

Final Exam

*Deadline: December 21; 2022, 11:59 AM EST***Instructions**

- All submissions must be made via Gradescope. No late submissions will be accepted.
- Public posts on Gradescope will be disabled for the duration of this exam. You can still ask for clarifications via private posts on Piazza.
- Please add the following declaration on the first page of your submission:

*“I have neither given nor received any unauthorized aid on this exam. I understand that this exam must be taken without the aid of any other online resources **besides** the lecture slides/videos, resources posted on the course website and the handouts sent via email. The work contained herein is wholly my own. I understand that violation of these rules, including using an unauthorized aid or collaborating with another person/student, may result in my receiving a 0 on this exam.”*

1. **(10 points) Key-Exchange:** Consider the following key-exchange protocol:

- (a) Alice chooses $k, r \leftarrow \{0, 1\}^n$ at random, and sends $s = k \oplus r$ to Bob.
- (b) Bob chooses $t \leftarrow \{0, 1\}^n$ at random and sends $u = s \oplus t$ to Alice.
- (c) Alice computes $w = u \oplus r$ and sends w to Bob.
- (d) Alice outputs k and Bob computes $w \oplus t$.

Show that Alice and Bob output the same key. Show a concrete attack on the security of the scheme by an eavesdropping adversary.

2. **(15 points) Digital Signatures:** Let $(\text{Gen}, \text{Sign}, \text{Verify})$ be a UF-CMA secure digital signature scheme for signing n -bit messages, and let $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ be a collision-resistant hash function. Consider the following new signature scheme $(\text{Gen}', \text{Sign}', \text{Verify}')$ for signing arbitrary-length messages:

- $\text{Gen}'(1^n)$: $(sk, vk) \leftarrow \text{Gen}(1^\lambda)$ and output (sk, vk) .
- $\text{Sign}'(sk, m)$: Compute $\sigma \leftarrow \text{Sign}(sk, H(m))$. Output σ .
- $\text{Ver}'(vk, m, \sigma)$: Output 1 if $\text{Ver}(vk, H(m), \sigma)$ outputs 1, else output 0.

Prove that $(\text{Gen}', \text{Sign}', \text{Verify}')$ is also a UF-CMA secure digital signature scheme.

3. **(15 points) Oblivious Transfer and Public Key Encryption:** A two-message oblivious transfer between a receiver R and a sender S is defined by a tuple of 3 PPT algorithms $(\text{OT}_R, \text{OT}_S, \text{OT}_{\text{out}})$. The OT protocol works as follows (let λ be the security parameter):

- (a) **Receiver:** The receiver computes $(\text{msg}_R, \rho) \leftarrow \text{OT}_R(1^\lambda, b)$, where $b \in \{0, 1\}$ is the receiver's input. It sends msg_R to the sender.
- (b) **Sender:** The sender computes $\text{msg}_S \leftarrow \text{OT}_S(1^\lambda, \text{msg}_R, (m_0, m_1))$, where $m_0, m_1 \in \{0, 1\}^*$ are the sender's input. The sender sends msg_S to the receiver.
- (c) **Receiver's Output:** The receiver computes $m_b \leftarrow \text{OT}_{\text{out}}(\rho, \text{msg}_S)$.

This protocol satisfies the following properties:

- **Correctness:** For each $m_0, m_1 \in \{0, 1\}^*$, $b \in \{0, 1\}$, it holds that

$$\Pr \left[\begin{array}{c} (\rho, \text{msg}_R) \leftarrow \text{OT}_R(1^\lambda, b) \\ \text{msg}_S \leftarrow \text{OT}_S(1^\lambda, \text{msg}_R, (m_0, m_1)) \end{array} \middle| \text{OT}_{\text{out}}(\rho, \text{msg}_R, \text{msg}_S) = m_b \right] = 1,$$

- **Security against Semi-Honest Sender:** It holds that,

$$\left\{ (\text{msg}_R^0, \rho^0) \leftarrow \text{OT}_R(1^\lambda, 0) \mid \text{msg}_R^0 \right\} \approx_c \left\{ (\text{msg}_R^1, \rho^1) \leftarrow \text{OT}_R(1^\lambda, 1) \mid \text{msg}_R^1 \right\}$$

- **Security against Semi-Honest Receiver:** It holds that for each $b \in \{0, 1\}$, $m_0, m_1, m'_0, m'_1 \in \{0, 1\}^*$, and $m_b = m'_b$,

$$\left\{ \text{OT}_S(1^\lambda, \text{msg}_R, (m_0, m_1)) \right\} \approx_c \left\{ \text{OT}_S(1^\lambda, \text{msg}_R, (m'_0, m'_1)) \right\}$$

where $(\text{msg}_R, \rho) \leftarrow \text{OT}_R(1^\lambda, b)$.

