Massachusetts Institute of Technology
6.5610: Applied Cryptography and Security
Professors Henry Corrigan-Gibbs and Yael Tauman Kalai

Handout 3
March 14, 2023
Due: March 24, 2023

Problem Set 3

This problem set is due on *Friday, March 24, 2023* at **4:59 PM**. Please note our late submission penalty policy in the course information handout. Please submit your problem set, in PDF format, on Gradescope. *Each problem should be in a separate page.*

You are to work on this problem set in groups. For problem sets 1, 2, and 3, we will randomly assign the groups for the problem set. After problem set 3, you are to work on the following problem sets with groups of your choosing of size three or four. If you need help finding a group, try posting on Piazza. See the course website for our policy on collaboration. Each group member must independently write up and submit their own solutions.

Homework must be typeset in LaTeXand submitted electronically! Each problem answer must be provided as a separate page. Mark the top of each page with your group member names, the course number (6.5610), the problem set number and question, and the date. We have provided a template for LaTeX on the course website (see the *Psets* tab at the top of the page).

With the authors' permission, we may distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this in your profile on your homework submission.

Problem 3-1. Hash functions

In this problem, we will explore which properties of hash functions imply other properties. In this problem we consider the definition given in class of a hash function as a family of keyed functions (see page 3 of the notes for Lecture 11), where for every security parameter $k \in \mathbb{N}$ and for every $\mathsf{hk} \leftarrow \{0,1\}^k$ there exists a hash function $H(\mathsf{hk}, \cdot) : \{0,1\}^* \to \{0,1\}^k$. Decide whether each of the following is True or False, and explain why.

(a) Any hash function that is collision resistant is also target collision resistant.

(b) Any hash function that is target collision resistant is also one way, where a keyed hash function is said to be one way if given a randomly chosen key $\mathsf{hk} \leftarrow \{0,1\}^k$ and a hash value $H(\mathsf{hk}, x)$ for a randomly chosen $x \leftarrow \{0,1\}^k$, it is hard to find x' such that $H(\mathsf{hk}, x) = H(\mathsf{hk}, x')$ (with non-negligible probability).

(c) Any hash function that is one way is also collision resistant.

Let H be family of keyed hash functions that are collision resistant, where for every security parameter $k \in \mathbb{N}$ and every $\mathsf{hk} \leftarrow \{0,1\}^k$, $H(\mathsf{hk}, \cdot) : \{0,1\}^* \to \{0,1\}^k$. For each of the following functions H' determine if H' is necessarily a collision resistant. If so, explain in 1-3 sentences why it is collision resistant, and if not, provide a counter example.

(d) $H'(\mathsf{hk}, x) = H(\mathsf{hk}, H(\mathsf{hk}, x))$.

(e) $H'(\mathsf{hk}, x) = H(\mathsf{hk}, x) || 0$.

(f) $H'(\mathsf{hk}, x) = H(\mathsf{hk}, x)[0 : \lfloor k/2 \rfloor]$, where $y[0 : d]$ is the first d bits of y.

Problem 3-2. Hash-based signatures

In this problem, we will construct a signature scheme using only a hash function. We will work over a message space $\mathcal{M} = \{0,1\}^n$. We take $H : \{0,1\}^* \to \{0,1\}^k$ to be a hash function. Then, we define the following signature scheme:

- $\mathsf{Gen}(1^k) \to (\mathsf{sk}, \mathsf{pk})$: The key generation algorithm samples $2n$ random elements $x_{0,1}, x_{1,1}, \ldots, x_{0,n}, x_{1,n}$ in $\{0,1\}^k$ and outputs them to be the secret key:

  $$\mathsf{sk} = \begin{pmatrix} x_{0,1} & x_{0,2} & x_{0,3} & \cdots & x_{0,n} \\ x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \end{pmatrix} \in \{0,1\}^{k \cdot 2n}.$$

  It outputs the hashes of these random elements as the public key:

  $$\mathsf{pk} = \begin{pmatrix} y_{0,1} = H(x_{0,1}) & y_{0,2} = H(x_{0,2}) & y_{0,3} = H(x_{0,3}) & \cdots & y_{0,n} = H(x_{0,n}) \\ y_{1,1} = H(x_{1,1}) & y_{1,2} = H(x_{1,2}) & y_{1,3} = H(x_{1,3}) & \cdots & y_{1,n} = H(x_{1,n}) \end{pmatrix} \in \{0,1\}^{k \cdot 2n}.$$

- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$: The signature algorithm outputs a signature σ on any message $m \in \mathcal{M}$ that is computed as

  $$\sigma = (x_{m_1,1}, x_{m_2,2}, x_{m_3,3}, \ldots, x_{m_n,n}) \in \{0,1\}^{k \cdot n},$$

  where we write m_i to denote the i-th bit in the binary representation of m.

- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to \mathsf{Accept}/\mathsf{Reject}$: The verification algorithm outputs Accept if and only if the following equality holds for all $1 \leq i \leq n$:

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

  where we write m_i to denote the i-th bit in the binary representation of m and σ_i to denote the i-th element in σ.

In the first part of this problem, we will show that this is a *one-time* secure signature scheme.

(a) Alice runs the key generation algorithm $\mathsf{Gen}(1^k) \to (\mathsf{sk}_A, \mathsf{pk}_A)$ and publishes her public key, pk_A. An adversary learns only Alice's public key pk_A, and does not see any signatures. What is the minimal assumption we need to assume about H (from the following 4 options) to ensure that the adversary cannot forge a signature to any message $m \in \{0,1\}^n$:

1. One-way function
2. Target collision resistant function
3. Collision resistant function
4. Random Oracle Model

Explain your answer.

