6.875: Cryptography and Cryptanalysis **November 22, 2021**

Problem Set 6

Instructor: Vinod Vaikuntanathan

TAs: Lali Devadas, Aparna Gupte, Sacha Servan-Schreiber

Instructions.

- **When:** This problem set is due on **December 8, 2021** before **11pm ET**.
- **How:** You should use LaTeX to type up your solutions (you can use our LaTeX template from the course webpage). Solutions should be uploaded on Gradescope as a single pdf file.
- **Acknowledge your collaborators:** Collaboration is permitted and encouraged in small groups of at most three. You must write up your solutions *entirely on your own* and *acknowledge your collaborators*.
- **Reference your sources:** If you use material from outside the lectures, you must reference your sources (papers, websites, wikipedia, ...).
- **When in doubt, ask questions:** Use Piazza or the TA office hours for questions about the problem set. See the course webpage for the timings.

Problem 1. Sharing secrets. Recall Shamir's secret-sharing scheme presented in class [Sha79]. Let $\mathsf{Interpolate}$ be the polynomial interpolation function (e.g., Lagrange) which given t distinct points on a degree-$(t-1)$ polynomial P, outputs the polynomial P. The scheme is described in Figure 1. We let Share be parameterized by integers $1 \leq t \leq n$ such that any t out of n distinct shares can recover the secret s. You can assume that $p > n$ is prime.

$\mathsf{Share}_{\mathbb{Z}_p,n,t}(s)$

1: $a_1, \ldots, a_{t-1} \xleftarrow{R} \mathbb{Z}_p$
2: $P(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + s \bmod p$
3: **return** $P(1), P(2), \ldots, P(n)$

$\mathsf{Recover}_{\mathbb{Z}_p}(z_1, \ldots, z_t)$

1: $P \leftarrow \mathsf{Interpolate}(z_1, \ldots, z_t)$
2: **return** $P(0)$

Figure 1: Shamir's Secret-sharing Scheme

Suppose we want to share a *vector* of k secrets $\mathbf{s} \in \mathbb{Z}^k$. A naïve idea is to apply Shamir's scheme independently k times so that the i^{th} party obtains a vector of shares $(P_1(i), \ldots, P_k(i))$. However, this results in each party's share being of size $k \cdot \log p$ bits.

(a) **Design a secret-sharing scheme such that (1) sharing a secret vector $\mathbf{s} \in \mathbb{Z}_p^k$ results in each share being *a single* $\mathbb{Z}_p$ element, (2) no $t-1$ shares reveal any information on the secret, and (3) any $t+k$ or more shares can be used to recover the entire secret vector $\mathbf{s}$.**

Solution

$\mathsf{PackedShare}_{\mathbb{F}_p,n,k,t}(\mathbf{s})$

1 : **parse** $\mathbf{s} = (s_1, \ldots, s_k)$
2 : $a_{k+1}, \ldots, a_{k+t-1} \xleftarrow{R} \mathbb{F}_p$
3 : $P(x) = a_{k+t-1}x^{k+t-1} + \cdots + a_{k+1}x^{k+1} + s_k x^k + s_{k-1}x^{k-1} + \cdots + s_2 x^2 + s_1 \bmod p$
4 : **return** $P(1), P(2), \ldots, P(n)$

$\mathsf{PackedRecover}_{\mathbb{F}_p}(z_1, \ldots, z_{t+k})$

1 : $P(x) \leftarrow \mathsf{Interpolate}(z_1, \ldots, z_{k+t})$
2 : **parse** $P(x) = a_{k+t-1}x^{k+t-1} + \cdots + a_{k+1}x^{k+1} + s_k x^k + s_{k-1}x^{k-1} + \cdots + s_2 x^2 + s_1 \bmod p$
3 : **return** $\mathbf{s} = (s_1, \ldots, s_k)$

We will prove correctness and privacy.

Correctness. With at least $k + t$ shares, each of which are distinct points on the polynomial P, Langrange interpolation guarantees the recovery of a *unique* degree $t + k - 1$ polynomial P. As such, the first k coefficients of P (including the constant term) will contain the k packed secrets.

Privacy. In $\mathsf{PackedShare}$, the polynomial P is constructed using $t - 1$ random coefficients. Any $t - 1$ distinct evaluation points of the polynomial P reveals no information on the coefficients $s_1, \ldots, s_k$ since the distribution of degree $k+t-1$ polynomials is independent of the $t-1$ evaluation points. In other words, conditioned on $t-1$ evaluation points, all possible tuples $(s_1, \ldots, s_k) \in \mathbb{Z}_p^k$ are equally probable.

