Foundations of Cryptography, Fall 2025
Problem Set 3
Due Friday, Oct 24

Total Number of Points: 100

Collaboration Policy: Collaboration is permitted and encouraged in small groups of at most three students. You are free to collaborate in discussing answers, but you must write up solutions on your own, and must specify in your submission the names of any collaborators. Do not copy any text from your collaborators; the writeup must be entirely your work. Do not write down solutions on a board and copy it verbatim; again, the writeup must be entirely your own words and your own work and should demonstrate clear understanding of the solution. **Solutions should be typeset in LaTeX.** You may make use of published material, provided that you clearly acknowledge all sources/tools used. Of course, scavenging for solutions from prior years is forbidden.

On the Use of LLMs: You may use AI however you wish to deepen your understanding of the lecture material. Upload the notes, talk to your AI about them, ask for more explanation or examples; it's all fine. You may not use LLMs in any way to work on your homework. You may not upload assignments, ask for hints, ask how certain concepts from the lectures might be applied to specific homework problems, or upload your assignments to check for correctness or clarity or anything else. You may not include any AI generated content whatsoever in your homework submissions. If it becomes clear that you have used an AI tool when working on your homework (either directly by making edits or to ask for hints/solutions), we may mark your grade down to reflect that.

1 Better Digital Signatures (30 points)

In lecture, we saw how to construct digital signatures from collision-resistant hash functions. However, it turns out that the construction we gave of digital signatures does not need to use the full power of collision-resistance. Indeed, it suffices to use what is called a family of *universal one way hash functions* (UOWHFs).[1] In this problem, we will show how to build UOWHFs from one-way permutations.[2]

Definition 1. *A hash family* $(\mathsf{Gen}, \mathsf{H} = \{\mathsf{H}_{\mathsf{hk}} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\lambda}\}_{\mathsf{hk}\in\mathcal{K}_\lambda})$ *is a UOWHF family if (a) given* hk, H_{hk} *is computable in polynomial time, (b)* $\lambda < n(\lambda)$*, and (c) for all* $x \in \{0,1\}^n$ *and all poly-size adversaries* $\mathcal{A}$*,*

$$\Pr_{\mathsf{hk}\leftarrow\mathsf{Gen}(1^\lambda)}[x' \leftarrow \mathcal{A}(1^\lambda, \mathsf{hk}, x) : x \neq x' \wedge \mathsf{H}_{\mathsf{hk}}(x) = \mathsf{H}_{\mathsf{hk}}(x')] \leq \mathrm{negl}(\lambda).$$

1. (10 points) We would first like to deal with the aspect of compression. Show that if there exists a UOWHF which compresses its input by a single bit, then there exists a UOWHF which compresses its output by any polynomial factor. That is, given a UOWHF

$$\mathsf{Gen}(1^\lambda) \to \mathcal{K}_\lambda, \mathcal{H} = \left\{\mathsf{H}_{\mathsf{hk}} : \{0,1\}^{\lambda+1} \to \{0,1\}^{\lambda}\right\}_{\mathsf{hk}\in\mathcal{K}_\lambda},$$

[1] These are also known as target-collision-resistant hash functions, for reasons that will be evident from the definition.
[2] It turns out that UOWHFs (and thus digital signatures) can be constructed from one-way functions.

construct a UOWHF that compresses an $n(\lambda)$-bit input to an λ-bit output. That is, construct

$$\mathsf{Gen}'(1^\lambda) \to \mathcal{K}'_\lambda, \mathcal{H}' = \left\{\mathsf{H}'_{\mathsf{hk}'} : \{0,1\}^{n(\lambda)} \to \{0,1\}^\lambda\right\}_{\mathsf{hk}' \in \mathcal{K}'_\lambda},$$

where $n(\lambda) = \mathrm{poly}(\lambda)$.

Thus, it suffices to construct a UOWHF that compresses its input by a single bit. We now show how to construct these objects using two tools: (1) one-way permutations, and (2) hash functions.

