6.5620 (6.875), Fall 2022

Homework # 4 **Due: November 2 2022, 11:59:59pm ET**

- **Typsetting:** You are encouraged to use LaTeX to typeset your solutions. You can use the following template.
- **Submissions:** Solutions should be submitted to Gradescope.
- **Reference your sources:** If you use material outside the class, please reference your sources (including papers, websites, wikipedia).
- **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.

Problems:

1. *(11 points)* **Enough hash functions!** Let $\mathcal{H} = \{H_k : \{0,1\}^m \to \{0,1\}^n\}_{k\in\{0,1\}^\lambda}$ be a keyed function family, $m = \text{poly}(\lambda), n = \text{poly}(\lambda)$ and $n < m$. Note that any given H_k compresses m bits into $n < m$ bits. $\mathcal{H}$ is a *good-enough* hash function family if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible probability:

   i. $\mathcal{A}$ outputs a string $x_0 \in \{0,1\}^m$.

   ii. The challenger samples a key $k \leftarrow \{0,1\}^\lambda$ uniformly at random, and gives it to $\mathcal{A}$. (The assumption, as with PRFs, is that anybody can evaluate H_k efficiently if they have k. Unlike in the PRF security definition, however, k is public here.)

   iii. $\mathcal{A}$ outputs a string $x_1 \in \{0,1\}^m$ with $x_1 \neq x_0$, and wins if and only if $H_k(x_0) = H_k(x_1)$.

   (a) *(1 points)* Formally prove that any collision resistant hash function family (see problem 2(a) from HW2 for a definition of a collision resistant hash function family) is good-enough.

   (b) *(4 points)* Show that, if a good-enough hash function family with $n \leq m - 1$ exists, then one-way functions exist.

   Hint. For 2 points of partial credit, show that, if a good-enough hash function family with $n \leq m/2$ exists, then one-way functions exist.

   We will now define the notion of a *nearly-enough hash function family.* For $\mathcal{H} = \{H_k : \{0,1\}^m \to \{0,1\}^n\}_{k\in\{0,1\}^\lambda}$, if any p.p.t. adversary cannot win the following game with non-negligible probability:

i. $\mathcal{A}$ outputs a pair of strings $x_0, x_1 \in \{0,1\}^m$ satisfying $x_0 \neq x_1$.

ii. The challenger samples a key $k \leftarrow \{0,1\}^\lambda$ uniformly at random.

iii. $\mathcal{A}$ wins if and only if $H_k(x_0) = H_k(x_1)$.

Then $\mathcal{H}$ is *nearly-enough*. Let $\mathcal{H} = \{H'_A : \{0,1\}^m \to \{0,1\}^{m-1}\}_A$ be defined as

$$H'_A(u) = Au \pmod 2$$

where $A \leftarrow \{0,1\}^{(m-1)\times m}$. Using the same proof as in 2(b) from HW2, we can show that $\mathcal{H}$ is nearly-enough.

(c) *(1 point)* Show that $\mathcal{H}$ is *not* good-enough.

(d) *(2 points)* Construct a p.p.t. algorithm $\mathsf{KeySimulate}(y', y)$ that outputs $A \in \{0,1\}^{m\times m}$, and satisfies that $H_A(y) = H_A(y')$ and that the induced distribution $\mathcal{D}_{y'} = \{A : y \leftarrow_R \{0,1\}^m, A \leftarrow \mathsf{KeySimulate}(y', y)\}$ is statistically indistinguishable from the uniform distribution over $\{0,1\}^{(m-1)\times m}$ for every fixed $y' \in \{0,1\}^m$.

(e) *(4 points)* We can try to upgrade a nearly-enough hash function family into a good-enough hash function family by composing it with our good old friend in cryptography, a one-way permutation (OWP). Let $f : \{0,1\}^m \to \{0,1\}^m$ be a one-way permutation. Two natural candidates for a good-enough hash function family are

i. $\mathcal{H} = \{H'_A\}_A$, $H'_A(x) = H_A(f(x))$, and

ii. $\mathcal{H} = \{H''_A\}_A$, $H''_A(x) = f(H_A(x))$.

For each of these candidates, either show that it is good-enough or provide an attack.

