Massachusetts Institute of Technology
6.5610: Applied Cryptography
Henry Corrigan-Gibbs and Yael Tauman Kalai

April 4, 2024
Due: March 15, 2024 at 4:59pm

Problem Set 3

Problem sets are due at 4:59pm on the due date.

Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate page.*

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza. See the course website for our policy on collaboration. Each group member must independently write up and submit their own solutions.

You must typeset your homework in LaTeX *and submit it electronically!* Each problem answer must be provided as a separate page. Mark the top of each page with your group member names, the course number (6.5610), the problem set number and question, and the date. We have provided a template for LaTeX on the course website (see the *Psets* tab at the top of the page).

Problem 3-1. [Extra Credit] Seminar attendance

We have another speaker coming to the security seminar this week. We will give **extra credit** on this problem set if you attend.

Title: Designing protocols that actually get deployed

Speaker: Eric Rescorla

Abstract: Designing good Internet protocols is hard. Designing protocols that actually get deployed is harder. We take a look at some protocols that have been widely deployed (e.g., TLS 1.3, QUIC, and the WebPKI), and some others which have been less so (e.g., IPsec, SCTP, and DNSSEC/DANE) and draw some lessons about the factors that lead to protocol success and failure.

Bio: Eric Rescorla has contributed extensively to many of the core security protocols used in the Internet, including TLS, DTLS, WebRTC, ACME, and QUIC. He was editor of the TLS 1.3 protocol, which secures the vast majority of Web traffic and co-founder of Let's Encrypt, a free and automated certificate authority that is now the largest on the Internet. He is the former Chief Technology Officer for Firefox and Internet Platform at Mozilla, where he was responsible for setting the overall technical strategy for the Firefox browser and Mozilla's participation in Internet standards and global policy.

Details are:

- Time: Noon on Thursday, March 7
- Place: 32-D463 (Star)
- Also important: There will be free food. We will order more than last week so that we won't run out.

If you have a timing conflict at this time, we cannot give make-up extra credit but there will be many future extra-credit opportunities!

There are other seminar talks as well and many are related to the content that we will cover in this class! See `https://securityseminar.csail.mit.edu/` for details.

Problem 3-2. Random Private Information Retrieval

Consider a weaker variant of the private information retrieval problem called random private informational retrieval (RPIR), where instead of the client querying a specific database index of their choosing, they want to receive a *random* index of the database. It turns out that we can convert an RPIR scheme into a PIR scheme with higher communication costs and extra rounds of back-and-forth at the start.

For a client and server, and a database $\mathbf{D} = (D_1, \ldots, D_N) \in \{0,1\}^N$, define $\mathsf{RPIR}(1^\lambda, \mathbf{D})$, on security parameter λ, to be a scheme which results in the client downloading and storing (j, D_j) where j is a random index $j \xleftarrow{R} [N]$. Security and succinctness apply as defined in lecture. Intuitively, the server will not learn j.

We now describe how to construct a PIR scheme from RPIR. Let $\mathsf{PIR} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ consist of the following algorithms:

- $\mathsf{Setup}(1^\lambda, \mathbf{D})$
  - –Run $\mathsf{RPIR}(1^\lambda, \mathbf{D})$ to produce (j, D_j) stored by client.
- $\mathsf{Query}(1^\lambda, i) \to (\mathsf{qu}, \mathsf{st})$
  - –Client outputs $\mathsf{qu} \leftarrow i \oplus j$ and client state $\mathsf{st} \leftarrow (i, j, D_j)$
- $\mathsf{Answer}(\mathbf{D}, \mathsf{qu}) \to \mathsf{ans}$
  - –Server partitions the set $[N] = \{1, \ldots, N\}$ into $N/2$ pairs $\{k, k \oplus \mathsf{qu}\} = \{k, k \oplus (i \oplus j)\}$
  - –For each pair $\{k, k \oplus \mathsf{qu}\}$, server computes $p_{\{k, k\oplus \mathsf{qu}\}} = D_k \oplus D_{k \oplus \mathsf{qu}}$.
  - –Server outputs $\mathsf{ans} \leftarrow \left\{p_{\{k,k\oplus\mathsf{qu}\}}\right\}$ enumerating over k such that we have one parity in ans per pair.
- $\mathsf{Reconstruct}(\mathsf{ans}, \mathsf{st}) \to b \in \{0,1\}$
  - –???

(a) Fill in $\mathsf{Reconstruct}(\mathsf{ans}, \mathsf{st})$. That is, describe how the client will use its stored state and the index pair parities in ans to recover $b = D_i$. Assume the server can output the p's in order so that the client knows that parity $p_{\{k,k\oplus\mathsf{qu}\}}$ corresponds to index k (assume the client and server can both produce the partition in the same order).