2. (5 points) Consider the field $\mathbb{F} := \mathbb{F}_{2^{\lambda+1}}$.[3] If we interpret an input in $\{0,1\}^{\lambda+1}$ as an element of $\mathbb{F}$ (and vice versa)[4], we can define the family of hash functions

$$\overline{\mathsf{H}} = \left\{\overline{\mathsf{H}}_{(a,b)} : \{0,1\}^{\lambda+1} \to \{0,1\}^\lambda\right\}_{a \neq 0, b \in \mathbb{F}}, \quad \overline{\mathsf{H}}_{(a,b)}(x) := (ax+b)_{[1:\lambda]}.$$

Prove that $\overline{\mathsf{H}}$ is a (efficiently computable) family of hash functions that compress their input by a single bit and are always 2-to-1.

3. (5 points) Show that the function family $\overline{\mathsf{H}}$ defined above has the following property: given $x_1 \neq x_2 \in \{0,1\}^{\lambda+1}$, it is possible to sample in polynomial time a random $(a,b) \in (\mathbb{F} \setminus \{0\}) \times \mathbb{F}$ subject to the constraint that $(ax_1+b)_{[1:\lambda]} = (ax_2+b)_{[1:\lambda]}$, that is, x_1 and x_2 collide under $\overline{\mathsf{H}}_{(a,b)}$.

4. (10 points) Finally, let $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way permutation. Let H be the functions defined by

$$\mathsf{H}_{\mathsf{hk}}(x) = \overline{\mathsf{H}}_{\mathsf{hk}}(f(x)).$$

Show that H is a UOWHF family which compresses its input by a single bit, where $\mathsf{Gen}(1^\lambda)$ samples a random pair $\mathsf{hk} := (a,b) \leftarrow (\mathbb{F} \setminus \{0\}) \times \mathbb{F}$.

Remark 1. *Note that since our adversaries are non-uniform/poly-size circuits, one can alternatively think of $\mathcal{A}$ (and any other adversaries) as having $x \in \{0,1\}^n$ hardwired into its circuit, as long as the value of x depends only on n (or λ) and not on the hash key being sampled.*

[3] If you want more information about finite fields, see references here, here, and here.

[4] We assume that the mapping associates the all zeros string $0^{\lambda+1}$ with the field element 0.

2 Collision-Resistant Hash Functions (30 points)

A *compressing collision-resistant function family* is a collection of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \{f_k\}_{k\in\{0,1\}^{\mathrm{poly}(\lambda)}}$ together with an efficient key generation algorithm $\mathsf{Gen}(1^\lambda)$, such that the following two properties hold:

1. Compressing: The functions $f_k : \{0,1\}^{n+m} \to \{0,1\}^n$ maps an input of length $n+m$ bits to an output of length n bits, thereby "compressing" its input, where $m, n = \mathrm{poly}(\lambda)$. For this problem, we will always consider $m > 1$.

2. Collision-resistant: For every polynomial-size adversary $\mathcal{A}$, there is a negligible function μ such that

$$\Pr_{k\leftarrow\mathsf{Gen}(1^\lambda)}[f_k(x) = f_k(x') \text{ and } x \neq x' : (x, x') \leftarrow \mathcal{A}(k)] \leq \mu(\lambda).$$

Our goal is build a *variable-length* collision-resistant hash function family $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{H}_\lambda = \{H_k\}_{k\in\{0,1\}^{\mathrm{poly}(\lambda)}}$

$$H_k : \{0,1\}^* \to \{0,1\}^n,$$

starting from a standard computational assumption from lattice-based cryptography, namely the Short Integer Solution problem.

2.1 A compressing collision-resistant function (10 points)

Definition 2 (Short Integer Solution (SIS) Assumption)**.** *Let N, M, q be integers with $q > 1$ and $M \gg N \log q$, and let $\mathbf{A} \in \mathbb{Z}_q^{N\times M}$ be a uniformly random matrix. The* SIS problem *with parameters (N, M, q) asks to find a nonzero integer vector $\mathbf{z} \in \{-1, 0, 1\}^M$ such that $\mathbf{Az} \equiv \mathbf{0} \pmod q$. The* SIS assumption *states that no poly-size algorithm can solve the SIS problem with non-negligible probability, given matrix $\mathbf{A}$.*

Assuming that the SIS assumption holds, give a family of collision-resistant compressing functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \{f_k : \{0,1\}^M \to \mathbb{Z}_q^N\}_{k\in\{0,1\}^{\mathrm{poly}(\lambda)}}$, where $M, N = \mathrm{poly}(\lambda)$.

