Homework 3 – Course 02231 Cryptography

Solutions

Group HW46

December, 2025

Exercise 3.1 (A simple CCA attack on Regev's encryption scheme)

Recall Regev's PKE:

- $\mathsf{Keygen}()$: sample
$$A \leftarrow \mathbb{Z}_p^{\ell \times n}, \quad s \leftarrow \mathbb{Z}_p^n, \quad e \leftarrow q_\sigma.$$
Set $b = As + e \bmod p$ and output $(\mathsf{sk}, \mathsf{pk}) = (s, (A, b))$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ with $m \in \{0,1\}$: sample $r \in \{0,1\}^\ell$, then compute
$$c_0 = r^\top A \bmod p, \qquad c_1 = r^\top b + m \cdot \frac{p-1}{2} \bmod p,$$
and output $c = (c_0, c_1)$.
- $\mathsf{Dec}(\mathsf{sk}, c)$ for $c = (c_0, c_1)$:
$$m' = \left\lfloor \frac{2\left(c_1 - c_0^\top s \bmod p\right)}{p} \right\rceil,$$
and output m'.

We always represent elements of $\mathbb{Z}_p$ as integers in $\{-(p-1)/2, \ldots, (p-1)/2\}$, as stated in the assignment.

(1) Decryption of the modified ciphertext

Let $c^* = (c_0^*, c_1^*)$ be a valid encryption of $m_b \in \{0,1\}$. We define
$$c' = \left(c_0^*,\, c_1^* + \tfrac{p-1}{2} \bmod p\right).$$

We show that c' decrypts to $1 - m_b$ (assuming decryption succeeds, i.e., the noise is small enough, as guaranteed by the scheme parameters).

First, expand the decryption for a general ciphertext $c = (c_0, c_1)$. Using $b = As + e \bmod p$, we get
$$\begin{aligned} c_1 - c_0^\top s &= r^\top b + m \cdot \frac{p-1}{2} - (r^\top A)s \bmod p \\ &= r^\top (As + e) + m \cdot \frac{p-1}{2} - r^\top As \bmod p \\ &= r^\top e + m \cdot \frac{p-1}{2} \bmod p. \end{aligned}$$

Denote $u := r^\top e + m \cdot \frac{p-1}{2}$ represented in $\{-(p-1)/2, \ldots, (p-1)/2\}$. The decryption computes
$$m' = \left\lfloor \frac{2u}{p} \right\rceil.$$

By construction of Regev's scheme, $r^\top e$ is small, so:

- If $m = 0$, then $u \approx r^\top e$ is close to 0, so $\frac{2u}{p}$ is close to 0 and rounds to 0.
- If $m = 1$, then $u \approx \frac{p-1}{2} + r^\top e$ is close to $\frac{p-1}{2}$, so $\frac{2u}{p}$ is close to 1 and rounds to 1.

Thus the decryption is correct.

Now look at the modified ciphertext

$$c' = (c_0^*, c_1^* + \tfrac{p-1}{2} \bmod p),$$

where c^* encrypts m_b with randomness r. Then

$$c_1' - (c_0')^\top s = \left(c_1^* + \tfrac{p-1}{2}\right) - (c_0^*)^\top s = \left(c_1^* - (c_0^*)^\top s\right) + \tfrac{p-1}{2} \bmod p.$$

From above,

$$c_1^* - (c_0^*)^\top s \equiv r^\top e + m_b \cdot \frac{p-1}{2} \pmod p,$$

so, writing $u^* = r^\top e + m_b \cdot \frac{p-1}{2}$ in the symmetric representation, we have

$$u' := c_1' - (c_0')^\top s \equiv u^* + \frac{p-1}{2} \pmod p.$$

We distinguish two cases:

Case $m_b = 0$. Then $u^* \approx r^\top e$ is small, so

$$u' \approx r^\top e + \frac{p-1}{2}.$$

This is close to $\frac{p-1}{2}$, so

$$\frac{2u'}{p} \approx 1,$$

and thus

$$\left\lfloor \frac{2u'}{p} \right\rceil = 1 = 1 - 0.$$

Case $m_b = 1$. Then $u^* \approx r^\top e + \frac{p-1}{2}$, hence

$$u' \approx r^\top e + \frac{p-1}{2} + \frac{p-1}{2} = r^\top e + (p-1).$$

In $\mathbb{Z}_p$ with symmetric representatives $\{-(p-1)/2, \ldots, (p-1)/2\}$, we have

$$p - 1 \equiv -1 \pmod p,$$

so $u' \approx r^\top e - 1$, i.e. it is close to -1. Since $|u'|$ is still very small compared to p, the value

$$\frac{2u'}{p}$$

is very close to 0, and therefore

$$\left\lfloor \frac{2u'}{p} \right\rceil = 0 = 1 - 1.$$

Hence in both cases the modified ciphertext c' decrypts to $1 - m_b$, as claimed.

(2) CCA attack distinguishing $\mathsf{L}_{\text{CCA}-0}$ and $\mathsf{L}_{\text{CCA}-1}$

In the IND-CCA experiment (equivalently, in the libraries $\mathsf{L}_{\text{CCA}-0}$ and $\mathsf{L}_{\text{CCA}-1}$ from the assignment), the adversary has:

- access to the public key pk,
- access to a decryption oracle for any ciphertext $c' \neq c^*$,
- and a challenge ciphertext c^* which is an encryption of some bit m_b (depending on the library $\mathsf{L}_{\text{CCA}-b}$).

We describe an adversary $\mathcal{A}$ that wins the CCA game with probability 1 (up to negligible decryption error):

1. $\mathcal{A}$ interacts with the challenger and obtains pk. It makes no use of the decryption oracle yet.
2. $\mathcal{A}$ triggers the challenge phase to receive the challenge ciphertext $c^* = (c_0^*, c_1^*)$, which encrypts an unknown bit m_b.
3. $\mathcal{A}$ constructs the modified ciphertext
$$c' = \big(c_0^*,\, c_1^* + \tfrac{p-1}{2} \bmod p\big).$$
Note that $c' \neq c^*$ because only c_1 was changed.
4. $\mathcal{A}$ queries the decryption oracle on c', obtaining a bit m'. From part (1) we know (except with negligible probability) that
$$m' = 1 - m_b.$$
5. $\mathcal{A}$ outputs the guess
$$\hat{b} = 1 - m'.$$
Then $\hat{b} = m_b$ with probability $1 - \text{negl}(\lambda)$.

This algorithm respects the interface of both libraries $\mathsf{L}_{\text{CCA}-0}$ and $\mathsf{L}_{\text{CCA}-1}$: it makes a single challenge query and one decryption query on $c' \neq c^*$. In both libraries it recovers m_b with overwhelming probability, and therefore can distinguish them with overwhelming advantage. Hence, Regev's encryption scheme is *not* IND-CCA secure.

Exercise 3.2 (One-time signatures)

We recall the Lamport one-time signature (OTS) scheme for n-bit messages using a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$:

- $\Sigma_{\text{Lam}}.\mathsf{KeyGen}(1^\lambda)$: sample
$$x_{0,1}, x_{1,1}, \ldots, x_{0,n}, x_{1,n} \leftarrow \{0,1\}^\lambda,$$
and set $y_{b,i} = H(x_{b,i})$ for $i \in [n]$, $b \in \{0,1\}$. The public key and secret key are
$$\mathsf{vk} = (y_{0,1}, y_{1,1}, \ldots, y_{0,n}, y_{1,n}), \quad \mathsf{sk} = (x_{0,1}, x_{1,1}, \ldots, x_{0,n}, x_{1,n}).$$
- $\Sigma_{\text{Lam}}.\mathsf{Sign}(\mathsf{sk}, m)$: for $m = m_1 \ldots m_n \in \{0,1\}^n$, output
$$\sigma = (x_{m_1,1}, \ldots, x_{m_n,n}).$$
- $\Sigma_{\text{Lam}}.\mathsf{Ver}(\mathsf{vk}, m, \sigma)$: parse vk as $(y_{0,1}, y_{1,1}, \ldots, y_{0,n}, y_{1,n})$ and $\sigma = (\sigma_1, \ldots, \sigma_n)$; accept iff
$$\forall i \in [n] : \quad H(\sigma_i) = y_{m_i,i}.$$