(b) **Show that your secret-sharing scheme from (a) is additively homomorphic; that is, two secret-shares encoding secret vectors $\mathbf{s}_0$ and $\mathbf{s}_1$, respectively, can be added together locally (i.e., without interacting with the other parties) to produce a share of the sum $\mathbf{s}_0 + \mathbf{s}_1 \in \mathbb{Z}_p^k$.**

Solution

Consider secret-shares $s_0 = P_0(i) \in Z_p$ and of $s_1 = P_1(i) \in \mathbb{Z}_p$ generated according to $\mathsf{PackedShare}$. The share addition $s_0 + s_1 = P_0(i) + P_1(i) \in \mathbb{Z}_p$ corresponds to the addition of the polynomials P_0 and P_1 when recovered through Lagrange interpolation. As such, the polynomial P' recovered by interpolating secret-shares $s^i = s_0^i + s_1^i$ for $i \subseteq \{1, \ldots, t + k - 1\}$ $P_0 + P_1$ (where the addition is defined coefficient-wise) will have the first k coefficients equal to $\mathbf{s}_0 + \mathbf{s}_1$, as required.

Problem 2. Upgrading oblivious transfer. Recall that in class, we learned about 1-out-of-2 Oblivious Transfer (OT) schemes. In the OT setting, a sender has two messages $m_0, m_1 \in \{0, 1\}$, and a receiver has choice bit $b \in \{0, 1\}$. The sender wants to send m_b to the receiver while satisfying

correctness (the receiver obtains m_b), *sender's privacy* (the receiver gains no knowledge about the message m_{1-b}, and *receiver's privacy* (the sender gains no knowledge about the choice bit b).

In this problem we focus on achieving security against semi-honest (or "honest-but-curious") senders and receivers.

(a) **Show how you can use any 1-bit OT scheme to build an ℓ-bit OT scheme for transferring ℓ-bit messages $m_0, m_1 \in \{0,1\}^\ell$. Here $\ell = \ell(\lambda)$ is polynomial in the security parameter λ. You can assume the existence of a PRG which expands λ bits to $\ell(\lambda)$ bits, but your scheme can only invoke the 1-bit OT scheme at most $\lambda \ll \ell$ times.**

Solution

Let $\mathsf{OT}(m_0, m_1, b)$ be a OT protocol. Define our protocol as follows:

long-OT protocol

sender samples $s_0, s_1 \xleftarrow{R} \{0,1\}^\lambda$

sender sends receiver $c_0 \leftarrow m_0 \oplus G(s_0), c_1 \leftarrow m_1 \oplus G(s_1)$

for $i = 1, \ldots, \lambda$:

  sender and receiver run $\mathsf{OT}(s_0[i], s_1[i], b)$

receiver reconstructs s_b and computes $G(s_b)$

receiver recovers $m_b \leftarrow c_b \oplus G(s_b)$

We have the following properties for our protocol:

Correctness. $c_b \oplus G(s_b) = m_b \oplus G(s_b) \oplus G(s_b) = m_b$.

Sender privacy. Let $\mathsf{Sim}_R(b, s_b[i])$ simulate the receiver's view of $\mathsf{OT}(s_0[i], s_1[i], b)$. Then we can define our receiver simulator as follows:

long-$\mathsf{Sim}_R(b, m_b)$

$s_b \xleftarrow{R} \{0,1\}^\lambda,\ c_b \leftarrow m_b \oplus G(s_b)$

$c_{1-b} \xleftarrow{R} \{0,1\}^\ell$

return $\left(c_0, c_1, \{\mathsf{Sim}_R(b, s_b[i])\}_{i=1}^\lambda\right)$

c_b is distributed correctly because it is generated identically as in our protocol. c_{1-b} is distributed correctly (i.e. computationally indistinguishable from random) by the pseudorandomness of G.

$\{\mathsf{Sim}_R(b, s_b[i])\}_{i=1}^\lambda$ is distributed correctly by the security of OT and a standard hybrid argument. This simulator is ppt, since it only performs efficient operations and then runs the ppt Sim_R λ times.

Receiver privacy. Let $\mathsf{Sim}_S(s_0[i], s_1[i])$ simulate the sender's view of $\mathsf{OT}(s_0[i], s_1[i], b)$. Then we can define our sender simulator as follows:

long-$\mathsf{Sim}_S(m_0, m_1)$

$s_0, s_1 \xleftarrow{R} \{0,1\}^\lambda$

$c_0 \leftarrow m_0 \oplus G(s_0), c_1 \leftarrow m_1 \oplus G(s_1)$

return $\left(s_0, s_1, c_0, c_1, \{\mathsf{Sim}_S(s_0[i], s_1[i])\}_{i=1}^\lambda\right)$

s_0, s_1, c_0, and c_1 are all distributed correctly because they are generated identically as in our protocol.

$\{\mathsf{Sim}_S(s_0[i], s_1[i])\}_{i=1}^\lambda$ is distributed correctly by the security of OT and a standard hybrid argument. This simulator is ppt, since it only performs efficient operations and then runs the ppt Sim_S λ times.

