a server. In the single-bit case, the server can evaluate f and return one ciphertext of size $\text{poly}(n)$. Extending this directly to the m-bit case by decomposing f into m single-bit functions would require sending m separate ciphertexts, each of size $\text{poly}(n)$—a total of $m \cdot \text{poly}(n)$ communication, which quickly becomes impractical.

Our goal is to extend the single-bit construction to the m-bit setting while keeping the communication overhead negligible. Specifically, the trivial insecure solution requires m bits of communication from the server to enable the client to learn the output. We would like to have the overhead of achieving security to approach 1 as the size of the message m grows, giving what we call a *rate-1* FHE scheme. In other words, we require the size of the ciphertexts encrypting an m-bit output to be $m + \text{poly}(n)$.

5.2.1 Definition of Rate-1 FHE

Recall the GSW encryption scheme [GSW13]:

$$\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \qquad \mathbf{s} \leftarrow \mathbb{Z}_q^n, \qquad \mathbf{e} \leftarrow \chi^m, \qquad m \geq 2n \log q$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{A} \\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top \end{bmatrix} \in \mathbb{Z}_q^{(n+1) \times m}, \qquad \mathbf{t} = \begin{bmatrix} -\mathbf{s} \\ 1 \end{bmatrix} \in \mathbb{Z}_q^{n+1}.$$

Encryption of a bit $x \in \{0,1\}$ looks like

$$\mathsf{ct} = \mathbf{C} \cdot \mathbf{R} + x \cdot \mathbf{G} \in \mathbb{Z}_q^{(n+1)\times(n+1)\log q}$$

where

$$\mathbf{R} \leftarrow \{0,1\}^{m\times(n+1)\log q}$$
$$\mathbf{G} = \mathbf{I}_{n+1} \otimes \mathbf{g}$$
$$\mathbf{g} = [1 \quad 2 \ldots \quad 2^{\log q - 1}]$$

Homomorphic evaluation over Boolean Functions. Since the GSW scheme is fully homomorphic, we can evaluate any Boolean function
$f : \{0,1\}^\ell \to \{0,1\}$ on encrypted bits $x_1, \ldots, x_\ell$ with ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$. The resulting ciphertext is

$$\mathrm{Eval}(\mathrm{pk}, f; \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) = \mathbf{C} \cdot \mathbf{R}' + f(x_1, \ldots, x_\ell) \cdot \mathbf{G}.$$

Homomorphic evaluation over Multi-output Functions. To compute a function $f : \{0,1\}^\ell \to \{0,1\}^m$, decompose it into m boolean functions $f_1, \ldots, f_m$, where $f_i : \{0,1\}^\ell \to \{0,1\}$ returns the i^{th} bit of output of function f i.e. $f_i(\mathbf{x}) = f(\mathbf{x})[i]$. Homomorphically evaluating the boolean functions over GSW ciphertexts produces m ciphertext $\mathsf{ct}_m \in \mathbb{Z}_q^{(n+1)\times(n+1)\log q}$. Naively, evaluating all m outputs requires $O(m \cdot \mathrm{poly}(n))$ communication, resulting in a multiplicative overhead.

Goal: Rate-1 FHE. We would like to achieve additive rather than multiplicative communication overhead; that is, total communication $m + \mathrm{poly}(n)$ instead of $m \cdot \mathrm{poly}(n)$. This results in the ciphertext overhead approaching 1 as the size of m grows.

5.3 Trapdoor Hash for Linear Predicates

We discuss an important primitive we will use as a building block toward rate-1 FHE encryption scheme.

Suppose two parties (Alice and Bob) represent the Server and Client, respectively. Alice and Bob have $\mathbf{x}, \mathbf{y} \in \{0,1\}^m$ and wish to output bits $v_1, v_2 \in \{0,1\}$ such that

$$v_1 \oplus v_2 \equiv \langle \mathbf{x}, \mathbf{y} \rangle \pmod 2 \tag{5.1}$$

with Alice's communication independent of m. The obvious solution is for Bob to reveal $\mathbf{y}$. Alice would set $v_2 = \langle \mathbf{x}, \mathbf{y} \rangle \pmod 2$, and Bob would set $v_1 = 0$. The shares would trivially satisfy Equation 5.1. Instead, we require that Alice learn no information about the other party's input. This is the precise functionality realised by the Trapdoor Hash Function (TDH) primitive introduced in [DGI+19] for evaluating Linear Predicates.