Solution: Find $p_{\{k,k\oplus\mathsf{qu}\}}$ such that $k = j$ or $k = i$. Then (assuming we get $k = j$, but the same result holds for $k = i$),

$$\begin{aligned} D_i &= D_j \oplus p_{\{j, j\oplus\mathsf{qu}\}} \\ &= D_j \oplus p_{\{j,i\}} \text{ as } j \oplus \mathsf{qu} = j \oplus (i \oplus j) = i \\ &= D_j \oplus (D_j \oplus D_i) = D_i \end{aligned}$$

See this cool paper from Gentry et al. for more details! https://eprint.iacr.org/2020/1248.pdf

(b) Argue in 1-2 sentences that $\mathsf{PIR} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ satisfies security.

Solution: The output of $\mathsf{Query}(1^n, i) = i \oplus j$ looks random, since j is a random index unknown to the server (by the security of RPIR). Therefore, the pairs of indices that the server computes are indistinguishable from random pairs of indices.

(c) Argue in 1-2 sentences that $(\mathsf{Query}, \mathsf{Answer})$ result in total communication $< N$.

Solution: The output of $\mathsf{Query}(1^\lambda, i)$ is $2\log(N)$ bits, since i and j are indices. The output of $\mathsf{Answer}(\mathbf{D}, \mathsf{qu})$ is $N/2$ (bit, index). So the total bit length is $2\log(N) + N/2 < N$ for sufficiently large N.

Problem 3-3. Learning parities with noise

The learning-parity-with-noise assumption is similar to the LWE assumption with two differences. First, LPN is over $\mathsf{GF}[2]$; i.e., addition and multiplication is done modulo 2. Second, the noise is a Bernouli random variable with probability ϵ; i.e., each linear relation is corrupted with probability ϵ. Specifically, the $\mathsf{LPN}_{n,m,\epsilon}$ assumption asserts that

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}) \approx (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \xleftarrow{R} \{0,1\}^{m \times n}$, $\mathbf{s} \xleftarrow{R} \{0,1\}^n$ and $\mathbf{e} = (e_1, \ldots, e_m) \in \{0,1\}^m$ where each e_i is a boolean random variable that is "1" with probability ϵ and "0" with probability $1 - \epsilon$.

(a) Is the $\mathsf{LPN}_{n,m,\epsilon}$ assumption true for $\epsilon = 0$? Is it true for $\epsilon = 1/2$? Is it true for $\epsilon = 1$?

Solution: No, yes, and no.
In cases $\epsilon = 0$ and $\epsilon = 1$ an adversary can find the secret $\mathbf{s}$ via Gaussian elimination since e is deterministic and known to the adversary.
In case $\epsilon = 1/2$, the vector $\mathbf{As} + \mathbf{e}$ is uniform random and independent of $\mathbf{s}$.

(b) Assuming that the $\mathsf{LPN}_{n,m,\epsilon}$ assumption holds, is it true that

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}, (s_1, \ldots, s_n)) \approx (\mathbf{A}, \mathbf{u}, (s_1, \ldots, s_n))$$

where $\mathbf{A} \xleftarrow{R} \{0,1\}^{m \times 2n}$, $\mathbf{s} = (s_1, \ldots, s_{2n}) \xleftarrow{R} \{0,1\}^{2n}$, $\mathbf{u} \xleftarrow{R} \{0,1\}^m$, and $\mathbf{e} = (e_1, \ldots, e_m) \in \{0,1\}^m$ where each e_i is a boolean random variable that is 1 with probability ϵ?

Solution: Yes. Denote by $A_1 \in \{0,1\}^{m \times n}$ the first n columns of A and denote by $A_2 \in \{0,1\}^{m \times n}$ the last n columns of A. Similarly, denote by $\mathbf{s}_1 \in \{0,1\}^n$ the first n coordinates of $\mathbf{s}$ and denote by $\mathbf{s}_2 \in \{0,1\}^n$ the last n coordinates of $\mathbf{s}$. Then $A\mathbf{s} + e = A_1\mathbf{s}_1 + A_2\mathbf{s}_2 + e$. The adversary knows $A_1\mathbf{s}_1$ so it can subtract it, and then this becomes an $\mathsf{LPN}_{n,m,\epsilon}$ instance. Formally, if an adversary can distinguish between $(A, A\mathbf{s} + e, (s_1, \ldots, s_n))$ and $(A, U, (s_1, \ldots, s_n))$ then it can break the $\mathsf{LPN}_{n,m,\epsilon}$ assumption, as follows: Given a tuple (A_2, y_2) where $A_2 \xleftarrow{R} \{0,1\}^{m \times n}$ and y_2 is either truly random in $\{0,1\}^m$ or is of the form $A_2\mathbf{s}_2 + e$ for $\mathbf{s}_2 \xleftarrow{R} \{0,1\}^n$ and e a Bernouli noise vector, choose at random $A_1 \xleftarrow{R} \{0,1\}^{m \times n}$ and $\mathbf{s}_1 \xleftarrow{R} \{0,1\}^n$, let $A \in \{0,1\}^{2n \times m}$ be the matrix whose first n columns are A_1 and last n columns are A_2 and let $y = A_1\mathbf{s}_1 + y_2$. Run the distinguisher on $(A, y, (s_1, \ldots, s_n))$. Note that if y_2 was random then y is random (independent of A and $\mathbf{s}$), and if $y_2 = \mathbf{s}_2 A_2 + e$ then $y = \mathbf{s}A + e$.