(b) **Show how you can use any ℓ-bit OT scheme (e.g., your scheme from the previous problem) to design a 1-*out-of*-n ℓ-bit OT scheme with integers $n \geq 2$ and $\ell = \ell(\lambda)$.**
Specifically, show how to construct a scheme where the sender has messages $m_1, \ldots, m_n \in \{0,1\}^\ell$ and the receiver has choice *index* $b \in \{1, \ldots, n\}$. You can assume the existence of a PRF family $\mathcal{F} : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$, but your scheme must invoke the ℓ-bit OT scheme at

most $O(\log n)$ number of times.

Solution

Intuitively, we want to arrange the n messages at the leaves of a tree of depth $k = \log n$. For each of the k levels, we pick two random ℓ-bit strings, one for the left branch and one for the right branch. This allows us to use the XOR of all the random strings along the path down the tree as a one-time-pad for a message m_i to get its encryption c_i. We can run the 1-out-of-2 OT protocol k times, so that the receiver learns the k random strings it needs to XOR to retrieve the message it wants. This almost works, except it's not sender-private. Consider the following attack: the receiver can compute $c_0 \oplus c_1 \oplus c_2 \oplus c_3$, then the one-time-pads will cancel and the receiver learns $m_0 \oplus m_1 \oplus m_2 \oplus m_3$. This is why we need an additional step of using a PRF to prevent this attack.

Let long-$\mathsf{OT}(m_0, m_1, b)$ be a 1-out-of-2 OT protocol for ℓ-bit messages. Define our protocol as follows:

many-OT protocol

sender samples $r_0^1, \ldots, r_0^{\log n}, r_1^1, \ldots, r_1^{\log n} \xleftarrow{R} \{0,1\}^\ell$

for $j = 0, \ldots, n-1$: (in bit representation)

  sender computes $R_j \leftarrow \mathcal{F}(r^1_{j[1]}, j||0^{\ell - \log n}) \oplus \cdots \oplus \mathcal{F}(r^{\log n}_{j[\log n]}, j||0^{\ell-\log n})$

  sender sends receiver $c_j \leftarrow m_j \oplus R_j$

for $i = 1, \ldots, \log n$:

  sender and receiver run long-$\mathsf{OT}(r_0^i, r_1^i, b[i])$

receiver computes $R_b \leftarrow \mathcal{F}(r^1_{b[1]}, b||0^{\ell - \log n}) \oplus \cdots \oplus \mathcal{F}(r^{\log n}_{b[\log n]}, b||0^{\ell-\log n})$

receiver recovers $m_b \leftarrow c_b \oplus R_b$

We have the following properties for our protocol:

Correctness. $c_b \oplus R_b = m_b \oplus R_b \oplus R_b = m_b$.

Sender privacy. Let long-$\mathsf{Sim}_R(b[i], r^i_{b[i]})$ simulate the receiver's view of long-$\mathsf{OT}(r_0^i, r_1^i, b[i])$. Then we can define our receiver simulator as follows:

many-$\mathsf{Sim}_R(b, m_b)$

$r^1_{b[1]}, \ldots, r^{\log n}_{b[\log n]} \leftarrow\!\!\$\, \{0,1\}^\ell,\ R_b \leftarrow \mathcal{F}(r^1_{b[1]}, b||0^{\ell-\log n}) \oplus \cdots \oplus \mathcal{F}(r^{\log n}_{b[\log n]}, b||0^{\ell-\log n})$

$c_b \leftarrow m_b \oplus R_b,\ c_0, \ldots, c_{b-1}, c_{b+1}, \ldots, c_{n-1} \leftarrow \{0,1\}^\ell$

return $\left(c_0, \ldots, c_{n-1}, \{\text{long-}\mathsf{Sim}_R(b[i], r^i_{b[i]})\}_{i=1}^{\log n}\right)$

c_b is distributed correctly because it is generated identically as in our protocol. For all $j \neq b$, there exists i s.t. $j[i] = 1 - b[i]$, so c_j is distributed correctly (i.e. computationally indistinguishable from random) by the pseudorandomness of $\mathcal{F}(r^i_{j[i]}, j||0^{\ell-\log n})$ (since the receiver learns nothing about $r^i_{j[i]}$). $\{\text{long-}\mathsf{Sim}_R(b[i], r^i_{b[i]})\}_{i=1}^{\log n}$ is distributed correctly by the security of OT and a standard hybrid argument. This simulator is ppt, since it only performs efficient operations and then runs the ppt long-Sim_R $\log n$ times.

