6.875: Cryptography and Cryptanalysis **November 22, 2021**
Last updated: December 6, 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 $\mathbf{s}$.**

(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 $\mathbf{s}_0 + \mathbf{s}_1 \in \mathbb{Z}_p^k$.**

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.**

(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.

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~~.

$\text{Sender}(\mathsf{QR}_p, g, (m_0, m_1) \in \mathsf{QR}_p \times \mathsf{QR}_p)$	$\text{Receiver}(\mathsf{QR}_p, g, b)$
	$\alpha, \beta, r \xleftarrow{R} \mathbb{Z}_q$
	$x = g^\alpha,\ y = g^\beta,\ z_b = g^{\alpha\beta}, z_{1-b} = g^r$
$\xleftarrow{(x, y, z_0, z_1)}$	
if $z_0 = z_1$ **then abort**	
$r_0, s_0, r_1, s_1 \leftarrow \mathbb{Z}_q$	
$k_0 \leftarrow z_0^{s_0} y^{r_0},\ w_0 \leftarrow x^{s_0} g^{r_0}$	
$k_1 \leftarrow z_1^{s_1} y^{r_1},\ w_1 \leftarrow x^{s_1} g^{r_1}$	
$c_0 \leftarrow (m_0 k_0, w_0)$	
$c_1 \leftarrow (m_1 k_1, w_1)$	
$\xrightarrow{c_0, c_1}$	
	parse $c_b = (u_b, w_b)$
	$k_b \leftarrow w_b^\beta \bmod p$
	$m_b \leftarrow u_b (k_b)^{-1} \bmod p$
	return m_b

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

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.**

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.