2.2 Extending to more input lengths (5 points)

Suppose there exists a collision-resistant compressing function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \{f_k\}_{k\in\{0,1\}^{\mathrm{poly}(\lambda)}}$ and $f_k : \{0,1\}^{n+m} \to \{0,1\}^n$. Define a hash function family $\mathcal{H}' = \{\mathcal{H}'_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{H}'_\lambda = \{H'_k\}_{k\in\{0,1\}^{\mathrm{poly}(\lambda)}}$ and

$$H'_k : \{\{0,1\}^m\}^* \to \{0,1\}^n.$$

That is, it takes as input strings of length any multiple of m and works follows:

- Divide the input x into t blocks of m bits each: $x = x_1\|x_2\|\ldots\|x_t$, where $x_i \in \{0,1\}^m$.
- Let $\mathsf{IV} \in \{0,1\}^n$ be an arbitrary fixed public string.
- Compute $y_0 := \mathsf{IV}$, and $y_i := f_k(y_{i-1}\|x_i)$ for $i = 1, 2, \ldots, t$.
- Finally, set $H'_k(x) = y_t$.

Prove that this hash function family is "collision-resistant" in the following sense: no polynomial-size adversary is able to produce with non-negligible probability $x \neq x'$ such that $|x| = |x'| \in m\mathbb{Z}$ such that $H'_k(x) = H'_k(x')$.

2.3 Insecure padding (5 points)

Now define the hash function family $\mathcal{H}'' = \{\mathcal{H}''_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{H}''_\lambda = \{H''_k\}_{k\in\{0,1\}^{\text{poly}(\lambda)}}$

$$H''_k : \{0,1\}^* \to \{0,1\}^n,$$

that takes as input strings of arbitrary length and outputs

$$H''_k(x) = H'_k(x\|0^\ell),$$

where $\ell \in \{0,1,\ldots,m-1\}$ such that $x\|0^\ell$ has length that is a multiple of m. Give an attack on this hash function—that is, give an efficient algorithm that produces (with probability 1) two inputs x, x' such that $H''_k(x) = H''_k(x')$.

2.4 Collision-resistant hash functions, finally (10 points)

Suppose there exists a collision-resistant compressing function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{F}_\lambda = \{f_k\}_{k\in\{0,1\}^{\text{poly}(\lambda)}}$ and $f_k : \{0,1\}^{n+m} \to \{0,1\}^n$. Define a hash function family $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ where $\mathcal{H}_\lambda = \{H_k\}_{k\in\{0,1\}^{\text{poly}(\lambda)}}$,

$$H_k : \{0,1\}^* \to \{0,1\}^n$$

that takes as input strings of arbitrary length and does the following to compute its output:

- Divide the input x as $x_1\|x_2\|\ldots\|x_t$, where $|x_1| = |x_2| = \ldots = |x_{t-1}| = m-1$ and $|x_t| = m-1-\ell$ for some $0 \le \ell \le m-2$. That is, $|x| = t\cdot(m-1)-\ell$.
- Let $\mathsf{IV} \in \{0,1\}^n$ be a fixed public string.
- For $1 \le i \le t-1$, let $y_i = x_i$, let $y_t = x_t\|0^\ell$ and let y_{t+1} be the $(m-1)$-bit binary representation of ℓ.
- Define $z_1 = f_k(\mathsf{IV}\|0\|y_1)$ and for $2 \le i \le t+1$, define $z_i := f_k(z_{i-1}\|1\|y_i)$.
- Finally, set $H_k(x) \coloneqq z_{t+1}$.

Prove that this hash function family is collision-resistant.

Note: A previous version of this question had a typo and incorrectly defined the type of H_k as $H_k : \{\{0,1\}^m\}^* \to \{0,1\}^n$. However in this part, we want the hash function to take as input strings of arbitrary length.

3 Lossy Encryption (20 points)

In this problem, we will explore an alternate notion of encryption called *lossy encryption*. It turns out that this notion is stronger than the usual notion of semantic security for public key encryption and also implies other cryptographic primitives, like oblivious transfer, that we will encounter later on.

