Name: Harris Christiansen (christih@purdue.edu)

# Homework 3

1. **Analyzing a new MAC.** Suppose $\{F_1, \ldots, F_\alpha\}$ is family of pseudorandom functions from $\{0,1\}^n$ to $\{0,1\}^{n/100}$.

   Consider the following MAC Scheme for message $m \in \{0,1\}^{tn}$, for some constant natural number $t \geqslant 2$.

   (a) $\mathsf{Gen}()$: Return $\mathsf{sk} \xleftarrow{\$} \{1, 2, \ldots, \alpha\}$

   (b) $\mathsf{Mac}_{\mathsf{sk}}(m)$: Interpret the message $m = (m_1, m_2, \ldots, m_t)$, where each $m_i \in \{0,1\}^n$ and $1 \leqslant i \leqslant t$. Define $\tau_i = F_{\mathsf{sk}}(m_i)$, for each $1 \leqslant i \leqslant t$. Return $\tau = (\tau_1, \tau_2, \ldots, \tau_t)$.

   (c) $\mathsf{Ver}_{\mathsf{sk}}(m, \tau)$: Interpret $m = (m_1, \ldots, m_t)$ and $\tau = (\tau_1, \ldots, \tau_t)$, where each $m_i \in \{0,1\}^n$ and $\tau_i \in \{0,1\}^{n/100}$. Return $\mathsf{true}$ if and only if $F_{\mathsf{sk}}(m_i) = \tau_i$, for all $1 \leqslant i \leqslant t$.

   (a) Prove that the above MAC scheme is not secure for $m \in \{0,1\}^{tn}$.

   (b) Prove that the above MAC scheme preserves message integrity for $m \in \{0,1\}^{tn}$.

   **Solution.**

   (a) For this MAC scheme to be secure, it should not be possible to determine the MAC for new messages that have not before been sent. However, since the pseudorandom functions map to a range 1/100 the size of the domain, it is therefor possible to determine a new message, $m_2$ that has the same MAC as a given message $m_1$, by swapping blocks that have equivalent MAC blocks.
   The MAC scheme is not secure because $P[Sign_{sk}(m') = \tau' | Sign_{sk}(m) = \tau] \neq \frac{1}{|T|}$

   (b) Message integrity is still preserved because of the low probability of mutating the message in a malicious and non-destructive way.
   $P[Sign_{sk}(m') = \tau' | Sign_{sk}(m) = \tau] \leqslant \frac{1}{|T|}$

2. **Designing a New MAC Scheme.** We shall work over the field $(\mathbb{Z}_p, +, \times)$, where $p$ is a prime number. Consider the MAC scheme defined by the $(\mathsf{Gen}, \mathsf{Mac}, ver)$ algorithms below for message $\mathbb{Z}_p^\ell$, where $\ell \geqslant 1$ is a constant integer.

$\mathsf{Gen}()$ :

(a) Sample $k_1 \xleftarrow{\$} \mathbb{Z}_p$ and $k_2 \xleftarrow{\$} \mathbb{Z}_p$

(b) Return $\mathsf{sk} = (k_1, k_2)$

$\mathsf{Mac}_{\mathsf{sk}=(k_1,k_2)}(m)$:

(a) Interpret $m = (m_1, m_2, \ldots, m_\ell)$, where each $m_i \in \mathbb{Z}_p$

(b) Let $\tau = k_1 + m_1 k_2 + m_2 k_2^2 + \cdots + m_\ell k_2^\ell$

(c) Return $\tau$ as the tag for the message $m$

$\mathsf{Ver}_{\mathsf{sk}=(k_1,k_2)}(m)$:

(a) Interpret $m = (m_1, m_2, \ldots, m_\ell)$, where each $m_i \in \mathbb{Z}_p$

(b) Return whether $\tau$ is identical to $k_1 + m_1 k_2 + m_2 k_2^2 + \cdots + m_\ell k^\ell$

(a) Given a message $m = (m_1, m_2, \ldots, m_\ell)$ and its tag $\tau$ what is the maximum probability that a different message $m' = (m'_1, m'_2, \ldots, m'_\ell)$ that has the same tag $\tau$?