(1) Correctness of Σ_{Lam}

Let $(\mathsf{vk}, \mathsf{sk})$ be generated by $\Sigma_{\text{Lam}}.\mathsf{KeyGen}$, and let $m \in \{0,1\}^n$.

Signing: $\sigma = (x_{m_1,1}, \ldots, x_{m_n,n})$.

Verification checks $H(\sigma_i) = y_{m_i,i}$ for all i. Since by definition $y_{b,i} = H(x_{b,i})$, we have for each i:

$$H(\sigma_i) = H(x_{m_i,i}) = y_{m_i,i}.$$

Thus the verification condition holds for every coordinate, and the signature is always accepted. Therefore the scheme is correct.

(2) Security of Σ_{Lam} from one-wayness of H

We are given the one-wayness libraries $\mathsf{L}_{H,m}^{\text{ow-real}}$ and $\mathsf{L}_{H,m}^{\text{ow-ideal}}$, and the Lamport signature libraries $\mathsf{L}_{\mathsf{sig\text{-}real}}^{\Sigma_{\text{Lam}}}$ and $\mathsf{L}_{\mathsf{sig\text{-}fake}}^{\Sigma_{\text{Lam}}}$ (from the lecture).

Intuitively, Lamport OTS is secure because forging a signature on a *new* message requires revealing at least one preimage x of some hash value $y = H(x)$ that has not been revealed before, which would break the one-wayness of H.

Formally, let $\mathcal{A}$ be any PPT adversary that distinguishes $\mathsf{L}_{\mathsf{sig\text{-}real}}^{\Sigma_{\text{Lam}}}$ and $\mathsf{L}_{\mathsf{sig\text{-}fake}}^{\Sigma_{\text{Lam}}}$ with non-negligible advantage. In the library formulation used in the course, such a distinguisher must, with non-negligible probability, produce a valid signature on a previously unsigned message (this is exactly the "success" event in the Lamport one-time signature experiment). We construct a PPT adversary $\mathcal{B}$ that distinguishes $\mathsf{L}_{H,m}^{\text{ow-real}}$ and $\mathsf{L}_{H,m}^{\text{ow-ideal}}$ for some $m \geq \lambda$.

High-level idea. $\mathcal{B}$ simulates for $\mathcal{A}$ the view of the Lamport scheme, but for one randomly chosen pair (b^*, i^*) it does *not* know the preimage x_{b^*,i^*}. Instead, it obtains y_{b^*,i^*} by calling the one-wayness oracle $\mathsf{Challenge}()$. If $\mathcal{A}$ ever produces a valid signature on a new message, then (with non-negligible probability) it must reveal a preimage for exactly that hidden y_{b^*,i^*}, which $\mathcal{B}$ can feed to $\mathsf{Check}()$ in the one-wayness game.

Construction of $\mathcal{B}$.

