# Homework 9

**Due: December 01, 2023**

CS 1510: Intro. to Cryptography and Computer Security

## 1 SWHE and FHE Discussion

The following are reflection questions that you may answer informally, with your understanding and intuition rather than mathematical arguments and proofs.

a. What is one potential real-world use-case of somewhat or fully homomorphic encryption? Precisely describe the participants and purpose in the SWHE or FHE application: who owns the data and who is computing on the data? Why is this setup useful?

b. When it comes to practical applications of computing privately over large datasets, it is often desirable to aggregate data from multiple sources or data owners, each with their own secret key. Consider the GSW scheme from class. Is it feasible to instantiate this cryptosystem such that two different key holders can homomorphically compute over their ciphertexts? Why or why not?

## 2 Random Oracles and RSA-FDH Discussion

The following are reflection questions that you may answer informally, with your understanding and intuition rather than mathematical arguments and proofs.

a. What is the fundamental modeling difference between a hash function and a random oracle? Specifically, where in the security reduction would the hash function exist, vs. where does the random oracle exist?

b. Why is it necessary to replace the hash function in RSA-FDH with a random oracle in order for the security reduction to go through?

c. What security properties would we need from the hash function in RSA-FDH to avoid the attacks discussed on Slide 16 in Lecture 19? How are these properties guaranteed by the random oracle?

d. In practice, "oracles" do not exist—participants run cryptographic protocols internally. Random oracles, for example, are instantiated with hash functions. What does this mean for the practical security of the cryptographic protocols like RSA-FDH,

Schnorr's identification scheme, and zero knowledge proofs, which are widely used in practice?

## 3 Digital Signatures

Let $\mathcal{S} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a secure digital signature scheme for $k$-bit messages.

Consider the following three signature schemes created using $\mathcal{S}$. Each will use the original $\mathsf{Gen}$ algorithm, but provide modified algorithms $\mathsf{Sign}'$ and $\mathsf{Vrfy}'$ that sign and verify variable-length messages that can be larger than $k$-bits.

1. **Scheme 1:** $\mathcal{S}^1 = (\mathsf{Gen}, \mathsf{Sign}^1, \mathsf{Vrfy}^1)$. For a given message, $M$, let $M = m_1 \| m_2 \| \ldots \| m_n$, such that each $m_i$ is of length $k$. Note that if $M$ is not a multiple of $k$, then we will pad the end of $M$ with extra 0s. Let $\mathsf{Sign}^1(sk, M) = (\mathsf{Sign}(sk, m_1),\ \mathsf{Sign}(sk, m_2),\ \ldots,\ \mathsf{Sign}(sk, m_n))$. Thus, the output of $S^1$ is a vector of signatures, $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$. Define $\mathsf{Vrfy}^1$ canonically:
   $\mathsf{Vrfy}^1(pk, M, \sigma) = (\mathsf{Vrfy}(pk, m_1, \sigma_1),\ \mathsf{Vrfy}(pk, m_2, \sigma_2),\ \ldots,\ \mathsf{Vrfy}(pk, m_n, \sigma_n)) \stackrel{?}{=} 1^n$
   This means checking each $\sigma$ with the corresponding message and outputting 1 only if all sub-verifications output 1.

2. **Scheme 2:** $\mathcal{S}^2 = (\mathsf{Gen}, \mathsf{Sign}^2, \mathsf{Vrfy}^2)$. For a given message, $M$, choose the smallest $n$ such that $\lceil \log_2(n+1) \rceil + \lceil \frac{|M|}{n} \rceil \leq k$. (Assume that $|M|$ is small enough and $k$ is large enough to make this is possible.) Then break $M$ up into $M = m_1 \| \ldots \| m_n$, where each $m_i$ is such that $|m_i| = k - \lceil \log_2(n+1) \rceil$, and $m_n$ is padded with 0s as necessary.

   Let $\mathsf{Sign}^2(sk, M) = (\mathsf{Sign}(sk, 1\|m_1),\ \mathsf{Sign}(sk, 2\|m_2),\ \ldots,\ \mathsf{Sign}(sk, n\|m_n))$, where each index $i$ is represented using $\lceil \log_2(n+1) \rceil$ bits. Again, define $\mathsf{Vrfy}^2$ canonically:
   $\mathsf{Vrfy}^2(pk, M, \sigma) = (\mathsf{Vrfy}(pk, 1\|m_1, \sigma_1),\ \mathsf{Vrfy}(pk, 2\|m_2, \sigma_2),\ \ldots,\ \mathsf{Vrfy}(pk, n\|m_n, \sigma_n)) \stackrel{?}{=} 1^n$.