(b) Next, assume that H is secure as in part (a). Suppose the adversary sees Alice's public key pk_A and a *single* message/signature pair (m, σ) that Alice signed with her corresponding secret key, sk_A, i.e.,

$$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_A, m).$$

Can the adversary forge a signature σ' on some new message $m' \neq m$ such that $\mathsf{Verify}(\mathsf{pk}_A, m', \sigma')$ outputs Accept (i.e., m' looks like it was signed by Alice)? Explain your answer in a few sentences.

(c) Now, suppose that the adversary learns Alice's public key, pk_A, and later also sees two signatures σ_1 and σ_2, each produced by Alice, on two messages m_1 and m_2 *of its choosing*. In other words, for $i = 1, 2$, m_i is chosen by the adversary and

$$\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}_A, m_i).$$

Explain how the adversary should choose m_1 and m_2 to then be able to forge a valid signature σ' on *any* message $m' \in \mathcal{M}$ (even if $m' \neq m_1$ and $m' \neq m_2$) such that $\mathsf{Verify}(\mathsf{pk}_A, m', \sigma')$ outputs Accept (i.e., σ' looks like it was generated by Alice).

We have just proved that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a one-time secure signature scheme but is not secure if the adversary sees signatures to multiple messages. Next, we will boost this scheme to be secure even if Alice signs two messages of arbitrary length.

(d) Show how to use a collision resistant hash function $H' : \{0,1\}^* \to \{0,1\}^n$ to convert the one-time signature scheme described above into a one-time signature scheme for signing messages of *unbounded* length, i.e., messages in $\{0,1\}^*$.

Hint: The size of secret keys and public keys should not change.

(e) Now, show how to convert the one-time signature scheme from part (d) into a *two*-time signature scheme for signing messages of unbounded length, i.e., messages in $\{0,1\}^*$. That is, construct a signature scheme that is secure even if the adversary sees signatures on *two* messages of its choosing. For the purposes of this problem, the Sign and Verify algorithms may be stateful; i.e., they can behave differently when signing the first message and when signing the second message and they may remember some information between invocations. However, we want the size of the secret key (that Alice stores) and of the public key (that Alice publishes) to be the same as in part (d).

Hint: The size of secret keys and public keys should again not change. It may be useful to sign a public key.

Problem 3-3. Commitments
Recall the notion of a commitment scheme: a cryptographic "safe" that allows one party to put some information in the safe and then later open it in a binding way; see the lecture notes for the precise definition. In class, we saw a hash-based commitment scheme that is secure assuming the hash function is modelled as a random oracle. In this problem, we will construct a group-based commitment scheme. Our scheme is associated with public parameters (in our hash-based construction you can think of the hash key hk as the public parameter).

Let $\mathbb{G}$ be a group of large prime order q, and let g, h be randomly chosen elements of $\mathbb{G}$. Define a commitment scheme for messages in $\mathbb{Z}_q$, as follows:

$$\mathsf{Com}_{g,h}(x, r) = g^x h^r,$$

where we think of (g, h) as public parameters associated with our commitment scheme. Namely, to commit to a message $x \in \mathbb{Z}_q$, draw random value $r \leftarrow \mathbb{Z}_q$ and output $g^x h^r$ (where the multiplication is in the group $\mathbb{G}$). In order to open the commitment, reveal x and r. For the following parts explain your answer in a few sentences.

(a) Is this commitment scheme computationally hiding or statistically hiding? If it is not statistically hiding, then under what assumption is it computationally hiding?

(b) Is this commitment scheme computationally binding or statistically binding? If it is not statistically binding, then under what assumption is it computationally binding?

Problem 3-4. CRIME attack In the real world, messages can vary in length. However, information theory means that encryption cannot generally hide the size of messages. In this problem, we will look at the CRIME attack (`https://en.wikipedia.org/wiki/CRIME`), which exploits "compress then encrypt."

For compression, we will use the gzip compression algorithm. This algorithm works by replacing repeated sequences of (3 or more) characters with "pointers" to previous repetitions. This means that if there is a large sequence of repeated characters, the compressed size will be smaller. If a secret and some text are compressed together, then the compressed result will be shorter when there is a repeated sequence of characters between the text and secret. A shorter message will then result in a shorter ciphertext, which can be observed by the adversary.

Use this idea to discover the secret held on `leaky.csail.mit.edu/encrypt`. You can send messages to the server with `curl "leaky.csail.mit.edu/encrypt?kerb=test&msg=hello-world"` , or use the provided code in `client.py`.

The skeleton code file implements a local version of the server which will be invoked by default from `client.py`. Use this local version to test your code before attacking the real server (which is rate-limited). It is recommended that you create a virtual environment to install the cryptography library for running the local server. You can find a tutorial at `https://www.geeksforgeeks.org/python-virtual-environment/`. This code will also help you to develop your attack; in particular note how the secret and your message are put together. When you are ready for the real server, set the `LOCAL` variable to False.

The secret consists of ASCII characters from the space character to the lowercase 'z' character (numbers 32 to 122 inclusive) and is 14 characters in length. Please provide your kerberos ID as the `kerb` parameter to the server. Different IDs may have different secrets. Provide the secret you find as the answer to this question, and attach your code on gradescope.