Hint. Let $\mathbb{F}_2 = \{0,1\}$ be the finite field with two elements where additions are done modulo two, and multiplications are as usual. You can use the following fact without proof: the probability that a uniformly random $A \in \mathbb{F}_2^{(m-1)\times m}$ has linearly independent rows over $\mathbb{F}_2$ is lower bounded by a constant $C > 0.28$. (*Contrary to popular belief, the probability that a uniformly random $A \in \mathbb{F}_2^{(m-1)\times m}$ is full-rank over $\mathbb{F}_2$ is not* $1 - \text{negl}$.)

Remarks. We have seen in this problem that one-way permutations are good enough to construct good-enough hash functions, which in turn are good enough for many cryptographic applications. You may wish to try constructing collision-resistant hash functions from one-way permutations over lunch.

2. *(6 points)* **Lossy encryption.** In this problem, we will explore an alternate notion of security for public key encryption schemes called *lossy encryption*. This definition of security is more powerful than IND-CPA security, and allows us to construct other primitives like oblivious transfer and encryption schemes secure against chosen ciphertext attacks.

Lossy encryption schemes have two modes of operation: *real* and *lossy*. In *real* mode, a lossy encryption scheme behaves like a public key encryption scheme. In *lossy* mode, the ciphertexts produced by the encryption algorithm contain *no information* about the message that was encrypted. Formally, a lossy encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has the following syntax:

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

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

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

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

  where $\approx_c$ means that the two distributions on the left– and the right-hand sides (the distribution of pk and the distribution of $\widetilde{pk}$, in this case) are computationally indistinguishable.
- **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. Formally, for every $n \in \mathbb{N}$, and every $\widetilde{pk} \leftarrow \mathsf{Gen}(1^n, \mathsf{lossy})$,

$$\mathsf{Enc}(\widetilde{pk}, 0) \approx_s \mathsf{Enc}(\widetilde{pk}, 1)$$

  where the randomness is the coins of the Enc algorithm, and $\approx_s$ means that the two distributions on the left– and the right-hand sides are negligibly close (in n) in statistical distance.

(a) *(3 points)* Show that every lossy encryption scheme is also IND-CPA secure (operating in the real mode).

(b) *(3 points)* Define a lossy key generation algorithm for the Goldwasser-Micali encryption scheme. Prove the three properties above (correctness, key indistinguishability and lossy encryption) for the Goldwasser-Micali scheme with your lossy key generation algorithm, assuming the Quadratic Residuosity assumption.

Remarks. It is relatively straightforward to define a variant of the ElGamal encryption scheme which has a lossy mode. (This 'variant' has similar real key generation, encryption, and decryption algorithms to classic ElGamal, but not entirely the same ones.) Therefore, lossy public-key encryption can also be constructed from the DDH assumption. You may wish to show this over lunch.

3. *(8 points)* **A signature scheme from** RSA^+**.** In this problem, we will consider the following security assumption:

Conjecture 1 (RSA^+ assumption). Let N be the product of two randomly chosen safe primes p and q with length λ. Let s be a uniformly random element of $\mathbb{Z}_N^*$. It is infeasible in probabilistic time $\text{poly}(\lambda)$ to find a pair (e, t) with $t \in \mathbb{Z}_N^*$, $1 < e < N$, such that $t^e = s \bmod N$.

Note that the RSA^+ assumption differs from the usual RSA assumption only in that the adversary is permitted to choose the exponent e.

(a) *(4 point)* Let $x, y \in \mathbb{Z}_N^*$, and let e, e' be non-negative integers. Define $\lambda := \log(N)$. Assume that e and e' are relatively prime to each other. Given as input (x, y, e, e') such that $x^e = y^{e'} \mod N$, give an efficient (PPT in λ) algorithm to find $\tilde{x} \in \mathbb{Z}_N^*$ such that $\tilde{x}^e = y \mod N$.