Receiver privacy. Let long-$\mathsf{Sim}_S(r_0^i, r_1^i)$ simulate the sender's view of long-$\mathsf{OT}(r_0^i, r_1^i, b[i])$. Then we can define our sender simulator as follows:

many-$\mathsf{Sim}_S(m_0, \ldots, m_{n-1})$

$r_0^1, \ldots, r_0^{\log n}, r_1^1, \ldots, r_1^{\log n} \leftarrow\!\!\$\, \{0,1\}^\ell$

for $j = 0, \ldots, n-1$: (in bit representation)

  $R_j \leftarrow \mathcal{F}(r^1_{j[1]}, j||0^{\ell-\log n}) \oplus \cdots \oplus \mathcal{F}(r^{\log n}_{j[\log n]}, j||0^{\ell-\log n}),\ c_j \leftarrow m_j \oplus R_j$

return $\left(r_0^1, \ldots, r_0^{\log n}, r_1^1, \ldots, r_1^{\log n}, c_0, \ldots, c_{n-1}, \{\text{long-}\mathsf{Sim}_S(r_0^i, r_1^i)\}_{i=1}^{\log n}\right)$

The r^i and c_j are distributed correctly because they are generated identically as in our protocol. $\{\text{long-}\mathsf{Sim}_S(r_0^i, r_1^i)\}_{i=1}^{\log n}$ is distributed correctly by the security of OT and a standard hybrid argument. This simulator is ppt, since it only performs efficient operations and then runs the ppt long-Sim_S $\log n$ times.

Problem 3. Malicious-receiver oblivious transfer.

In Figure 2, we construct a candidate scheme for 1-out-of-2 ℓ-bit OT (where $\ell = \lfloor \log(p-1) \rfloor$) that we claim is secure against a *malicious* receiver based on the hardness of DDH in the subgroup QR_p of $\mathbb{Z}_p^*$, where $p = 2q+1$ and g is a generator of QR_p.

Prove that the scheme in Figure 2 is correct. Then either prove that the scheme is secure assuming DDH holds or provide an attack showing how a malicious receiver can learn both of the sender's messages ~~bits~~.

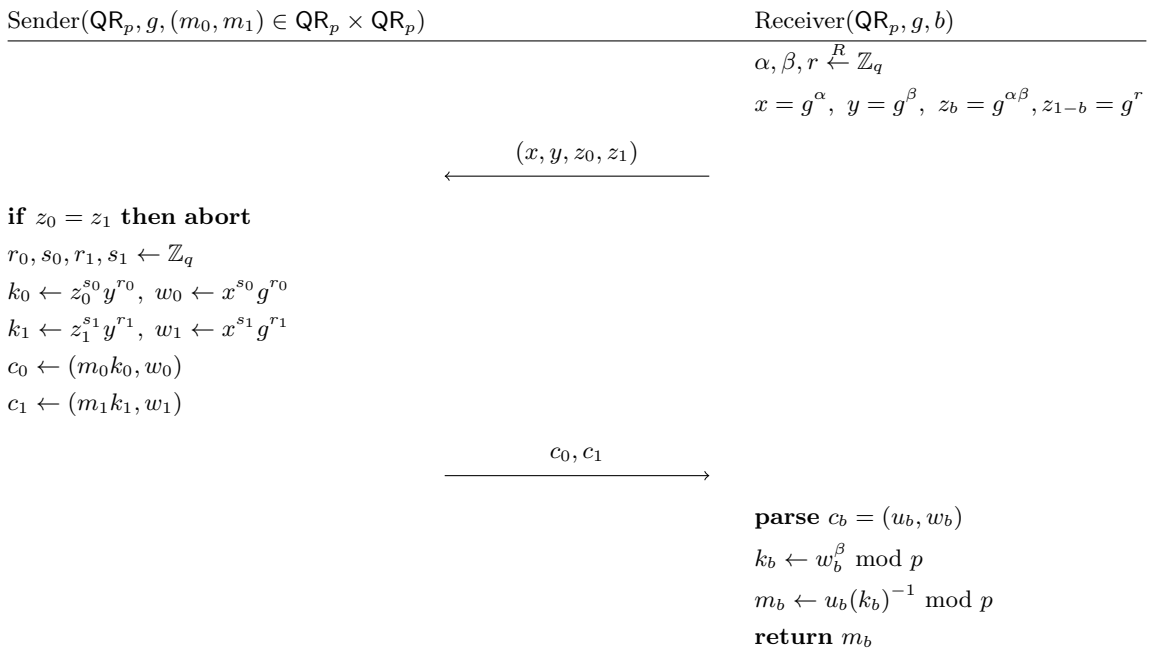

Figure 2: Candidate construction for malicious-receiver 1-out-of-2 ℓ-bit OT.

Solution

The scheme is correct and secure assuming DDH is hard in QR_p. Note that both z_0 and z_1 have to be elements of QR_p for the equality check to parse, which rules out a trivial attack that sets $z_0 = 1$ and $z_1 = -1$.