(b) Given a message $m = (m_1, m_2, \ldots, m_\ell)$ and its tag $\tau$ what is the maximum probability that a different message $m' = (m'_1, m'_2, \ldots, m'_\ell)$ and $\tau'$ as its valid tag?

(*Remark*: You will need to use Schwartz-Zippel Lemma to compute the probability.)

**Solution.**

(a) The maximum probability that a different message m' has the same tag $\tau$ as message m is $P[\tau - \tau' = 0]$.

By Schwartz-Zippel Lemma, $P[\tau - \tau' = 0] \leqslant \frac{deg(\tau - \tau')}{|Z_p|}$

$\frac{deg(\tau-\tau')}{|Z_p|} = \frac{deg((k_1+m_1k_2+m_2k_2^2+\cdots+m_\ell k_2^\ell)-(k_1+m'_1k_2+m'_2k_2^2+\cdots+m'_\ell k_2^\ell))}{|Z_p|}$

$= \frac{deg((m_1-m'_1)k_2+(m_2-m'_2)k_2^2+\cdots+(m_\ell-m'_\ell)k_2^\ell)}{|Z_p|} = \frac{l}{p}$

(b) The probability that a different message m' has a different tag $\tau'$ than message m is the same, $\frac{l}{p}$

3. **New Pseudorandom Function Family.** In the lectures, we saw the following GGM construction for pseudorandom functions. Given a length-doubling PRG $G\colon \{0,1\}^B \to \{0,1\}^{2B}$, the GGM construction produces a family of pseudorandom functions $\{F_1, \ldots, F_\alpha\}$ from the domain $\{0,1\}^n$ to the range $\{0,1\}^B$.

   In this problem, we shall generalize the GGM PRF construction in two ways.

   (a) Given a length-doubling PRG $G\colon \{0,1\}^B \to \{0,1\}^{2B}$, construct a family of pseudorandom function from the domain $\{0,1\}^n$ to the range $\{0,1\}^{100B}$.

   (b) Why is the GGM construction not a pseudorandom function family from the domain $\{0,1\}^*$ to the range $\{0,1\}^B$?

   (c) Consider the following function family $\{H_1, \ldots, H_\alpha\}$ from the domain $\{0,1\}^*$ to the range $\{0,1\}^B$. We define $H_k(x) = F_k(x, [\![x]\!]_2)$, for $k \in \{1, 2, \ldots, \alpha\}$. Show that $\{H_1, \ldots, H_\alpha\}$ is not a secure PRF from $\{0,1\}^*$ to the range $\{0,1\}^B$.

   (*Recall*: The expression $[\![x]\!]_2$ represents the length of $x$ in $n$-bit binary expression.)

   **Solution.**

   (a) Sol 1

   (b) Sol 2

4. **Variant of ElGamal Encryption.** Let $(G, \circ)$ is a group where the DDH assumption holds and $g$ is a generator for this group.

   Recall that in the ElGamal Encryption scheme encrypts a message $m \in G$ as follows.

   $\mathsf{Enc}_{\mathsf{pk}}(m)$:

   (a) Sample $a \xleftarrow{\$} \{0, 1, \ldots, |G| - 1\}$

   (b) Compute $A = g^a$

   (c) Output the cipher-text $(A, m \circ \mathsf{pk}^a)$.

   Consider the following alternate encryption scheme for $m \in \{0, 1, \ldots, |G| - 1\}$.

   $\mathsf{Enc}_{\mathsf{pk}}(m)$:

   (a) Sample $a \xleftarrow{\$} \{0, 1, \ldots, |G| - 1\}$

   (b) Compute $A = g^a$

   (c) Output the cipher-text $(A, g^m \circ \mathsf{pk}^a)$.

   Why can't this encryption scheme be used?

   **Solution.**

   (a) Because decrypting the cipher-text (multiplying by the inverse of the secret key) would give $g^m$, and not m. Since $m$ cannot be easily determined from $g^m$, this would make decrypting the cipher-text computationally expensive.

5. **Understanding Asymptotics.** Suppose we have a cryptographic protocol $P_n$ that is implemented using $\alpha n^2$ CPU instructions, where $\alpha$ is some constant. The protocol is expected to be broken using $\beta 2^{n/10}$ CPU instructions.

   Suppose, today, everyone in the world uses the primitive $P_n$ using $n = n_0$, a constant value such that even if the entire computing resources of the world were put together for 8 years we cannot compute $\beta 2^{n_0/10}$ CPU instructions.