(b) *(4 points)* Consider the following signature scheme $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$:

- $\mathsf{Gen}(1^\lambda)$ randomly chooses two safe primes p and q of length λ (i.e. p and q are between $2^{\lambda-1}$ and $2^\lambda - 1$ inclusive, such that $(p-1)/2$ and $(q-1)/2$ are both primes) and computes their product $N = pq$. It also chooses a uniformly random element $r \leftarrow \mathbb{Z}_N^*$. It outputs $pk := (N, r)$, $sk := ((p-1)(q-1), r)$.
- $\mathsf{Sign}(sk, m)$ does the following. Fix a public function $H : \mathbb{Z}_M \to \mathbb{Z}_K$ for $K := 2^{\lambda-2} - 1$ which is an injective, efficiently computable map from integers in $\mathbb{Z}_M$ to *primes* > 2 in $\mathbb{Z}_K$.[1] (We choose the modulus M appropriately so that H can indeed be injective.) We assume that the message m is in $\mathbb{Z}_M$. $\mathsf{Sign}(sk, m)$ computes and outputs $r^{1/H(m)} \bmod N$. Here, $1/H(m)$ refers to the multiplicative inverse of $H(m)$ in $\mathbb{Z}_{\phi(N)}^*$.
- $\mathsf{Ver}(pk, m, \sigma)$ outputs 1 iff $\sigma^{H(m)} = r \bmod N$.

It should be easy to convince yourself that this signature scheme is perfectly correct, given that H has only primes larger than 2 and smaller than both $(p-1)/2$ and $(q-1)/2$ in its range (so that $H(m)$ is always relatively prime to $\phi(N)$, and

[1] There is no particular reason for this choice of K except that we want K to be computable from public information—so that it does not leak information about $\phi(N)$—and smaller than both $(p-1)/2$ and $(q-1)/2$.

always has a multiplicative inverse mod $\phi(N)$ which can be computed by the Sign algorithm).

We will define the following notion of *non-adaptive signature security*, which is weaker than the usual notion of EUF-CMA security. Consider the following security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- $\mathcal{A}$ sends messages $m_1, \ldots, m_L$ to $\mathcal{C}$.
- $\mathcal{C}$ runs $\mathsf{Gen}(1^\lambda)$ to obtain (pk, sk), and shares pk with $\mathcal{A}$, retaining sk for itself.
- $\mathcal{C}$ computes $\sigma_1 = \mathsf{Sign}(sk, m_1), \ldots, \sigma_L = \mathsf{Sign}(sk, m_L)$, and sends $\sigma_1, \ldots, \sigma_L$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a purported forgery (m^*, σ^*) such that $m^* \neq m_i$ for all $i \in [L]$.

$\mathcal{A}$ wins the above game iff $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$. We say that the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is *non-adaptively secure* if no PPT $\mathcal{A}$ can win the game with non-negligible probability.

Note that, in this security game, $\mathcal{A}$ has to send its queries *before* it is given the public key.

Prove that the signature scheme $\mathcal{S}$ is non-adaptively secure, assuming the RSA$^+$ assumption.

(Hint: how should the reduction from the security of $\mathcal{S}$ to the RSA$^+$ assumption choose r in the public key (N, r) that it provides to $\mathcal{A}$? Make sure you justify that the way the reduction chooses r results in a distribution which is identical to the distribution produced by the legitimate challenger $\mathcal{C}$.)

(c) *(Optional; ungraded)* Prove that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ from part (b) is EUF-CMA secure assuming that H is a random oracle from $\mathbb{Z}_M$ to primes in $\mathbb{Z}_K$.

4. *(14 points)* **Database verification.** Clint stores a database on his laptop at work, but he has an evil officemate Mallory. While Clint has a USB drive that he carries everywhere and protects with his dear life, he likes to leave his laptop unlocked and accessible in his office (I guess he lives life on the edge). One day, he decides enough is enough, and he wants to put his cryptography skills from 6.875 to protect his database from tampering. He designs a database verification scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}, \mathsf{DBVer})$ with the following syntax:

   - $(L, s) \leftarrow \mathsf{Setup}(D)$: Takes a table D, and sets up a data structure L and a secret state s. (Note: L may not be a table, but it should represent D in some form. The requirements on L will be clearer once correctness is defined.)
   - $(x, a) \leftarrow \mathsf{Read}(L, i)$: Retrieves entry i from L along with some additional information a.
   - $(L', s') \leftarrow \mathsf{Write}(L, s, i, y)$: Updates entry i in L to be y, and possibly updates the secret state.

- $\mathsf{DBVer}(s, i, x, a)$: Takes in the secret state s, index i, and $(x, a) \leftarrow \mathsf{Read}(L, i)$, and outputs either accept or reject.