Intuitively, you should think of the receiver's first message as containing two public keys for an encryption scheme, one which is valid and one which is *lossy*. Encryptions by the valid public key can be correctly decrypted, but encryptions by the lossy public key cannot, and are in fact distributed independently of the plaintext. The sender checks that the receiver didn't send two valid public keys to ensure that a malicious receiver can never recover both messages. The sender then encrypts her messages under the public keys and sends them to the receiver, who is able to decrypt only the chosen message and learns nothing about the other message.

Correctness

$$c_b = (m_b k_b, w_b) = (\underbrace{m_b z_b^{s_b} y^{r_b}}_{u_b}, \underbrace{x^{s_b} g^{r_b}}_{w_b}) \Rightarrow u_b w_b^{-\beta} = m_b (g^{\alpha\beta s_b} g^{\beta r_b})(g^{\alpha s_b} g^{r_b})^{-\beta} = m_b.$$

Sender privacy Only one of the keys, namely k_b, is a valid DDH tuple. We will show that c_{1-b} is information theoretically hiding as a result. To see this, consider the case where $z_0 \neq z_1$ (otherwise the sender aborts). Then we have that (x, y, z_0) or (x, y, z_1) is a valid DDH tuple. Another way of stating the above is that: $z_{1-b} = g^{\alpha\beta+v}$ for some integer $v \in \mathbb{Z}_q$ where $v \neq 0$. This results in the value $k_{1-b} = (g^{\alpha\beta+v})^{s_{1-b}} (g^{\beta})^{r_{1-b}}$ and $w_{1-b} = (g^{\alpha})^{s_{1-b}} (g^{r_{1-b}})$. Consider then $c_{1-b} = (m_{1-b} k_{1-b}, w_{1-b})$. The value $m_{1-b} k_{1-b} = (g^{(\alpha\beta+v)s_{1-b}} g^{\beta r_{1-b}})$ is independent of $w_{1-b} = g^{\alpha s_{1-b}} g^{r_{1-b}}$ due to the additive factor $v s_{1-b}$ in the exponent. Specifically, for any choice of v, there exists an $m'_{1-b} \neq m_{1-b}$ and $s'_{1-b} \neq s_{1-b}$ such that:

$$m_{1-b} g^{\alpha\beta s_{1-b} + v s_{1-b}} g^{\beta r_{1-b}} = m'_{1-b} g^{\alpha\beta s'_{1-b} + v s'_{1-b}} g^{\beta r_{1-b}}.$$

This in essence makes c_{1-b} a one-time pad encryption of m_{1-b} ensuring perfect hiding.

Receiver privacy Assuming DDH is hard, recovering the bit b is computationally infeasible. Suppose, towards contradiction that this is not the case. Then there exits a PPT $\mathcal{A}$ (modeling the sender) which receives as input $(\mathsf{QR}_p, g, x, y, z_0, z_1)$ and outputs b (the receiver's choice bit) with non-negligible advantage $\delta(\lambda)$. Construct PPT $\mathcal{B}$ that breaks DDH with non-negligible advantage as follows:

1. Receive as input DDH tuple $(\mathsf{QR}_p, g, x, y, z)$
2. $b \xleftarrow{R} \{0, 1\}$, $v \xleftarrow{R} \mathbb{Z}_q$ such that $v \neq 0$
3. Compute $z_b = z$, $z_{1-b} \leftarrow z g^v$
4. Run $b' \leftarrow \mathcal{A}(\mathsf{QR}_p, g, x, y, z_0, z_1)$.
5. Return 1 if $b = b'$ and 0 otherwise.

Analysis: If $\mathcal{B}$ receives as input a valid DDH tuple, then $\mathcal{A}$ receives an input distributed exactly as in the protocol. Therefore, if $\mathcal{A}$ has non-negligible advantage in recovering the receiver's choice bit, then $\mathcal{B}$ distinguishes distinguishes between DDH and random with the same advantage as $\mathcal{A}$, namely $\delta(\lambda)$. If $\mathcal{B}$ receives as input a random tuple, then with all-but-negligible probability, z_0 and z_1 are uniformly random. In this case, $\mathcal{A}$'s behavior is undefined and so $\mathcal{B}$'s advantage is 0. Putting these two cases together, we get that $\mathcal{B}$ has advantage $\frac{1}{2}\delta(\lambda) - \mathsf{negl}(\lambda)$ in the DDH game assuming that $\mathcal{A}$ has advantage $\delta(\lambda)$ in distinguishing the receiver's choice bit. By contrapositive, we have that recipient privacy is preserved conditioned on DDH being computationally hard in QR_p.