5.3.1 (Noisy) TDH for Linear Predicates

In [DGI+19], the authors introduces Trapdoor Hash Functions for linear predicates, where the Hasher (Alice) and Encoder (Bob) have inputs $\mathbf{x}, \mathbf{y} \in \{0,1\}^m$ such that they want to learn "noisy" shares (v_2, v_1) of $\langle \mathbf{x}, \mathbf{y} \rangle \frac{q}{2} \pmod 2$ over $\mathbb{Z}_q$ i.e.

$$v_2 - v_1 = \langle \mathbf{x}, \mathbf{y} \rangle \frac{q}{2} + \mathbf{e}$$

where $\mathbf{e}$ is a low-norm "noise". Additionally, the Hasher learns no information about Encoder's input.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and return $\mathsf{crs} := \mathbf{A}$.
- $\mathsf{Hash}(\mathsf{crs}, \mathbf{x}) \to (\mathbf{h}, \mathsf{td}_h)$: The hashing Algorithm essentially computes the SIS hash with respect to the public matrix $\mathbf{A}$ i.e. it computes $\mathbf{h} = \mathbf{A}\mathbf{x} \pmod q$ and $\mathsf{td}_h = \mathbf{x}$.
- $\mathsf{Enc}(\mathsf{crs}, \mathbf{y}) \to (\mathbf{E}, \mathsf{td}_e)$: The encoding algorithm essentially computes a bitwise Regev encryption of input $\mathbf{y}$. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^m$, and compute

  $$\mathbf{u}^\top = \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top + \mathbf{y}^\top \cdot \frac{q}{2} \in \mathbb{Z}_q^m.$$

  It sets $\mathbf{E} := \mathbf{u}^\top$ and trapdoor $\mathsf{td}_e := \mathbf{s}$
- $\mathsf{HashEval}(\mathsf{crs}, \mathbf{E}, \mathsf{td}_h) \to v_2$: The hash evaluation algorithm computes $v_2 = \mathbf{u}^\top \mathbf{x} = \mathbf{s}^\top \mathbf{A}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \langle \mathbf{y}, \mathbf{x} \rangle \frac{q}{2}$
- $\mathsf{EncEval}(\mathsf{crs}, \mathbf{h}, \mathsf{td}_e) \to v_1$: The encoding evaluation algorithm computes $v_1 = \mathbf{s}^\top \mathbf{h} = \mathbf{s}^\top \mathbf{A}\mathbf{x}$

Here, the tuple (v_1, v_2) is a noisy additive sharing of $\langle \mathbf{x}, \mathbf{y} \rangle \frac{q}{2}$ over $\mathbb{Z}_q$, since

$$v_2 - v_1 = \mathbf{e}' + \langle \mathbf{y}, \mathbf{x} \rangle \frac{q}{2}.$$

where $\mathbf{e}' = \mathbf{e}^\top \mathbf{x}$ and $||\mathbf{e}'||_\infty \leq n||\mathbf{e}||_\infty$.

The protocol proceeds as follows:

1. Alice computes $(\mathbf{h}, \mathsf{td}_h) \leftarrow \mathsf{Hash}(\mathsf{crs}, \mathbf{x})$ and Bob computes $(\mathbf{E}, \mathsf{td}_e) \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathbf{y})$. Alice sends $\mathbf{h}$ to Bob and receives $\mathbf{E}$ from Bob.
2. Alice and Bob evaluate $v_2 \leftarrow \mathsf{HashEval}(\mathsf{crs}, \mathbf{E}, \mathsf{td}_h)$ and $v_1 \leftarrow \mathsf{EncEval}(\mathsf{crs}, \mathbf{h}, \mathsf{td}_e)$ respectively.

Communication cost. Alice sends one vector $\mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$, i.e., $n \log q$ bits (independent of m).

Security (sketch). Bob's message $\mathbf{E}$ is the Regev Encryption of input $\mathbf{y}$. Invoking the semantic security of the encryption scheme(based on the hardness of LWE), we can conclude that Alice learns no information about Bob's input.

5.3.2 Spooky Rounding

Up until now, Alice and Bob learns "noise" shares of $\langle \mathbf{x}, \mathbf{y} \rangle \frac{q}{2}$ over $\mathbb{Z}_q$. In this section, we use the "Spooky Rounding" technique introduced in [DHRW16], which can be used to translate the "noisy" shares to shares of $\langle \mathbf{x}, \mathbf{y} \rangle$ over $\mathbb{Z}_2$.