1. $\mathcal{B}$ interacts with the one-wayness oracle, which runs either $\mathsf{L}_{H,m}^{\text{ow-real}}$ or $\mathsf{L}_{H,m}^{\text{ow-ideal}}$.
2. Key generation for the simulation: $\mathcal{B}$ chooses a random index $i^* \in [n]$ and a random bit $b^* \in \{0,1\}$. For all pairs $(b,i) \neq (b^*, i^*)$ it samples $x_{b,i} \leftarrow \{0,1\}^\lambda$ and sets $y_{b,i} = H(x_{b,i})$. For the special pair (b^*, i^*) it calls the one-way $\mathsf{Challenge}()$ oracle to obtain some $y^* = H(x^*)$ for a random and unknown x^*. It sets
$$y_{b^*,i^*} := y^*.$$
The public key given to $\mathcal{A}$ is $\mathsf{vk} = (y_{0,1}, y_{1,1}, \ldots, y_{0,n}, y_{1,n})$.
3. Signature oracle simulation (Lamport is one-time, so we handle one signing query): when $\mathcal{A}$ asks for a signature on a message m, $\mathcal{B}$ returns
$$\sigma = (x_{m_1,1}, \ldots, x_{m_n,n}),$$
provided that $m_{i^*} \neq b^*$. In that case, $\mathcal{B}$ never needs x_{b^*,i^*}, so the simulation is perfect. If $m_{i^*} = b^*$, $\mathcal{B}$ aborts (this happens with probability at most $1/2$ over the random choice of b^*).
4. Eventually, by assumption on its distinguishing advantage, $\mathcal{A}$ outputs a forged signature (m', σ') for some new message $m' \neq m$. For the forgery to verify, we must have for all i:
$$H(\sigma'_i) = y_{m'_i,i}.$$
Since $m' \neq m$, there exists at least one position i such that $m'_i \neq m_i$. With probability at least $1/n$, this differing index is exactly i^*, and with additional probability $1/2$ we have $m'_{i^*} = b^*$ (recall that b^* was random). In that case, the coordinate (b^*, i^*) appears in the forgery. Then we have
$$H(\sigma'_{i^*}) = y_{m'_{i^*},i^*} = y_{b^*,i^*} = y^*.$$

Thus σ'_{i^*} is a preimage of the challenge y^*.

$\mathcal{B}$ now calls $\mathsf{Check}(\sigma'_{i^*})$. In $\mathsf{L}_{H,m}^{\text{ow-real}}$ this returns 1 (because $H(\sigma'_{i^*}) \in Y$), while in $\mathsf{L}_{H,m}^{\text{ow-ideal}}$ it returns 0. Hence $\mathcal{B}$ distinguishes the one-wayness libraries with non-negligible advantage, derived from that of $\mathcal{A}$.

Thus, if H is one-way (for length $m \geq \lambda$), then Σ_{Lam} is secure as a one-time signature scheme.

(3) Sizes for $n = 256$, $\lambda = 128$, and comparison with RSA-FDH

For Lamport OTS:

- Secret key: $2n$ values $x_{b,i}$ of λ bits each (not counted here since we compare pk+signature).
- Public key: $2n$ hash outputs $y_{b,i}$ of λ bits each:

$$|\mathsf{vk}| = 2n\lambda = 2 \cdot 256 \cdot 128 = 65536 \text{ bits.}$$

- Signature: n values, each λ bits:

$$|\sigma| = n\lambda = 256 \cdot 128 = 32768 \text{ bits.}$$

- Combined size:

$$|\mathsf{vk}| + |\sigma| = 65536 + 32768 = 98304 \text{ bits.}$$

This is $98304/8 = 12288$ bytes ≈ 12 KB.

For RSA-FDH with 4096-bit modulus N (so $2^{4095} < N < 2^{4096}$) and a fixed public exponent e that does not need to be included in the pk:

- Public key: just N, of size 4096 bits.
- Signature: one RSA element modulo N, also 4096 bits.
- Combined size:

$$4096 + 4096 = 8192 \text{ bits} \approx 1024 \text{ bytes} = 1 \text{ KB.}$$

So, for these parameters, Lamport OTS uses about $98304/8192 = 12$ times more bits for one public key plus one signature than RSA-FDH.