Problem 4. Private information retrieval. For this problem, we will use a simplified version of the fully homomorphic encryption scheme due to Gentry, Sahai, and Waters [GSW13]. The GSW encryption scheme described below is parametrized by key length n, modulus q, and error bound B:

$\mathsf{Gen}(1^\lambda)$:

- sample $\overline{\mathbf{s}} \xleftarrow{R} \mathbb{Z}_q^n$,
- construct $\mathbf{s} := \begin{bmatrix} \overline{\mathbf{s}} \\ -1 \end{bmatrix}$,
- output the secret key $\mathbf{s}$.

$\mathsf{Enc}(\mathbf{s}, b \in \{0,1\})$:

- parse $\mathbf{s} = \begin{bmatrix} \overline{\mathbf{s}} \\ -1 \end{bmatrix}$,
- sample $\overline{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$,
- sample $\mathbf{e} \xleftarrow{R} \{-B, \ldots, B\}^m$,
- construct $\mathbf{A} := \begin{bmatrix} \overline{\mathbf{A}} \\ \overline{\mathbf{s}}^\top \overline{\mathbf{A}} - \mathbf{e} \end{bmatrix}$,
- output the ciphertext $\mathbf{C} := \mathbf{A} + b\mathbf{G}$.

We set $m = (n+1)\log q$ so the dimensions of $\mathbf{A}$ match the dimensions of $\mathbf{G}$ (the gadget matrix). To decrypt, note that for a ciphertext $\mathbf{C} = \mathbf{A} + b\mathbf{G}$, we have

$$\mathbf{s}^\top \mathbf{C} = \mathbf{e} + b\mathbf{s}^\top \mathbf{G} \bmod q$$

so we output 0 if $\mathbf{s}^\top \mathbf{C} \approx 0 \bmod q$ and 1 otherwise.

Recall from class that the GSW scheme also supports homormorphic operations. Given ciphertexts $\mathbf{C}_1 = \mathbf{A}_1 + b_1\mathbf{G}$ and $\mathbf{C}_2 = \mathbf{A}_2 + b_2\mathbf{G}$ of bits b_1 and b_2, we can compute

- $\mathsf{ADD}(\mathbf{C}_1, \mathbf{C}_2) = \mathbf{C}_1 + \mathbf{C}_2$. This is an encryption of $b_1 + b_2$ because

$$\begin{aligned} \mathbf{s}^\top(\mathbf{C}_1 + \mathbf{C}_2) &= \mathbf{s}^\top \mathbf{C}_1 + \mathbf{s}^\top \mathbf{C}_2 \\ &= (\mathbf{e}_1 + \mathbf{e}_2) + (b_1 + b_2)\mathbf{s}^\top \mathbf{G} \approx (b_1 + b_2)\mathbf{s}^\top \mathbf{G} \bmod q. \end{aligned}$$

  (Note that if $b_1 = b_2 = 1$, then $\mathbf{C}_1 + \mathbf{C}_2$ will decrypt to 1.)
- $\mathsf{MULT}(\mathbf{C}_1, \mathbf{C}_2) = \mathbf{C}_1 \mathbf{G}^{-1}(\mathbf{C}_2)$. This is an encryption of $b_1 b_2$ because

$$\begin{aligned} \mathbf{s}^\top(\mathbf{C}_1 \mathbf{G}^{-1}(\mathbf{C}_2)) &= \left(\mathbf{s}^\top \mathbf{C}_1\right) \mathbf{G}^{-1}(\mathbf{C}_2) = \left(\mathbf{e}_1 + b_1 \mathbf{s}^\top \mathbf{G}\right) \mathbf{G}^{-1}(\mathbf{C}_2) \\ &= (\mathbf{e}_1 \mathbf{G}^{-1}(\mathbf{C}_2) + b_1 \mathbf{s}^\top \mathbf{C}_2 = \mathbf{e}_1 \mathbf{G}^{-1}(\mathbf{C}_2) + b_1 \left(\mathbf{e}_2 + b_2 \mathbf{s}^\top \mathbf{G}\right) \\ &= \left(\mathbf{e}_1 \mathbf{G}^{-1}(\mathbf{C}_2) + b_1 \mathbf{e_2}\right) + b_1 b_2 \mathbf{s}^\top \mathbf{G} \approx b_1 b_2 \mathbf{s}^\top \mathbf{G} \bmod q. \end{aligned}$$

You may assume that we are always able to decrypt a ciphertext $\mathbf{C}$ correctly if the associated error $\mathbf{e}$ is bounded by $q/8$ (i.e. all entries of the error are in $\{-q/8, q/8\}$).