Lemma 5.1 (Spooky rounding[DHRW16], adapted) *Let p, q be modulus with $v_1 \leftarrow \mathbb{Z}_q$ be sampled uniformly and let $v_2 = v_1 + \mu \cdot \frac{q}{p} + e$ for $\mu \in \mathbb{Z}_p$ and $|e| \leq B$. Define $\mathsf{Round}_p(z)$ to round z to the nearest multiple of $\frac{q}{p}$ and output z. Then*

$$\Pr[\mathsf{Round}_p(v_2) \oplus \mathsf{Round}_p(v_1) \neq \mu \pmod p] \leq O(pB/q).$$

Proof (Sketch). We present an overview of the proof sketch for $p = 2$. Similar ideas can be extended for arbitrary p.

When $b = 0$, we have $v_2 = v_1 + e$. An error occurs only if $v_1 < \frac{q}{2}$ and $v_2 \geq \frac{q}{2}$ i.e. $v_1 \in [\frac{q}{2} - B, \frac{q}{2}]$. Since v_1 is uniformly distributed, this happens with probability $2B/q$.

For $b = 1$, the error occurs iff $v_1 \in [q - B, q)$, which occurs with probability $2B/q$.

Application to TDH for linear predicates It should be noted that in order to apply "spooky" rounding technique on the "noise" shares of TDH, we need to ensure that the share v_1 is distributed uniformly. Therefore, we can add a publicly sampled random offset u to v_1 and v_2 i.e. let

$$v_1 = \mathbf{s}^\top \mathbf{A}\mathbf{x} + u, \qquad v_2 = \mathbf{s}^\top \mathbf{A}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \langle \mathbf{x}, \mathbf{y} \rangle \frac{q}{2} + u.$$

Each party computes its local v_i and then outputs the bit $b_i = \mathsf{Round}_2(v_i)$. By the definition above, $b_1 \oplus b_2 = \langle \mathbf{x}, \mathbf{y} \rangle$ except with probability $O(B/q)$. Choosing $q \gg B$ makes this error negligible.

5.4 Extension to $\mathbb{Z}_p$

So far we showed a TDH for linear predicates over $\mathbb{Z}_2$ Our next goal is to extend this primitive to the case where $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^m$ while preserving communication cost and security guarantees. The idea is simple: the encoder scales its input $\mathbf{y}^\top$ with $\frac{q}{p}$ instead of $\frac{q}{2}$ and invoke "spooky" rounding (Lemma 5.1) which returns values over $\mathbb{Z}_p$ instead of $\mathbb{Z}_2$

5.4.1 Noisy Protocol

Setting. Fix $p \geq 2$. Pick a large modulus q that is a multiple of p, e.g.,

$$q \;=\; p \cdot n \cdot \lambda^{\omega(1)} \quad \text{so that} \quad \frac{q}{p} \gg n \cdot B,$$

where B is a bound on the error sampled by Encoder. Sample the CRS $\mathsf{crs} \leftarrow (\mathbf{A}, \mathbf{u})$, where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and a public mask $u \leftarrow \mathbb{Z}_q$

Goal. Given $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^m$, Alice and Bob output $\hat{v}_1, \hat{v}_2 \in \mathbb{Z}_p$ such that

$$\hat{v}_2 - \hat{v}_1 \equiv \mathbf{x}^\top \mathbf{y} \pmod{p}.$$

Protocol. Proceeds in an identical fashion as in Section 5.3.1, with $\frac{q}{2}$ replaced with $\frac{q}{p}$. Alice and Bob return $\hat{v_1} := \mathsf{Round}_p(v_1)$

Security. Exactly as in the bit-vector case, Bob's message is a Regev ciphertext:

$$(\mathbf{A},\ \mathbf{u}^\top) \;=\; (\mathbf{A},\ \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top + \frac{q}{p}\,\mathbf{y}^\top).$$

and security follows from the LWE assumption.