(4) Correctness of the hash-chain OTS Π

The variant Π uses a small message space $M = \{0, 1, 2, 3, 4, 5, 6, 7\}$:

- $\Pi.\mathsf{KeyGen}(1^\lambda)$: sample $x \leftarrow \{0,1\}^\lambda$, set

$$y = H^8(x) = H(H(\ldots H(x)\ldots)),$$

  and output $(\mathsf{sk}, \mathsf{vk}) = (x, y)$.
- $\Pi.\mathsf{Sign}(\mathsf{sk}, m)$: output $\sigma = H^m(x)$.
- $\Pi.\mathsf{Ver}(\mathsf{vk}, m, \sigma)$: accept iff $y = H^{8-m}(\sigma)$.

  Correctness: For any $m \in \{0, \ldots, 7\}$, the signer outputs $\sigma = H^m(x)$. Then

$$H^{8-m}(\sigma) = H^{8-m}(H^m(x)) = H^8(x) = y.$$

Thus the verification condition $y = H^{8-m}(\sigma)$ always holds, so Π is correct.

(5) Insecurity of Π

We show that Π is not secure by constructing an adversary that can distinguish its signature libraries using at most one query to GetSig.

The essential problem: if the adversary sees a valid signature $\sigma = H^m(x)$ for some m, then it can easily compute valid signatures for all *larger* messages $m' > m$:

$$\sigma' = H^{m'-m}(\sigma) = H^{m'-m}(H^m(x)) = H^{m'}(x),$$

which verifies for message m'.

Adversary $\mathcal{A}$ (one signing query):

1. $\mathcal{A}$ obtains the public key $\mathsf{vk} = y$ from the signature library (either real or fake).
2. It queries $\mathsf{GetSig}(0)$ and receives some string σ.
3. It computes $\sigma' = H(\sigma)$.
4. It checks whether $(m' = 1, \sigma')$ is accepted by $\Pi.\mathsf{Ver}$, i.e. whether $y = H^{8-1}(\sigma') = H^7(\sigma')$ holds.
   - In the *real* signature library: σ is a valid signature on message 0, so $\sigma = H^0(x) = x$, hence $\sigma' = H(x)$ is a valid signature on message 1. Therefore verification of $(1, \sigma')$ always succeeds.
   - In the *fake* signature library: the response to $\mathsf{GetSig}(0)$ is a random λ-bit string, independent of x, so σ' is also random and the probability that $y = H^7(\sigma')$ holds is negligible (on the order of $2^{-\lambda}$).
5. If verification accepts, $\mathcal{A}$ outputs "real"; otherwise it outputs "fake".

Hence $\mathcal{A}$ distinguishes the real and fake signature libraries with non-negligible advantage. Therefore, Π is not a secure one-time signature scheme.

(6) Why the Winternitz OTS Π_W prevents this attack

The Winternitz OTS Π_W is defined as:

- $\Pi_W.\mathsf{KeyGen}$: generate two key pairs $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \Pi.\mathsf{KeyGen}$ for $i \in \{0, 1\}$ and set

$$\mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1), \quad \mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1).$$

- $\Pi_W.\mathsf{Sign}(\mathsf{sk}, m)$: output

$$\sigma = (\sigma_0, \sigma_1) \quad \text{where} \quad \sigma_0 = \Pi.\mathsf{Sign}(\mathsf{sk}_0, m), \quad \sigma_1 = \Pi.\mathsf{Sign}(\mathsf{sk}_1, 7 - m).$$

- $\Pi_W.\mathsf{Ver}(\mathsf{vk}, m, \sigma)$: accept iff

$$\Pi.\mathsf{Ver}(\mathsf{vk}_0, m, \sigma_0) = 1 \quad \text{and} \quad \Pi.\mathsf{Ver}(\mathsf{vk}_1, 7 - m, \sigma_1) = 1.$$