Let λ be a security parameter. Assume the original database D which Clint wishes to store on his laptop is a table of size N, where $\lambda \leq N \leq \text{poly}(\lambda)$; and assume that each entry of D has size $\log N$. In order to use the scheme $\mathcal{S}$, Clint gets $(L, s) \leftarrow \mathsf{Setup}(D)$, and stores the secret state s on his USB drive and stores L on his laptop. He wants to be able to efficiently verify that the contents of L are not tampered with by Mallory. In particular, he wants the scheme to satisfy the following properties:

- *Correctness:* If L is not tampered with by Mallory, then $(x, a) \leftarrow \mathsf{Read}(L, i)$ is such that $\mathsf{DBVer}(s, i, x, a)$ always accepts. Formally, for all possible update sequences $(D, (i_1, y_1), \ldots, (i_q, y_q))$, and for all $i^* \in [N]$, if

$$\begin{aligned}(L_0, s_0) &\leftarrow \mathsf{Setup}(D),\\ (L_1, s_1) &\leftarrow \mathsf{Write}(L_0, s_0, i_1, y_1),\\ &\vdots\\ (L_q, s_q) &\leftarrow \mathsf{Write}(L_{q-1}, s_{q-1}, i_q, y_q),\\ (x, a) &\leftarrow \mathsf{Read}(L_q, i^*),\end{aligned}$$

  then $\mathsf{DBVer}(s_q, i^*, x, a) = 1$, and $x = D[i^*]$ if Write was never called with index i^*, and otherwise $x = y_j$, where j is the largest index such that Write was called with index i_j such that $i_j = i^*$.
- *Soundness:* If L is tampered with by Clint's p.p.t. (in λ) officemate Mallory, then Clint should detect the tampering. Formally, for all possible sequences $(D, (i_1, y_1), \ldots, (i_q, y_q))$, and for all possible $i^* \in [N]$, if

$$\begin{aligned}(L_0, s_0) &\leftarrow \mathsf{Setup}(D),\\ (L_1, s_1) &\leftarrow \mathsf{Write}(L_0, s_0, i_1, y_1),\\ &\vdots\\ (L_q, s_q) &\leftarrow \mathsf{Write}(L_{q-1}, s_{q-1}, i_q, y_q),\\ (x, a) &\leftarrow \mathsf{Read}(L_q, i^*),\end{aligned}$$

  then Mallory should not be able to come up with (x', a'), $x' \neq x$ such that $\mathsf{DBVer}(s_q, i^*, x', a') = 1$ with non-negligible probability.

First, consider the case where Clint **never performs a write operation** (e.g. L is actually an Austen novel that he reads when he is bored).

(a) *(2 points)* Let $\mathsf{MAC} = (\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Ver})$ be an EUF-CMA secure MAC scheme where $\mathsf{Tag} : \{0,1\}^{2\log N} \to \{0,1\}^{\log^2 N}$. Suppose Clint defines the database verification scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}, \mathsf{DBVer})$ as follows. (For notational convenience, in this subpart only, we will rename Clint's original database from D to D_0.)

- $\mathsf{Setup}(D_0)$ outputs $s \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$. It also outputs $L = (D_0, T)$, where D_0 is the original database and T is the table whose ith entry is $\mathsf{Tag}(s, D[i] \| i)$.
- $\mathsf{Read}(L = (D, T), i)$ returns $(x = D[i], a = T[i])$.
- $\mathsf{DBVer}(s, i, x, a)$ outputs $\mathsf{Ver}(s,\ x \| i,\ a)$.

The correctness of this scheme is clear. Is the scheme secure? Either prove that soundness holds, or exhibit an attack.

Now, suppose Clint would like to both **read and write** to the database (i.e. when he is actually doing work because his advisor is around). In addition, he wants the secret state s output by Setup to be as long as a MAC key generated by $\mathsf{Gen}_{\mathsf{MAC}}$ plus at most $O(\log N)$ additional bits, so that it fits on his USB drive.

(b) *(2 points)* Suppose that Clint defines $\mathsf{Setup}, \mathsf{Read}, \mathsf{DBVer}$ the same way that he does in part (a). Suppose in addition that he defines Write as follows: $\mathsf{Write}(L = (D, T), s, i, y)$ outputs $L' = (D', T')$ such that the ith entry of D' is y, the ith entry of T' is $\mathsf{Tag}(s, y \| i)$, and every other entry of D' (resp. T') is equal to the corresponding entry in D (resp. T).