   Assume Moore's law that the every two years, the amount of CPU instructions we can run per second doubles.

   (a) Assuming Moore's law, how much faster will be the CPUs 8 years into the future as compared to the CPUs now?

   (b) At the end of 8 years, what choice of $n_1$ will ensure that setting $n = n_1$ will ensure that the protocol $P_n$ for $n = n_1$ cannot be broken for another 8 years?

   (c) What will be the run-time of the protocol $P_n$ using $n = n_1$ on the new computers as compared to the run-time of the protocol $P_n$ using $n = n_0$ on today's computers?

   (d) What will be the run-time of the protocol $P_n$ using $n = n_1$ on today's computers as compared to the run-time of the protocol $P_n$ using $n = n_0$ on today's computers?

   (*Remark*: This problem explains why we demand that our cryptographic algorithms run in polynomial time and it is exponentially difficult for the adversaries to break the cryptographic protocols.)

   **Solution.**

   (a) By Moore's law, in 8 years CPU speed will increase by $2^(8/2) = 16$

   (b) Since CPU speed increased by 16x, we want the following property to hold true: $\frac{\beta 2^{n_1/10}}{\beta 2^{n_0/10}} = 16$
   This means $n_1 = 40 + n_0$

   (c) Because the protocol using $n_1$ will be using 16 times the number of CPU instructions, at a speed 16 times faster than today's computers, the run-time will be the same. In Big-O, the run-time will be equal.

   (d) Because the protocol using $n_1$ will be using 16 times the number of CPU instructions, the run-time would be 16 times longer on today's computers. In Big-O, the run-time will be equal.

6. **CRHF from Discrete Log Assumption.** We shall work over the group $(\mathbb{Z}_p^*, \times)$, where $p$ is a prime number. Let $g$ be a generator of this group.

   Let us define the hash function $h_y(b, x) = y^b g^x$, where $y \in \mathbb{Z}_p^*$, $b \in \{0, 1\}$, and $x \in \{0, 1, \ldots, p-1\}$. Note that the domain is of size $2(p-1)$ and the range is of size $(p-1)$. So, this hash function family compresses its input.

   Consider the hash function family $\mathcal{H} = \{h_1, h_2, \ldots, h_{p-1}\}$.

   Suppose, we sample $y \xleftarrow{\$} \mathbb{Z}_p^*$. Once $h_y$ was announced to the world, a hacker releases two distinct inputs $(b, x)$ and $(b', x')$ such that $h_y(b, x) = h_y(b', x')$.

   (a) Prove that $b = b'$ is not possible.

   (b) If $b \neq b'$, then calculate $t \in \{0, 1, \ldots, p-1\}$ such that $g^t = y$.

   (*Remark*: This is a secure CRHF construction based on the Discrete-Log Hardness Assumption. Discrete-Log Hardness Assumption states that given $y \xleftarrow{\$} \mathbb{Z}_p^*$ it is computationally hard to find $t$ such that $g^t = y$. Based on this computational hardness assumption the CRHF construction presented above is secure.

   Why? Suppose some hacker can indeed break the CRHF, i.e., find two distinct pre-images that collide. Then following your algorithm, we can find $t$ such that $g^t = y$, which was assumed to be a computationally hard task! Hence, contradiction.)

   **Solution.**

   (a) Because the inputs are not equal, we know $b \neq b'$ and/or $x \neq x'$. If $x \neq x'$, then we know $g^x$ will be a unique value in $h_y(b, x) = y^b g^x$. Thus, if $x \neq x'$, then $y^b \neq y^{b'}$ and $b \neq b'$ since $h_y(b, x) = h_y(b', x')$. Thus, no matter if $x$ is equal to $x'$, $b \neq b'$.

   (b) Given $h_y(b, x) = h_y(b', x')$
   If $(b = 0, b' = 1)$: $y = g^x \mathsf{inv}(g^{x'}) = g^(x - x') = g^((b - b')(x' - x))$
   If $(b = 1, b' = 0)$: $y = g^{x'} \mathsf{inv}(g^x) = g^(x' - x) = g^-(x - x')) = g^((b - b')(x' - x))$
   Thus $t = (b - b')(x' - x)$