Suppose an adversary queries the signing oracle once on some message m^* and obtains a valid signature

$$\sigma^* = (\sigma_0^*, \sigma_1^*),$$

where

$$\sigma_0^* = H^{m^*}(x_0), \qquad \sigma_1^* = H^{7-m^*}(x_1).$$

From the first component σ_0^* (a hash-chain signature at depth m^*), the adversary can derive valid signatures for all messages $m \geq m^*$ using the same trick as in part (5). From the second component σ_1^* at depth $7 - m^*$, it can derive signatures for all *chain depths* $\geq 7 - m^*$, which correspond to messages t such that

$$\Pi.\mathsf{Sign}(\mathsf{sk}_1, 7 - t) = H^{7-t}(x_1).$$

Starting at depth $7 - m^*$, we can only move forward, so we can cover all depths $d \geq 7 - m^*$; these correspond to messages

$$7 - t = d \geq 7 - m^* \iff t \leq m^*.$$

So:

- From σ_0^* we can sign all messages m with $m \geq m^*$.
- From σ_1^* we can sign all messages m with $m \leq m^*$.

To forge a signature on some $m \neq m^*$ in Π_W, the adversary would need to produce *both* components:

$$\sigma_0 \text{ valid for } m \quad \text{and} \quad \sigma_1 \text{ valid for } 7 - m.$$

However:

- If $m > m^*$: then the adversary can compute σ_0 from σ_0^* (by going forward in the chain), but for the second component it would need a value corresponding to $7 - m < 7 - m^*$, i.e. an earlier point in the chain than $7 - m^*$, which requires inverting H (assumed hard).
- If $m < m^*$: then the adversary can compute σ_1 from σ_1^*, but would need to go *back* from depth m^* in the first chain, again requiring inversion of H.

The only message for which the adversary can construct *both* components is $m = m^*$ itself. Therefore, the previous attack (which used a single signature to derive signatures for different messages) no longer works for Π_W: to forge for $m \neq m^*$, the adversary would have to invert H on at least one of the chains.

(7) Size comparison: Winternitz OTS vs. Lamport with $n = 3$

We compare the combined size (public key + signature) of:

- the Winternitz OTS Π_W as defined above (message space size 8),
- the Lamport OTS Σ_{Lam} with $n = 3$ (message space size $2^3 = 8$),

both using the same hash output length λ.

Winternitz OTS Π_W.

- Each underlying Π key pair has
$$|\mathsf{sk}_i| = \lambda, \quad |\mathsf{vk}_i| = \lambda.$$
- The Winternitz public key is $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$, so
$$|\mathsf{vk}_{\Pi_W}| = 2\lambda.$$
- A signature is $\sigma = (\sigma_0, \sigma_1)$, where each σ_i is one λ-bit string produced by Π, hence
$$|\sigma_{\Pi_W}| = 2\lambda.$$
- Combined size:
$$|\mathsf{vk}_{\Pi_W}| + |\sigma_{\Pi_W}| = 2\lambda + 2\lambda = 4\lambda \text{ bits}.$$

Lamport OTS with $n = 3$.

- Public key: $2n = 6$ hash outputs of λ bits each:

$$|\mathsf{vk}_{\text{Lam}}| = 2n\lambda = 6\lambda.$$

- Signature: $n = 3$ values, each λ bits:

$$|\sigma_{\text{Lam}}| = n\lambda = 3\lambda.$$

- Combined size:

$$|\mathsf{vk}_{\text{Lam}}| + |\sigma_{\text{Lam}}| = 6\lambda + 3\lambda = 9\lambda \text{ bits.}$$

Thus, for the same message space size $|M| = 8$, the Winternitz OTS as defined here uses

$$4\lambda \text{ bits}$$

for one public key plus one signature, while the corresponding Lamport scheme uses

$$9\lambda \text{ bits.}$$

So the Winternitz construction is more than a factor 2 smaller in this measure.