3. **Scheme 3:** For a given message, $M$, choose the smallest $n$ such that $2\lceil \log_2(n+1) \rceil + \lceil \frac{|M|}{n} \rceil \leq k$. (Assume that $|M|$ is small enough and $k$ is large enough to make this is possible.) Then break $M$ up into $M = m_1 \| \ldots \| m_n$, where each $m_i$ is such that $|m_i| = k - 2\lceil \log_2(n+1) \rceil$, and $m_n$ is padded with 0s as necessary.

   Let $S^3(sk, M) = (\mathsf{Sign}(sk, n\|1\|m_1),\ \mathsf{Sign}(sk, n\|2\|m_2),\ \ldots,\ \mathsf{Sign}(sk, n\|n\|m_n))$, where each index is represented using $\lceil \log_2(n+1) \rceil$ bits. Let $\mathsf{Vrfy}^3$ be defined canonically:

$$\mathsf{Vrfy}^3(sk, M, \sigma) =$$

$$(\mathsf{Vrfy}(sk, m_1, n\|1\|m_1),\ \mathsf{Vrfy}(sk, m_2, n\|2\|m_2),\ \ldots,\ \mathsf{Vrfy}(sk, m_n, n\|n\|m_n)) \stackrel{?}{=} 1^n$$

a. Show that, by issuing just one query to the signer in **Scheme 1**, the adversary can succeed in forging a signature on a message of its choosing; find an attack that breaks **Scheme 1** but not **Scheme 2**. This can be informal.

b. Show that **Scheme 2** is still broken: by issuing just a single signing query to the signer in **Scheme 2**, the adversary can succeed in forging a signature on a message of its choosing; find an attack that breaks **Scheme 2** but not **Scheme 3**. This too can be informal.

c. Show that **Scheme 3** is still broken: by issuing two signing queries to the signer in **Scheme 3**, the adversary can still succeed in forging a signature on a message of its choosing as long as this message is long enough (how long does it need to be?). Again, this can be informal.

d. Give a scheme that is based on **Scheme 3**, but is in fact secure (You may NOT use CRHFs as part of your construction; the only building block you're given is the signature scheme for $k$-bit messages. You may assume that $k$ is sufficiently large—at least in the order of the security parameter.) Explain why your scheme fixes the vulnerability that is exhibited by **Scheme 3**, and prove it secure.

## 4 Signatures: From Weak to Strong

A signature scheme is **weakly secure** if the probability that a ppt adversary wins the following game is negligible:

**Signing query:** On input $1^k$, the adversary chooses the messages $M_1, \ldots, M_n$ to be signed.

**Response:** The signer runs the key generation and the signing algorithms and sends to the adversary the public key $pk$ and the signatures $\{\sigma_j\}_{j=1}^n$ on the adversary's messages.

**Forgery:** The adversary outputs a message $M^*$ and a signature $\sigma^*$ and wins the game if $M^*$ was not included in its signing query, and yet the verification algorithm accepts the signature $\sigma^*$.

The key difference for a weakly secure scheme being that the adversary must submit their messages all at once rather than adaptively asking for messages to be signed (i.e. they submit messages to be signed one-by-one and can thus use responses from previous queries to inform their next message query).

A signature scheme is **one-time secure** if the probability that a ppt adversary wins the following game is negligible:

**Key generation:** The challenger runs the key generation algorithm and generates $(pk, sk)$.

**Signing query:** On input $pk$, the adversary chooses a message $m$ to be signed.

**Response:** The challenger computes a signature $\sigma$ on $m$ using the signing algorithm, and returns it to the adversary.

**Forgery:** The adversary outputs a message $m^*$ and a signature $\sigma^*$ and wins the game if $m^* \neq m$, and yet the verification algorithm accepts the signature $\sigma^*$.

Construct a secure signature scheme given a weakly secure signature scheme $(\mathsf{Gen}_{\mathsf{weak}}, \mathsf{Sign}_{\mathsf{weak}}, \mathsf{Vrfy}_{\mathsf{weak}})$ and a one-time secure signature scheme $(\mathsf{Gen}_{\text{one-time}}, \mathsf{Sign}_{\text{one-time}}, \mathsf{Vrfy}_{\text{one-time}})$. Let the message space and key space for all the signature schemes here be binary strings of length $k$, the security parameter, and let the weakly secure scheme sign $n = p(k)$ messages for any polynomial $p$. Don't forget to prove that your construction is correct and secure.