Suppose we want to perform PIR on a database DB of size $N = 2^k$ bits using the GSW FHE scheme. For convenience, we represent database indices as bit-strings in $\{0,1\}^k$ rather than integers in $[N]$. To retrieve $\mathsf{DB}[\mathbf{x}]$ for some index $\mathbf{x} = x_1 x_2 \ldots x_k$, the server needs to homomorphically evaluate the polynomial

$$f(\mathbf{x}) = \sum_{\mathbf{i} \in \{0,1\}^k} \mathsf{DB}[\mathbf{i}](x_1 - \overline{i_1}) \cdots (x_k - \overline{i_k})$$

where the terms in the summation will be 0 for all $\mathbf{i} \neq \mathbf{x}$ and $\mathsf{DB}[\mathbf{i}]$ for $\mathbf{i} = \mathbf{x}$.

The database must evaluate $f(\mathbf{x})$ given

- encryptions $\mathbf{B}_0, \mathbf{B}_1$ of 0 and 1 (part of the encoded database stored by the server),
- encryptions $\mathbf{C}_1, \ldots, \mathbf{C}_k$ of $x_1, \ldots, x_k$ (sent to the server by the client as the query), and
- encryptions $\{\mathbf{D_i}\}_{\mathbf{i} \in \{0,1\}^k}$ of $\{\mathsf{DB}[\mathbf{i}]\}_{\mathbf{i} \in \{0,1\}^k}$ (part of the encoded database stored by the server).

Different ways to homomorphically evaluate $f(\mathbf{x})$ using the homomorphic operations described above will cause the resulting error in the evaluated ciphertext to grow by different amounts. In this problem, your job is to figure out the best way to evaluate $f(\mathbf{x})$ so as to minimize the error growth as much as possible.

> **Show how to homomorphically evaluate the polynomial f so that the resulting error in the evaluated ciphertext is as small as possible, and provide a bound on the resulting error in terms of n, B, and k. Points will be awarded based on how much your solution minimizes the error growth.**

Solution

$\mathsf{Retrieve}(\mathbf{B}_0, \mathbf{B}_1, \mathbf{C}_1, \ldots, \mathbf{C}_k, \{\mathbf{D}_\mathbf{i}\}_{\mathbf{i}\in\{0,1\}^k})$

1 : **for** $\mathbf{i} \in \{0,1\}^k : \mathbf{F}_\mathbf{i} := \mathbf{D}_\mathbf{i}$
2 : **for** $j \in [k]$:
3 : $\quad \mathbf{C}_j^0 := \mathbf{C}_j - \mathbf{B}_1$
4 : $\quad \mathbf{C}_j^1 := \mathbf{C}_j - \mathbf{B}_0$
5 : $\quad$ **for** $\mathbf{i}' \in \{0,1\}^{k-j}$:
6 : $\quad\quad \mathbf{F}_{\mathbf{i}'} := \mathbf{C}_j^0 \mathbf{G}^{-1}(\mathbf{F}_{0\|\mathbf{i}'}) + \mathbf{C}_j^1 \mathbf{G}^{-1}(\mathbf{F}_{1\|\mathbf{i}'})$
7 : **return** $\mathbf{F}_{\mathsf{null}}$

Correctness of evaluation.

To see why $\mathsf{Retrieve}$ computes f, note that if we define for all $j \in [k]$, $\mathbf{i}' \in \{0,1\}^{k-j}$,

$$F_{\mathbf{i}'} = \sum_{\mathbf{i}\in\{0,1\}^j} \mathsf{DB}[\mathbf{i}\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_j - \overline{i_j}),$$

we can recursively compute this value as follows:

$$\begin{aligned}
F_{\mathbf{i}'\in\{0,1\}^{k-j}} &= \sum_{\mathbf{i}\in\{0,1\}^j} \mathsf{DB}[\mathbf{i}\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_j - \overline{i_j}) \\
&= \sum_{\mathbf{i}\in\{0,1\}^{j-1}} \mathsf{DB}[\mathbf{i}\|0\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_j - 1) + \sum_{\mathbf{i}\in\{0,1\}^{j-1}} \mathsf{DB}[\mathbf{i}\|1\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_j - 0) \\
&= (x_j - 1) \sum_{\mathbf{i}\in\{0,1\}^{j-1}} \mathsf{DB}[\mathbf{i}\|0\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_{j-1} - \overline{i_{j-1}}) \\
&\quad + (x_j - 0) \sum_{\mathbf{i}\in\{0,1\}^{j-1}} \mathsf{DB}[\mathbf{i}\|1\|\mathbf{i}'](x_1 - \overline{i_1}) \cdots (x_{j-1} - \overline{i_{j-1}}) \\
&= (x_j - 1) F_{0\|\mathbf{i}'\in\{0,1\}^{k-(j-1)}} + (x_j - 0) F_{1\|\mathbf{i}'\in\{0,1\}^{k-(j-1)}}.
\end{aligned}$$

For $j = 0$, we have that

$$F_{\mathbf{i}\in\{0,1\}^k} = \mathsf{DB}[\mathbf{i}]$$

and for $j = k$, we have that

$$F_{\mathsf{null}} = \sum_{\mathbf{i}\in\{0,1\}^k} \mathsf{DB}[\mathbf{i}](x_1 - \overline{i_1}) \cdots (x_k - \overline{i_k}) = f(\mathbf{x}).$$

Lastly note that for all $j \in [k]$, $\mathbf{i}' \in \{0,1\}^{k-j}$, we have that $\mathbf{F}_{\mathbf{i}'}$ is an encryption of $F_{\mathbf{i}'}$, and thus $\mathbf{F}_{\mathsf{null}}$ is an encryption of $f(\mathbf{x})$.