A lossy encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has the following syntax:

- $\mathsf{Gen}(1^\lambda, \mathsf{mode})$: The Gen algorithm takes as input the security parameter and a mode which can be either real or lossy. In the real mode, it outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$. In the lossy mode, it output a lossy public key $\widetilde{\mathsf{pk}}$.
- $\mathsf{Enc}(\mathsf{pk}, b)$: The Enc algorithm takes a public key (either a real or a lossy public key) and a bit b and outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: The Dec algorithm takes as input a secret key (which must be a real secret key) and outputs a decrypted bit b.

Furthermore, it has the following properties:

- **Correctness:** The encryption scheme is correct in the real mode. That is, for every bit b,

  $$\Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{real}) : b = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, b))] = 1,$$

  where the probability is over the randomness of Gen, Enc, and Dec.
- **Key Indistinguishability:** Real public keys are computationally indistinguishable from lossy public keys. That is, for any $\lambda \in \mathbb{N}$,

  $$\{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{real})\} \approx_c \{\widetilde{\mathsf{pk}} : \widetilde{\mathsf{pk}} \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{lossy})\}.$$

- **Lossy encryption:** Encryption using the lossy key completely loses information about the message encrypted. That is, output distributions of encryptions of 0 and 1, under lossy keys, are statistically indistinguishable. For every $\widetilde{\mathsf{pk}} \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{lossy})$,

  $$\mathsf{Enc}(\widetilde{\mathsf{pk}}, 0) \equiv_{\mathrm{negl}(\lambda)} \mathsf{Enc}(\widetilde{\mathsf{pk}}, 1).$$

3.1 Semantic security is for free (10 points)

Show that every lossy encryption scheme (when operating in the real mode) is also semantically secure: that is,

$$\{(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0)) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{real})\} \approx_c \{(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1)) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{real})\}.$$

3.2 DDH tuples (5 points)

We will now see how to construct a lossy encryption scheme assuming the Decisional Diffie-Hellman assumption (DDH). Consider a cyclic group $\mathbb{G}$ of prime order p on which DDH assumption holds. That is, suppose that for any generator $g \in \mathbb{G}$,

$$\{(g, g^x, g^y, g^{xy}) : x, y \leftarrow \mathbb{Z}_p\} \approx_c \{(g, g^x, g^y, g^z) : x, y, z \leftarrow \mathbb{Z}_p\}.$$

We say that a tuple (g, h, g', h') is a DDH tuple if there exists an x such that $g^x = g'$ and $h^x = h'$. Let $\mathsf{DLOG}_{\mathbb{G}}(x) = \{h \in \mathbb{G} : (h, h^x)\}$ (h does not have to be a generator).

Algorithm 1: $\mathsf{Rand}(g, h, g', h')$

$s, t \xleftarrow{R} \mathbb{Z}_p$;
$u \leftarrow g^s h^t$ and $v \leftarrow (g')^s (h')^t$;
return (u, v)

Suppose that g and h are generators. **Show that a) if** $(g, g'), (h, h') \in \mathsf{DLOG}_{\mathbb{G}}(x)$**, then** $(u, v) \leftarrow \mathsf{Rand}(g, h, g', h')$ **is a uniformly random element of** $\mathsf{DLOG}_{\mathbb{G}}(x)$ **and b) if** $(g, g') \in \mathsf{DLOG}(x)$ **and** $(h, h') \in \mathsf{DLOG}(y)$ **for** $x \neq y$**, then** $(u, v) \leftarrow \mathsf{Rand}(g, h, g', h')$ **is uniformly random in** $\mathbb{G}^2$**.**

3.3 Lossy encryption from DDH (5 points)

Modify the following encryption scheme by designing a lossy key generation algorithm and proving that the resulting scheme satisfies all three properties of a lossy encryption scheme (assuming DDH). Note that in the encryption scheme, $v \cdot m$ refers to the group operation in $\mathbb{G}$.