The correctness of this scheme is clear. Is the scheme secure? Either prove that soundness holds, or exhibit an attack.

Clint decides to come up with a different database verification scheme using *collision resistant hash functions* (CRHFs). Consider a collision-resistant family of hash functions $\mathcal{H} = \{h_k : \{0,1\}^{2\log^2 N} \to \{0,1\}^{\log^2 N}\}_{k \in \{0,1\}^\lambda}$. For this part, assume that the **length of the elements in the database D is** $\log^2 N$.

(c) *(2 points; a toy problem)* Suppose for now that Clint would only like to store *two* elements, y and y', each of size $\log^2 N$ bits. That is, suppose in this subpart that Clint's database D has 2 entries. (**In this part of the problem, we will go back to the read-only model.**) In addition, suppose that Clint's USB drive (which, recall, stores $s \leftarrow \mathsf{Setup}(D)$) can only store one CRHF key k and at most $\log^2 N$ additional bits. Using only CRHFs, construct a triple of algorithms $(\mathsf{TWOGen}, \mathsf{TWOTag}, \mathsf{TWOVer})$ and an algorithm DBVer that causes $\mathcal{S} = (\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}, \mathsf{DBVer})$ (with $\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}$ defined as follows) to satisfy correctness and soundness.

- $\mathsf{Setup}(D)$ (where $D = ((0, y), (1, y'))$) outputs $L = D$, $s = (k \leftarrow \mathsf{TWOGen}(1^\lambda), \sigma \leftarrow \mathsf{TWOTag}(k, y, y'))$.
- $\mathsf{Read}(L, i)$ outputs $(x = L[i], a = L[1 - i])$.
- $\mathsf{DBVer}(s = (k, \sigma), i, x, a) = \mathsf{TWOVer}(k, \sigma, t)$, where

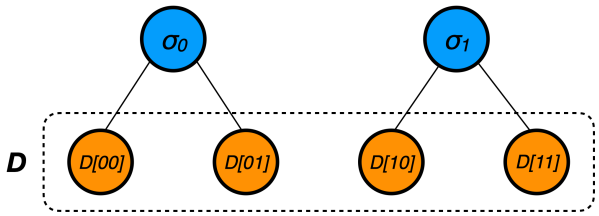
$$t = \begin{cases} (x, a) & i = 0 \\ (a, x) & i = 1. \end{cases}$$

In addition, prove the correctness and soundness of $\mathcal{S} = (\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}, \mathsf{DBVer})$.

(d) *(8 points; recurse!)* **In this part, we will return to the read-write model.** Now, assume that Clint would like to store a database D of size N, where each element consists of $\log^2 N$ bits. **In this part, we would like you to describe a scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Read}, \mathsf{Write}, \mathsf{DBVer})$, and prove the correctness and soundness properties of your scheme assuming only that CRHFs exist.** Your scheme should satisfy the following efficiency constraints:

- $(L, s) \leftarrow \mathsf{Setup}(D)$ should be such that $|s|$ contains one CRHF key and at most $\log^2 N$ additional bits.
- $(x, a) \leftarrow \mathsf{Read}(L, i)$ should be such that $|a| = (2 \log N - 2) \log^2 N$, and $|x| = \log^2 N$.
- $\mathsf{Write}(L, s, i, y)$ should modify at most $\log^3 N$ bits of L.

Hint. Consider the most naive solution which satisfies correctness and soundness: the one where Clint stores the entire database D, consisting of $N \cdot \log^2 N$ bits, on his USB drive. Of course, this solution does not satisfy the efficiency constraints. Perhaps we can, as a first step towards satisfying the efficiency constraints, come up with a solution in which Clint only stores about $(N/2) \cdot \log^2 N$ bits on his USB drive and still satisfies correctness and soundness. It may help you to find inspiration in the direction of making this first step if you look at the following picture.



You should identify the σs in this picture with the σ that your algorithm TWOTag from the previous problem outputs.

(e) (Open problem, ungraded of course.) Does there exist a database verification scheme where (using notation from the previous part) $|a|$ is only $O(|x|)$?