(c) (Extra credit) Assuming that the $\mathsf{LPN}_{n,m,\epsilon}$ assumption holds, is it true that

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}, s_1 \oplus \ldots \oplus s_{n+1}) \approx (\mathbf{A}, \mathbf{u}, s_1 \oplus \ldots \oplus s_{n+1})$$

where $\mathbf{A} \xleftarrow{R} \{0,1\}^{m \times (n+1)}$, $\mathbf{s} = (s_1, \ldots, s_{n+1}) \xleftarrow{R} \{0,1\}^{n+1}$, $\mathbf{u} \xleftarrow{R} \{0,1\}^m$, and $\mathbf{e} \in \{0,1\}^m$ where each e_i is a boolean random variable that is 1 with probability ϵ?

Solution: Yes. Let $b = s_1 \oplus \ldots \oplus s_{n+1}$, so we can write $s_{n+1} = s_1 \oplus \ldots \oplus s_n \oplus b$. Then, we can define a matrix $\mathbf{B} \in \{0,1\}^{m \times n}$ where $\mathbf{B}_{i,j} = \mathbf{A}_{i,j} + \mathbf{A}_{i,n+1}$. Because $\mathbf{A}$ is a random matrix, $\mathbf{B}$ is also random. Let $\mathbf{s}'$ be the first n elements of $\mathbf{s}$, and let $\mathbf{A}'$ be the first n columns of $\mathbf{A}$. Then, $\mathbf{As} + \mathbf{e} = \mathbf{A}'\mathbf{s}' + (s_1 \oplus \ldots \oplus s_n \oplus b)\mathbf{A}_{\cdot,n+1} + \mathbf{e} = \mathbf{Bs}' + b\mathbf{A}_{\cdot,n+1} + \mathbf{e}$. Since the adversary knows b and $\mathbf{A}_{\cdot,n+1}$ they can subtract $b\mathbf{A}_{\cdot,n+1}$, and then we just get an LPN instance $(\mathbf{B}, \mathbf{Bs}' + \mathbf{e})$.

Problem 3-4. Fully Homomorphic Encryption

In what follows we change the encryption algorithm of the fully homomorphic encryption scheme presented in class. We do not change its key generation algorithm or its decryption and homomorphic evaluations, so assume that we use the same parameters (m, n, q, χ) and the same secret key $\mathbf{s} = (-\mathbf{s}', 1)$.

Evaluate the following schemes on correctness, security, homomorphic addition, and homomorphic multiplication. State whether each scheme satisfies each property with a brief proof/explanation.

(a) We change the encryption algorithm to be the following:
$\mathrm{Enc}(\mathbf{s} \in \mathbb{Z}_q^n, b \in \{0,1\})$:

- Sample a random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{(n+1) \times n}$.
- Construct matrix $\mathbf{B} = [\mathbf{A} || \mathbf{As}']$, where $\mathbf{s} = (-\mathbf{s}', 1)$.
- Output $\mathbf{B} + \mathbf{b} \cdot \mathbf{I}_{n+1}$, where $\mathbf{I}_{n+1}$ is the identity matrix in $\mathbb{Z}_q$ of dimension $n + 1$.

Solution:

- Correctness is satisfied: $\text{Enc}(\mathbf{s}, b) \cdot \mathbf{s} = (B + bI)(-\mathbf{s}', 1) = [A||A\mathbf{s}'](-\mathbf{s}', 1) + b(-\mathbf{s}', 1) = b(-\mathbf{s}', 1)$.
- Security is not satisfied: we don't have the error term, so we can just solve with Gaussian elimination to get the secret key.
- We have homomorphic addition because $(C_1 + C_2)\mathbf{s} = C_1\mathbf{s} + C_2\mathbf{s} = b_1\mathbf{s} + b_2\mathbf{s} = (b_1 + b_2)\mathbf{s}$
- We have homomorphic multiplication because $C_1C_2\mathbf{s} = C_1(b_2\mathbf{s}) = (C_1\mathbf{s})b_2 = b_1b_2\mathbf{s}$