5.5 Matrix–Vector Product Extension

Motivation. For our rate-1 FHE construction we need *several* inner products with the same vector $\mathbf{x}$. Rather than rerun the vector protocol k times, we let Bob hold a matrix $\mathbf{M} \in \mathbb{Z}_p^{k\times m}$ whose rows are represented by $\mathbf{m}_1^\top, \mathbf{m}_2^\top, \ldots, \mathbf{m}_k^\top \in \mathbb{Z}_p^m$. The vector product $\mathbf{Mx}$ is essentially the dot product of row vector $\mathbf{m}_i^\top$ and vector $\mathbf{x}$. This extension was proposed by Abram et. al.[ARS24].

Protocol.

- The Setup phase is identical to the protocol presented above.
- Let $\mathbf{M} \in \mathbb{Z}_p^{k\times m}$ and $\mathbf{x} \in \mathbb{Z}_p^m$. Bob reuses the same secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ across multiple instantiations of TDH (one corresponding to each row of $\mathbf{M}$, but samples independent noise $\mathbf{e}_r \leftarrow \chi^m$ for $r \in [k]$. i.e. Bob generates encoding of input $\mathbf{U} \in \mathbb{Z}_q^{k\times m}$ with rows

$$\mathbf{u}_r^\top \;=\; \mathbf{s}^\top\mathbf{A} \;+\; (\mathbf{e}_r)^\top \;+\; (\mathbf{m}_r)^\top \cdot \frac{q}{p}.$$

  Alice computes her hash $\mathbf{h} := \mathbf{Ax}$ in a similar manner as before.
- Bob invokes $\mathsf{EncEval}()$ function as described in the previous construction to get $v_1 = \mathbf{s}^T\mathbf{Ax}$. Alice computes $\mathbf{Ux}$ to obtain all row-wise shares at once,

$$\mathbf{v}_2 = \mathbf{Ux} \;=\; \underbrace{(\mathbf{s}^\top\mathbf{Ax})}_{v_1} \;+\; \begin{bmatrix}\mathbf{e}_1^\top\mathbf{x}\\ \vdots \\ \mathbf{e}_k^\top\mathbf{x}\end{bmatrix} \;+\; (\mathbf{Mx})\cdot\frac{q}{p}.$$

Note that v_1 and $\mathbf{v}_2^{(r)}$ are the "noise" secret shares of $\mathbf{m}_r^\top\mathbf{x}\cdot\frac{q}{p}$ over $\mathbb{Z}_q$,i.e. ,

$$\mathbf{v}_2^{(r)} \;=\; \mathbf{u}_r^\top\mathbf{x} \;=\; v_1 \;+\; \mathbf{e}_r^\top\mathbf{x} \;+\; \mathbf{m}_r^\top\mathbf{x}\cdot\frac{q}{p},$$

so

$$v_2^{(r)} - v_1 \;=\; (\mathbf{e}_r)^\top\mathbf{x} \;+\; \mathbf{m}_r^\top\mathbf{x}\cdot\frac{q}{p}$$

This is exactly the same noisy inner-product relation as in the vector protocol

Security. Bob encodes $\mathbf{M}$ as in the single-vector case: each row is a separate Regev encryption, so the security follows semantic security of Regev encryption.

5.6 Rate-1 FHE Protocol

In the following section, assume that Bob plays the role of client and Alice plays the role of server.

Recall that evaluating $f : \{0,1\}^\ell \to \{0,1\}^m$ on GSW ciphertexts yield m ciphertexts corresponding to m output bits

$$\mathbf{D}^{(j)} \;=\; \mathbf{C}\,\mathbf{R}'_j \;+\; f_j(\mathbf{x})\,\mathbf{G}, \qquad j = 1,\ldots,m,$$

the decryption procedure computes $\mathbf{t}^\top \mathbf{D}^{(j)}$ and recover the i^{th} output bit (via "spooky" rounding technique), where $\mathbf{t} = (-\mathbf{s}, 1)$ is the secret key used in the encryption scheme. The decryption procedure computes

$$\mathbf{t}^\top \mathbf{D}^{(j)} \;=\; \mathbf{e}^\top \mathbf{R}'_j \;+\; y_j \cdot (-\mathbf{s}, 1)^\top \mathbf{G} \tag{5.2}$$

where $y_j = f_j(\mathbf{x})$, and recovers the output using *spooky rounding* technique.