Solution (continued...)

Bounding the error growth.

We will use an inductive argument to bound the error of $\mathbf{F}_{\mathsf{null}}$. We will denote the error of a ciphertext $\mathbf{C}$ as $\mathsf{err}(\mathbf{C})$.

Base case: $j = 1$, $\mathbf{i}' \in \{0,1\}^{k-1}$. $\mathbf{F}_{\mathbf{i}'}$ has error

$$\begin{aligned}&\mathsf{err}(\mathbf{C}_1^0)\mathbf{G}^{-1}(\mathbf{F}_{0\|\mathbf{i}'}) + (x_1 - 1)\mathsf{err}(\mathbf{F}_{0\|\mathbf{i}'}) + \mathsf{err}(\mathbf{C}_1^1)\mathbf{G}^{-1}(\mathbf{F}_{1\|\mathbf{i}'}) + (x_1 - 0)\mathsf{err}(\mathbf{F}_{1\|\mathbf{i}'})\\ &= \mathsf{err}(\mathbf{C}_1^0)\mathbf{G}^{-1}(\mathbf{F}_{0\|\mathbf{i}'}) + \mathsf{err}(\mathbf{C}_1^1)\mathbf{G}^{-1}(\mathbf{F}_{1\|\mathbf{i}'}) + \mathsf{err}(\mathbf{F}_{x_1\|\mathbf{i}'}).\end{aligned}$$

$\mathbf{C}_1^b$ is generated by homomorphic addition of two ciphertexts with error bounded by B, and $\mathbf{G}^{-1}(\mathbf{F}_{b\|\mathbf{i}'}) \in \{0,1\}^{m\times m}$, so $\mathsf{err}(\mathbf{C}_1^b)\mathbf{G}^{-1}(\mathbf{F}_{b\|\mathbf{i}'})$ is bounded by $2Bm$. $\mathbf{F}_{x_1\|\mathbf{i}'} = \mathbf{D}_{x_1\|\mathbf{i}'}$, so it has error bounded by B. Thus $\mathbf{F}_{\mathbf{i}'}$ has error bounded by $4Bm + B = 4Bjm + B$ (recall: $j = 1$).

Inductive step: $\mathbf{i}' \in \{0,1\}^{k-j}$, suppose that $\mathbf{F}_{\mathbf{0}\|\mathbf{i}'}$ and $\mathbf{F}_{\mathbf{1}\|\mathbf{i}'}$ have error bounded by $4B(j-1)m + B$. $\mathbf{F}_{\mathbf{i}'}$ has error

$$\begin{aligned}&\mathsf{err}(\mathbf{C}_j^0)\mathbf{G}^{-1}(\mathbf{F}_{0\|\mathbf{i}'}) + (x_j - 1)\mathsf{err}(\mathbf{F}_{0\|\mathbf{i}'}) + \mathsf{err}(\mathbf{C}_j^1)\mathbf{G}^{-1}(\mathbf{F}_{1\|\mathbf{i}'}) + (x_j - 0)\mathsf{err}(\mathbf{F}_{1\|\mathbf{i}'})\\ &= \mathsf{err}(\mathbf{C}_j^0)\mathbf{G}^{-1}(\mathbf{F}_{0\|\mathbf{i}'}) + \mathsf{err}(\mathbf{C}_j^1)\mathbf{G}^{-1}(\mathbf{F}_{1\|\mathbf{i}'}) + \mathsf{err}(\mathbf{F}_{x_1\|\mathbf{i}'}).\end{aligned}$$

As above, $\mathsf{err}(\mathbf{C}_j^b)\mathbf{G}^{-1}(\mathbf{F}_{b\|\mathbf{i}'})$ is bounded by $2Bm$. Thus $\mathbf{F}_{\mathbf{i}'}$ has error bounded by $4Bm + 4B(j-1)m + B = 4Bjm + B$. We can conclude that $\mathbf{F}_{\mathsf{null}}$ has error bounded by $4Bkm + B$.

References

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

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. https://web.mit.edu/6.857/OldStuff/Fall03/ref/Shamir-HowToShareASecret.pdf.