Public-Key Encryption Scheme: Now consider the following construction of a public-key encryption scheme:

- **KeyGen(1^λ):** Compute $(\text{msg}_R, \rho) \leftarrow \text{OT}_R(1^\lambda, b)$. Set $pk = \text{msg}_R$ and $sk = \rho$. Output (pk, sk) .
- **Enc(pk, m):** Compute $\text{msg}_S \leftarrow \text{OT}_S(1^\lambda, pk, (m, m))$ and set $c = \text{msg}_S$. Output ciphertext c .
- **Dec(sk, c):** Compute and output $m \leftarrow \text{OT}_{\text{out}}(sk, c)$.

Prove that the above construction (KeyGen, Enc, Dec) is an IND-CPA secure public-key encryption.

4. (10 points) **Pseudo-random Functions** Consider the following notion of “Restricted PRFs” consisting of three algorithms:

- **Gen(1^n):** is PPT algorithm that samples a PRF key $K \xleftarrow{\$} \{0, 1\}^n$.
- **Restrict(K, x):** is PPT algorithm that takes as input a PRF key K and a point x in the input space of the PRF and outputs a restricted key K_x .
- **Eval(K_x, x'):** is a deterministic polynomial time algorithm that takes as input a restricted key K_x and an input x' and outputs an element y .

We require two properties:

- **Restricted Correctness:** on any input point $x' \neq x$, PRF evaluation using the restricted key K_x yields the same result as PRF evaluation using the (unrestricted) key K .
- **Restricted Pseudorandomness:** the output of the PRF on input x looks pseudorandom to any PPT adversary, even if the restricted key K_x is given to the adversary.

Construction of Restricted PRF:

Recall the binary tree based construction of PRFs discussed in class. Let $G : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ be a length doubling pseudorandom generator. We can imagine $G(s) = G_0(s) \| G_1(s)$, where $G_0(s)$ simply outputs the first n -bits of $G(s)$ and $G_1(s)$ outputs

the last n -bits of $G(s)$. Let $x = x_1 \| x_2 \| \dots \| x_n$ be an n -bit input. As discussed in class, the PRF $F_K(x) = G_{x_n}(G_{x_{n-1}}(\dots(G_{x_1}(K))))$.

The PRF $F_K(x)$ can also be viewed as a binary tree of size 2^n . The root corresponds to the PRF key K . The nodes at the first level are $K_0 = G_0(K)$ and $K_1 = G_1(K)$. Similarly, the nodes at the second level are $K_{00} = G_0(K_0)$, $K_{01} = G_1(K_0)$, $K_{10} = G_0(K_1)$ and $K_{11} = G_1(K_1)$. Nodes at the remaining levels can be defined in a similar way. The leaf nodes correspond to the outputs of the PRF.

A restricted PRF can also be designed based on the above tree as follows:

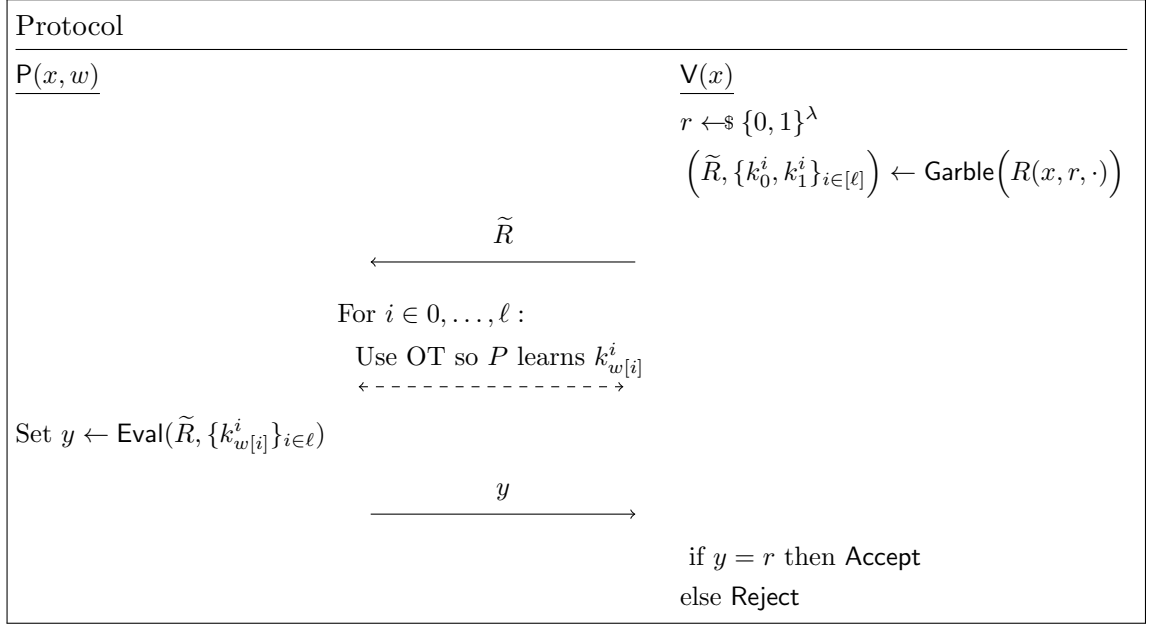
- **Gen(1^n)**: Sample a random $K \xleftarrow{\$} \{0,1\}^n$.
- **Restrict(K, x)**: Let $x = x_1 \| x_2 \| \dots \| x_n$ and let **Sibling(y)** denote the sibling of node y in the above binary tree. $K_x = \text{Sibling}(K_{x_1}) \| \text{Sibling}(K_{x_1 x_2}) \| \dots \| \text{Sibling}(K_{x_1 \dots x_n}) \| x$.
- **Eval(K_x, x')**: Parse $K_x = \text{Sibling}(K_{x_1}) \| \text{Sibling}(K_{x_1 x_2}) \| \dots \| \text{Sibling}(K_{x_1 \dots x_n}) \| x$. Let $x = x_1 \| x_2 \| \dots \| x_n$ and $x' = x'_1 \| x'_2 \| \dots \| x'_n$. Find the largest ℓ such that $x_1 \| x_2 \| \dots \| x_\ell = x'_1 \| x'_2 \| \dots \| x'_\ell$. Output $G_{x'_n} \left(G_{x'_{n-1}} \left(\dots \left(G_{x'_{\ell+2}}(\text{Sibling}(K_{x_1 \dots x_{\ell+1}})) \right) \right) \right)$.

In the above discussion we showed how to define the syntax and security of a PRF that is restricted on a single input and also gave a construction for such a PRF. We can also consider PRFs that are further restricted on more than 1 inputs. The syntax of **Restrict** and **Eval** algorithms for such PRFs and their correctness and pseudorandomness properties can be defined in a similar way.

- (a) Show how you can restrict the key in the above tree-based construction such that the PRF is restricted on all inputs with a fixed 3-bit prefix, i.e., on inputs of the form $x = a_1 \| a_2 \| a_3 \| x_4 \| \dots \| x_n$, where $a_1 \| a_2 \| a_3$ is a fixed value.
 - (b) Show how you can restrict the key in the above tree-based construction such that the PRF is restricted on all inputs $X \leq x \leq Y$, where $X < Y$ and $X, Y \in \{0,1\}^n$ are fixed values.
5. **(10 points) Hash Functions** We saw in the homework that an *order-preserving* encryption scheme would be hard to achieve. Here we will show something similar for *hash functions*. Suppose a function $H : \{0,1\}^{2n} \mapsto \{0,1\}^n$ has the following property. For each $x, y \in \{0,1\}^{2n}$, if $x \leq y$, then $H(x) \leq H(y)$. Show that H is not collision resistant (describe how to efficiently find a collision in such a function).
6. **(15 points) Zero-Knowledge** Consider the following protocol to prove that $x \in L$, for some language L . Let R be a associated relation (viewed as a circuit) such that $R(x, w) = 1$ if and only if w is a witness for the fact that $x \in L$. Let $R(x, r, w)$ denote the circuit such that $R(x, w) = r$ if and only if w is a valid witness.

The verifier samples a random string r and constructs a garbled version of the circuit $R(x, r, \cdot)$, with r and the statement x fixed. It sends the garbled circuit to the prover. The prover and the verifier then run a series of OTs that allow the prover to learn the wire labels corresponding to w . The prover then uses the labels to evaluate the circuit, and sends the output to the verifier. Finally, the verifier accepts if it received r , and rejects otherwise.

The full description is given below. For simplicity, assume that all valid witnesses are of the same length ℓ . Let $w[i]$ denote the i -th bit of a witness w .



- (a) Prove that the above protocol satisfies correctness and soundness, with soundness error negligible in λ .
- (b) Explain why the above protocol is zero-knowledge for a *semi-honest* verifier.
- (c) Explain why the above protocol is **not** zero-knowledge for a *malicious* verifier.

For (b) and (c), it is sufficient to explain in words why the above protocol is or is not zero-knowledge.