High Level Idea. After receiving GSW ciphertexts $\{\mathsf{ct}_i\}_{i\in\ell}$ where ct_i is the GSW ciphertext corresponding to the message $\mathbf{x}^{(i)}$, if Alice (server) were to homomorphically evaluate the function f over the ciphertexts and send the output ciphertexts to Bob (client), the communication would be $m \cdot \mathrm{poly}(n)$. Instead, we observe that the decryption procedure is a linear operation over the ciphertext and the secret key, followed by "spooky" rounding. Exploiting this *almost-linear* decryption property of the encryption scheme, Alice and Bob invoke the TDH primitive to perform *shared decryption*: Alice and Bob jointly obtain the decryption values without Bob sending ciphertexts individually.

Here, Alice's input is the matrix $\mathbf{D} \;=\; [\mathbf{D}^{(1)} \;||\; \mathbf{D}^{(2)} \;||\; \cdots \;||\; \mathbf{D}^{(m)}] \;\in\; \mathbb{Z}_q^{(n+1)\times(m(n+1)\log q)}$ where $\mathbf{D}^{(i)}$ is the ciphertext obtained after homomorphically evaluating input ciphertexts over function $f_i()$. Bob' input is essentially the secret key used to encrypt the input ciphertexts i.e. Bob's input is $\mathbf{t}^\top = [-\mathbf{s}^\top \quad 1]$. Bob's encoding for the TDH primitive is the Regev encryption of the secret key message $\mathbf{t}^\top$ under the secret key $\mathbf{s}$ itself (*circular encryption)*, which can be used across multiple instance of TDH. Therefore, per evaluation the communication is $\mathrm{poly}(n)$ overhead from Alice plus the final m outputs, i.e., $m + \mathrm{poly}(n)$.

Alice's input. Alice has m ciphertexts $\mathbf{D}^{(i)}$ obtained after homomorphically evaluating input ciphertexts received from Bob (client). Concatenate the blocks into one matrix

$$\mathbf{D} \;=\; [\mathbf{D}^{(1)} \;||\; \mathbf{D}^{(2)} \;||\; \cdots \;||\; \mathbf{D}^{(m)}] \;\in\; \mathbb{Z}_q^{(n+1)\times(m(n+1)\log q)},$$

Let $\mathbf{d}_i^\top$ be the i^{th} row of the matrix $\mathbf{D}$

Bob's input Let $t_i \in \mathbb{Z}_p$ be the i-th entry of $\mathbf{t}$. For row i, we need to multiply each entry with the same scalar t_i. To realise this computation, Bob sets its i^{th} input $\mathbf{T}_i$ as follows:

$$\mathbf{T}_i \;=\; \begin{bmatrix} t_i & 0 & \cdots & 0 \\ 0 & t_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t_i \end{bmatrix} \in \mathbb{Z}_q^{(m(n+1)\log q)\times(m(n+1)\log q)} \tag{5.3}$$

Multiplying $\mathbf{T}_i$ with $\mathbf{d}_i$, we get:

$$\mathbf{u}_i^\top = \mathbf{d}^\top \mathbf{T}_i; = \; [t_i d_{i,1} \quad t_i d_{i,2} \quad \ldots \quad t_i d_{i,m(n+1)\log q}] \in \mathbb{Z}_p^{m(n+1)\log q}.$$

We note that computing $\mathbf{u} = \sum_{i=1}^{n+1} \mathbf{u}_i^\top$ essentially gives us the desired linear product

$$\mathbf{u} \;=\; \left[\mathbf{t}^\top \mathbf{D}^{(1)} \| \mathbf{t}^\top \mathbf{D}^{(2)} \| \cdots \| \mathbf{t}^\top \mathbf{D}^{(j)}\right]$$

Therefore, if Alice and Bob obtain secret shares of $\mathbf{u}$, they can perform "spooky" rounding to obtain shares of the decrypted value. We can exploit this observation to define a rate-1 FHE scheme.

Rate-1 FHE scheme The scheme proceeds as follows:

- **Bob:**
  1. Computes GSW ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ corresponding to its input $\mathbf{x} \in \{0,1\}^\ell$ using the secret key $\mathbf{s}$. Consider the corresponding secret key vector $\mathbf{t}^\top = [\mathbf{s}^\top \quad 1]$.
  2. Compute the encoding of $(n+1)$ instance of Trapdoor Hash primitive with the input to the i^{th} instance being the matrix $\mathbf{T}_i$ as in Equation 5.3. Let the encoding be $\mathbf{E}_i$. Bob sends the ciphertexts $\{\mathsf{ct}_i\}$ along with the $n+1$ encodings $\{\mathbf{E}_j\}_{j \in n+1}$ to Alice
- **Alice**: On receiving GSW ciphertexts, perform the following computation
  1. Homomorphically evaluate the output ciphertexts $\mathbf{D}^{(1)}, \mathbf{D}^{(2)}, \ldots \mathbf{D}^{(m)}$ and set
  $$\mathbf{D} = [\mathbf{D}^{(1)} \mid\mid \mathbf{D}^{(2)} \mid\mid \cdots \mid\mid \mathbf{D}^{(m)}]$$
  2. Decomposed $\mathbf{D}$ into $n+1$ rows $\mathbf{d}_1^\top, \ldots, \mathbf{d}_{n+1}^\top$ and use these as vector input for $(n+1)$ instances of Trapdoor Hash scheme. Let the hash computed for the i^{th} instance be $\mathbf{h}_i$.
  3. For each $i \in [n+1]$, she computes her hash value $\mathbf{h}_i$ with input being $\mathbf{d}_i^\top$ using the *Matrix-Vector Product Extension* of Trapdoor Hash.
  4. Alice computes share of $\mathbf{u}_i^\top$ using the $\mathsf{HashEval}()$ procedure on Bob's encoding $\mathbf{E}_i$.
  5. Alice takes the sum shares of $\mathbf{u}_i^\top$ to get share of $\left[\mathbf{t}^\top\mathbf{D}^{(1)} \| \mathbf{t}^\top\mathbf{D}^{(2)} \| \cdots \| \mathbf{t}^\top\mathbf{D}^{(j)}\right]$.
  6. For each $j \in [m]$, she isolates the share u_j corresponding to the last entry of $\mathbf{t}^\top \cdot \mathbf{D}^{(j)}$ and performs spooky rounding operation on the share to obtain $\alpha_j = \mathsf{Round}_2(u_j)$.
  7. She sends the hash values $\{\mathbf{h}_i\}$ along with shares $\{\alpha_i\}$ to Bob.

Decryption. Bob on input the hash values $\{\mathbf{h}_i\}$ along with shares $\{\alpha_i\}$ does the following:

1. For each $i \in [n+1]$, he computes his share of $\mathbf{u}_i^\top$ by evaluating the function $\mathsf{EncEval}()$ on the hash value $\mathbf{h}_i$ to get "noisy" share of $\mathbf{u}_i^\top$.
2. Bob sums his shares to obtain share of $\left[\mathbf{t}^\top\mathbf{D}^{(1)} \| \mathbf{t}^\top\mathbf{D}^{(2)} \| \cdots \| \mathbf{t}^\top\mathbf{D}^{(j)}\right]$.
3. For each $j \in [m]$, he isolates his share v_j corresponding to the last entry of $\mathbf{t}^\top \cdot \mathbf{D}^{(j)}$ and perform *spooky* rounding to obtain $\beta_j = \mathsf{Round}_2(v_j)$.
4. For each $j \in [m]$, Bob returns $y_j = \alpha_j \oplus \beta_j$.

Correctness. From the *almost-linear* decryption property of GSW encryption scheme (Equation 5.2) and the definition of gadget matrix $\mathbf{G}$, we realise that the last entry of $\mathbf{t}^\top \cdot \mathbf{D}^{(j)}$ is the value $\mathbf{e}^\top + y_j \frac{p}{2}$. Therefore, the shares (u_i, v_i) obtained by Alice and Bob are shares of $\mathbf{e}^\top + y_j \frac{p}{2}$. Applying *spooky rounding* techniques on these shares results in shares of the value y_i over $\mathbb{Z}_2$. Furthermore, from the correctness of homomorphic evaluation over GSW ciphertexts, we conclude that $y_i = f_i(\mathbf{x})$. Therefore, Bob decrypts to the correct value with high probability.

Communication. Alice's per-row communication is $n \log q$ bits. Thus the overall communication per evaluation is $m + \mathrm{poly}(n)$ (function description and Alice's outputs), achieving the Rate-1 target.

Security. The security of Bob's secret key comes from the security of the two-party protocol as discussed in the previous sections. Once we have used the security of two-party protocol, we can rely on the security of GSW encryption scheme.

References

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part VI*, page 301–330, Berlin, Heidelberg, 2024. Springer-Verlag.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2019.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 93–122, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. Cryptology ePrint Archive, Paper 2013/340, 2013.