DDH-Based Encryption Scheme

- $\mathsf{Gen}(1^\lambda)$: Sample $x, y \leftarrow \mathbb{Z}_p$. Compute $\mathsf{pk} = (g, g^x, g^y, g^{xy})$ and $\mathsf{sk} = y$ before outputting $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: Parse $\mathsf{pk} := (g, h, g', h')$. Compute $(u, v) \leftarrow \mathsf{Rand}(g, h, g', h')$ and output $c = (u, v \cdot m)$.
- $\mathsf{Dec}(\mathsf{sk}, c)$: Parse $c := (c_0, c_1)$ and $\mathsf{sk} := y$. Output c_1 / c_0^y.

4 The QR Assumption (20 points)

For any prime p, define the Legendre symbol

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{if } a \equiv 0 \bmod p, \\ 1 & \text{if } a \not\equiv 0 \bmod p \wedge \exists x : a \equiv x^2 \bmod p, \\ -1 & \text{otherwise.} \end{cases}$$

In other words, $(\frac{a}{p})$ (roughly) indicates whether a is a square modulo p. For any integer a and odd integer $N > 0$ with factorization $N = p_1^{e_1} \dots p_k^{e_k}$, define the Jacobi symbol

$$\left(\frac{a}{N}\right) := \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i}.$$

For any odd integer $N > 0$, let $\mathcal{J}_N^1 = \{a \in \mathbb{Z}_N^* : \left(\frac{a}{N}\right) = 1\}$ be the set of elements in $\mathbb{Z}_N^*$ with Jacobi symbol equal to 1. Let $\mathsf{QR}_N = \{x^2 : x \in \mathbb{Z}_N^*\}$ be the set of quadratic residues modulo N. Let $\mathsf{QNR}_N = \mathbb{Z}_N^* \setminus \mathsf{QR}_N$ be the set of quadratic non-residues modulo N.

Definition 3 (Quadratic Residuosity Problem)**.** *Let $N = pq$ be a product of two distinct odd primes, and let*

$$\mathbb{Z}_N^* = \{\, x \in \mathbb{Z}_N : \gcd(x, N) = 1 \,\}$$

denote the multiplicative group of invertible residues modulo N. An element $x \in \mathbb{Z}_N^*$ is called a quadratic residue modulo N *if there exists some $y \in \mathbb{Z}_N^*$ such that*

$$x \equiv y^2 \pmod N.$$

Otherwise, x is called a quadratic nonresidue *modulo N.*

The Quadratic Residuosity Problem (QRP) *is the following computational task: given an integer $N = pq$, which is a product of two primes p, q, and an element $x \in \mathbb{Z}_N^*$ such that the Jacobi symbol $\left(\frac{x}{N}\right) = 1$, decide whether x is a quadratic residue modulo N.*

We can now define the Quadratic Residuosity Assumption:

Assumption 1 (Quadratic Residuosity Assumption for Blum Integers[5])**.** *For any poly-size algorithm $\mathcal{A}$, its advantage in solving the Quadratic Residuosity Problem is negligible in the security parameter λ. That is, for random $p \neq q$ of λ bits each where $p, q \equiv 3 \bmod 4$, and random $x \in \mathbb{Z}_N^*$ with $\left(\frac{x}{N}\right) = 1$,*

$$\Pr[\mathcal{A}(N, x) = 1 \text{ if } x \text{ is a residue, and } 0 \text{ otherwise}] \leq \frac{1}{2} + \mathrm{negl}(\lambda).$$

Using the QR assumption, we can design a public-key encryption scheme as follows:

QR-Based Encryption Scheme $\mathsf{GM} := (\mathsf{GM.Gen}, \mathsf{GM.Enc}, \mathsf{GM.Dec})$

- $\mathsf{GM.Gen}(1^\lambda)$: Sample distinct λ-bit primes $p, q \equiv 3 \pmod 4$ uniformly from the set

  $$\{a \in \mathsf{Primes} \cap [1, 2^\lambda] : a \equiv 3 \bmod 4\}.$$

  Set $N = pq$, output public key $\mathsf{pk} = N$ and secret key $\mathsf{sk} = (p, q)$.
- $\mathsf{GM.Enc}(\mathsf{pk}, m)$: On input $\mathsf{pk} = N$ and $m \in \{0, 1\}$, choose a uniformly random $r \in \mathbb{Z}_N^*$, and output the ciphertext $c = (-1)^m \cdot r^2$. (Note that $c \in \mathcal{J}_N^1$, regardless of the values of m and r.)