(b) We change the encryption algorithm to be the following: $\text{Enc}(\mathbf{s} \in \mathbb{Z}_q^n, b \in \{0, 1\})$:

- Sample a random matrix $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{(n+1)\times n}$.
- Sample random vector $\mathbf{e} \xleftarrow{R} \chi^{n+1}$.
- Construct matrix $\mathbf{B} = [\mathbf{A}||\mathbf{A}\mathbf{s}' + \mathbf{e}]$, where $\mathbf{s} = (-\mathbf{s}', 1)$ and $\mathbf{I}_{n+1}$ is the identity matrix in $\mathbb{Z}_q$ of dimension $n + 1$.
- Output $\mathbf{B} + \mathbf{b} \cdot \mathbf{I}_{n+1}$.

Solution:

- Correctness is satisfied: $\text{Enc}(\mathbf{s}, b) \cdot \mathbf{s} = (B + bI)(-\mathbf{s}', 1) = [A||A\mathbf{s}' + \mathbf{e}](-\mathbf{s}', 1) + b(-\mathbf{s}', 1) = b(-\mathbf{s}', 1) + \mathbf{e}$.
- Security is satisfied by LWE because B looks like a random matrix, so B would act as a one time pad for bI.
- We have homomorphic addition because $(C_1 + C_2)\mathbf{s} = C_1\mathbf{s} + C_2\mathbf{s} = b_1\mathbf{s} + e_1 + b_2\mathbf{s} + e_2 = (b_1 + b_2)\mathbf{s} + (e_1 + e_2)$
- We do not homomorphic multiplication because $C_1C_2\mathbf{s} = C_1(b_2\mathbf{s}) = C_1(b_2\mathbf{s} + e_2) = (C_1\mathbf{s})b_2 + C_1e_2 = (b_1\mathbf{s} + e_1)b_2 + C_1e_2 = b_1b_2\mathbf{s} + (e_1b_2 + C_1e_2)$. We know e_1b_2 is small but C_1e_2 is large, making it so that our error term becomes too large for decryption.

Problem 3-5. Linearization attack on LWE with low noise

In this problem, you are going to implement the linearization attack on LWE. Consider the following mini example of a LWE public key (we omit the modulus for simplicity):

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 1 & -4 \end{pmatrix}, \mathbf{u} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} = \begin{pmatrix} 5 \\ -3 \end{pmatrix}$$

Define $\mathbf{s} = \begin{pmatrix} x \\ y \end{pmatrix}$ and $\mathbf{e} = \begin{pmatrix} e_0 \\ e_1 \end{pmatrix}$, we can rewrite it as a linear system:

$$\begin{cases} 2x + 3y = 5 - e_0 \\ x - 4y = -3 - e_1 \end{cases}$$

Assuming the error is either 0 or 1, the first equation can be phrased as "$2x + 3y$ is either 4 or 5", so we can guarantee that the following equation is always true:

$$(2x + 3y - 4)(2x + 3y - 5) = 0$$

which is equivalent to

$$4x^2 + 12xy + 9y^2 - 18x - 27y + 20 = 0$$

If we perform the above step on each equation in the linear system, we will get m (number of rows in A) polynomials that evaluate to 0. It is still difficult to solve systems of multivariate polynomial equations, but "linearization" comes to the rescue. The idea is simple: when we have the equation

$$4x^2 + 12xy + 9y^2 - 18x - 27y + 20 = 0$$

treat it as a linear equation with 5 variables:

$$4z_1 + 12z_2 + 9z_3 - 18z_4 - 27z_5 + 20 = 0$$

With a sufficient number of equations, we can solve the system using Gaussian elimination. In our mini example, we will have 2 equations and 5 variables, which is not enough, but if $\mathbf{A}$ is a matrix with dimension 5×2, we can recover the secret key with only the public key, assuming that the equations are linearly independent. You may assume that the equations after linearization are linearly independent for the following questions.

(a) If the dimension of $\mathbf{A}$ is $m \times n$ and the error is chosen in the range $[-B, B]$ (so there are $2B + 1$ possible error values), what's the minimum m so that the above attack works? In other words, how many variables are there after linearization?

Solution: $m \geq \binom{2B+n+1}{n} - 1$.

(b) In the file `output.txt`, you can find the information of a LWE public key. Your task is to recover the secret $\mathbf{s}$. Submit the value of $\mathbf{s}$ in the pdf file and the code on Gradescope.

You can find an introduction on SageMath here.

Solution: `s = [58154, 31596, 332, 56837, 4766, 46239, 62131, 34689]`.