## 5 Deterministic Digital Signature

Assume the existence of a digital signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ for which $\mathsf{Sign}$ is a probabilistic algorithm. Construct a digital signature scheme $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Verify}')$ where $\mathsf{Sign}'$ is deterministic.

*Hint: You may also assume the existence of one-way functions (and other symmetric-key primitives implied by one-way functions) as it is implied by the existence of digital signatures.*

## 6 Zero Knowledge Proofs

In class we have seen ZK protocols for proving membership in 3-colorability. Let us design a proof for a different NP-complete problem.

**Definition 1 (Vertex Cover)** *Given a graph $G = (V, E)$, a vertex cover of size $k$ is a subset $C \subseteq V$ such that $|C| = k$ and that for all edges $e \in E$ at least one endpoint is in $C$. Consider the language* VertexCover *defined as follows:*

$$\textsc{VertexCover} = \{(G, k) \ : \ G \text{ has a vertex cover of size } k\}$$

Suppose the prover Alice and the verifier Bob share a graph $G$ and the description of a cryptographic commitment algorithm $\mathsf{Com}$. Alice claims that $G$ has a size $k$ vertex cover. Alice will attempt to prove it as follows.

- We assume that each vertex of the original graph is labeled with a unique number from 1 to $n$ (such that $n = |V|$) known to both the prover Alice and the verifier Bob. Alice permutes the set of vertices using a random permutation $\pi : [n] \to [n]$, where we denote the permuted set $V' = \{\pi(1), \ldots, \pi(n)\}$. Alice computes a list of commitments to the permuted vertices $C_{V'} = \{\mathsf{Com}(\pi(1)), \mathsf{Com}(\pi(2)), \ldots, \mathsf{Com}(\pi(n))\}$.
- Define a new set of edges $E'$ such that for each original edge $e = (u, v) \in E$ between two vertices $u$ and $v$, there exists an edge $e' = (\pi(u), \pi(v))$ between the permuted vertices $\pi(u)$ and $\pi(v)$. Alice computes commitments to all the edges in $E'$, namely $\{\mathsf{Com}(u', v') \mid e' = (u', v') \in E'\}$, and then randomly permutes these commitments to obtain a set $C_{E'}$.
- Finally, for all $v' \in V'$, Alice proceeds as follows: If $\pi^{-1}(v') \in C$, then set $b_{v'} := 1$; otherwise set $b_{v'} := 0$. Alice computes a list of commitments to the permuted indicators $C_{B'} = \{\mathsf{Com}(b_1), \mathsf{Com}(b_2), \ldots, \mathsf{Com}(b_n)\}$.

How does the rest of this protocol work? We're going to divide the solution into several small steps.

a. Suppose Bob only wanted to check that the graph is represented correctly (i.e. that the graph is the same one, $G$, that he and Alice have agreed on.) How can he confirm this without learning any other information?

b. Now suppose Bob only wants to check that the cover has the appropriate size (the agreed upon $k$). How can he confirm this without learning any other information?

c. Finally, suppose Bob only wants to check that each edge is covered. How can Bob do this without learning any other information (he is allowed to examine only one edge in each round)? With what probability is he guaranteed to catch Alice if she does not know a $k$-cover?

d. What should Bob's overall strategy for verifying Alice's vertex cover proof be, and what is the minimum probability that Alice will be caught if she cheats? (Hint: the strategy will probably have to be randomized.)

e. Assuming Alice is willing to repeat this process as many times as necessary, how many times should Bob run this algorithm so that if Alice cheats she will be caught with probability at least $\Omega(1)$?

f. Give an informal sketch for how the zero-knowledge simulator for this proof system might work.

## 7 Summary Question

Summarize the most important insights from this week's material, including from the lectures, notes, textbooks, homework problems, and other resources you find helpful, into a one-page resource. You will be permitted to use this one-page resource (along with the other weeks' resources) on the midterm and final.

Changes to this document prior to the exams are permitted, but for each change, you will be asked to state what you changed and why. For example, if you dropped something and replaced it with something else, justify why the thing you dropped wasn't as important as the thing you inserted, why you think it might be more useful for the exam, etc.

Please note that the purpose of this question is to help you organize and synthesize the material for your own future use. It will be graded based on completion—we will not be checking it for correctness.