[5] One can also consider QRP for general primes (not just those which are equivalent to 3 modulo 4), but these two assumptions are known to be equivalent.

- $\mathsf{GM.Dec}(\mathsf{sk}, c)$: On input secret key $\mathsf{sk} = (p, q)$ and ciphertext c, output 0 if $c \in \mathsf{QR}_N$, and output 1 if $c \in \mathsf{QNR}_N$.

4.1 A computationally easy problem? (10 points)

Show that if the factors p, q are *known*, there is a polynomial-time algorithm that solves QRP.

Hint: Prove that for odd primes p, one can equivalently define $(\frac{a}{p}) \equiv a^{(p-1)/2} \bmod p$. When is a number a quadratic residue modulo $N = pq$?

4.2 GM is additively homomorphic (5 points)

Show that GM is *additively homomorphic*. That is, given the public key pk and ciphertexts $c = \mathsf{GM.Enc}(\mathsf{pk}, m)$ and $c' = \mathsf{GM.Enc}(\mathsf{pk}, m')$, show how to compute an encryption of $m + m' \bmod 2$.

4.3 GM is re-randomizable (5 points)

Show that GM is *re-randomizable*. That is, provide a randomized algorithm $\mathsf{GM.Rerand}$ that takes as input pk and $c = \mathsf{GM.Enc}(\mathsf{pk}, m)$ and outputs a "fresh encryption" of m, in the following sense:

$$\forall \mathsf{pk} \in \mathsf{Supp}(\mathsf{Gen}(1^n)),\ \forall m \in \{0,1\},\ \forall c \in \mathsf{Supp}(\mathsf{Enc}(\mathsf{pk}, m))$$
$$\mathsf{GM.Rerand}(\mathsf{pk}, c) \equiv \mathsf{GM.Enc}(\mathsf{pk}, m) \tag{1}$$

where $\equiv$ denotes equality of probability distributions, and Supp denotes the set of all possible outputs.

5 RSA and Håstad's Broadcast Attack (*Bonus*: 15 points)

The key generation and encryption algorithms of the RSA scheme work as follows:

RSA Key Generation and Encryption

- $\mathsf{Gen}(1^\lambda)$:
  1. Choose two λ-bit distinct odd primes p and q, and compute the modulus $N = pq$, and Euler's totient function $\varphi(N) = (p-1)(q-1)$.
  2. Choose a public exponent e satisfying $\gcd(e, \varphi(N)) = 1$. Compute the multiplicative inverse $d \equiv e^{-1} \mod \phi(N)$.
  3. Output the public key and secret key as $\mathsf{pk} = (N, e)$, $\mathsf{sk} = d$.
- $\mathsf{Enc}(\mathsf{pk}, m)$: To encrypt a message $m \in \{0,1\}$, sample random $r \in \mathbb{Z}_N^*$. Using the public key (N, e), compute the ciphertext
$$c = (r^e \bmod N, \mathsf{lsb}(r) \oplus m),$$
where lsb is the least significant bit of r.
- $\mathsf{Dec}(\mathsf{sk}, c)$:
  1. Parse $c = (s, b)$ and $\mathsf{sk} = d$.
  2. Compute $r' \equiv s^d \bmod N$ and output $\mathsf{lsb}(r) \oplus b$.

5.1 RSA decryption (5 points)

Show that the RSA encryption scheme satisfies correctness.

5.2 Broadcast attack (10 points)

To make encryption efficient, sometimes the same small public exponent e is used (e.g. $e = 3$) across multiple users. Now suppose a sender transmits encryptions with the same randomness r to k recipients whose public keys are $(N_1, e), \ldots, (N_k, e)$, where each $N_i = p_i q_i$ is pairwise coprime. For each recipient, the ciphertext is

$$c_i = (r^e \bmod N_i, \mathsf{lsb}(r) \oplus m_i) \qquad \text{for } i = 1, \ldots, k.$$

Suppose $e = 3$. Given an attack that efficiently recover m given three messages $(N_i, c_i)_{i=1}^3$.