Homework 2

Applied Cryptography
CS 598 : Fall 2013

Released: Sat April 6, Due: Thu April 25

Exercises on Hashing and Secure Function Evaluation [Total 100 pts]

1. [25 pts] **Preimage collision resistance and second-preimage collision resistance are incomparable.** In this problem we consider hash functions on a finite domain (from $\{0,1\}^{n(k)}$ to $\{0,1\}^k$).

   (a) Suppose $\mathcal{H}$ is preimage collision resistant. Modify $\mathcal{H}$ to $\mathcal{H}'$ (possibly with a different domain), so that the latter remains preimage collision resistant, but is not second-preimage collision resistant. (Prove these properties of $\mathcal{H}'$.)

   (b) Given a CRHF $\mathcal{H}$ which compresses by two bits (say from n bits to $n-2$ bits), construct a CRHF $\mathcal{H}'$ that compresses by one bit (say from $n+1$ bits to n bits), such that the function $f(h', x) = (h', h'(x))$ (where $h' \in \mathcal{H}'$) is **not** a OWF. (In both $\mathcal{H}$ and $\mathcal{H}'$, collision-resistance holds when the hash function is drawn uniformly at random from the family.)

   (c) **[Extra] (Sufficiently Shrinking) CRHF implies OWF.** Below we say that "x has a collision under f" if there exists an $x' \neq x$ such that $f(x) = f(x')$.

      i. Let $\mathcal{H}$ be a CRHF and suppose that for every $h \in \mathcal{H}$ and every x, x has a collision under h. Show that the function $f(h, x) = (h, h(x))$ is a OWF.

      ii. Now suppose that for each $h \in \mathcal{H}$, all but a negligible fraction of x's have a collision under h. Show that the function $f(h, x) = (h, h(x))$ is a OWF.

      iii. Show that if $\mathcal{H}$ is a CRHF from n bits to $n/2$ bits, then the function $f(h, x) = (h, h(x))$ is a OWF.

2. [25 pts] **Power of 2-party SFE with only one output.** In this problem we shall see how deterministic secure function evaluation (SFE) functionalities in which only one party receives the outcome can be easily used to realize more general functionalities securely, against passive (honest-but-curious) adversaries.

   (a) Suppose $\mathcal{R}$ is an arbitrary *randomized* 2-party functionality which takes x and y from Alice and Bob respectively, and samples a uniform random string r (of a fixed length) and gives $R_A(x, y, r)$ and $R_B(x, y, r)$ respectively to Alice and Bob. Describe a *deterministic* 2-party SFE functionality $\mathcal{F}$ (which takes x and y from Alice and Bob respectively, and gives $f_A(x, y)$ and $f_B(x, y)$ to them respectively; f_A, f_B can depend on R_A, R_B), and a protocol $\pi^{\mathcal{F}}$ (i.e., a protocol in which Alice and Bob can access a trusted party implementing $\mathcal{F}$), such that $\pi^{\mathcal{F}}$ securely realizes $\mathcal{R}$. In your protocol $\pi^{\mathcal{F}}$, Alice and Bob should access $\mathcal{F}$ exactly once. Security must hold against both passive and active adversaries.

(b) Suppose $\mathcal{F}$ is an arbitrary 2-party SFE functionality which takes x and y from Alice and Bob respectively, and gives $f_A(x, y)$ and $f_B(x, y)$ to them respectively. Describe another 2-party SFE functionality $\mathcal{G}$ which provides output only to Bob (i.e., Alice gets a dummy output $g_A(x, y) = \perp$), and a protocol $\rho^{\mathcal{G}}$ (i.e., a protocol in which Alice and Bob can access a trusted party implementing $\mathcal{G}$), such that $\rho^{\mathcal{G}}$ securely realizes $\mathcal{F}$. In your protocol $\rho^{\mathcal{G}}$, Alice and Bob should access $\mathcal{G}$ exactly once. Security needs to hold only against passive adversaries.

3. [25 pts] **OT from Correlated Random Variables.** Define Oblivious Transfer (OT) functionality over a field $\mathbb{F}$ (or, over a ring) as an SFE in which Alice inputs $(x_0, x_1) \in \mathbb{F}^2$ and Bob inputs $b \in \{0, 1\}$; then Alice gets $\perp$ as output, but Bob gets x_b.

   (a) Consider an inputless, randomized functionality RandOT, which outputs a random pair $(z_0, z_1) \in \mathbb{F}^2$ to Alice and (c, z_c) to Bob, where $c \in \{0, 1\}$ is a random bit. Give a protocol π^{RandOT} that securely realizes OT, by accessing RandOT exactly once at the beginning of the protocol.

   (b) Consider another inputless, randomized SFE functionality RandShare, which outputs $(s_A, p_A) \in \mathbb{F}^2$ to Alice and $(s_B, p_B) \in \mathbb{F}^2$ to Bob, where (s_A, s_B, p_A, p_B) are uniformly random conditioned on the relation $s_A + s_B = p_A p_B$. Give a protocol $\rho^{\text{RandShare}}$ that securely realizes OT, by accessing RandShare exactly once at the beginning of the protocol.

4. [25 pts] In secure multi-party computation protocols designed for honest-majority, a commonly used tool is a secret-sharing scheme like Shamir's secret-sharing. Let $t \leq (n-1)/2$. Consider an $(n, t+1)$ (i.e., $t+1$ out of n) Shamir secret-sharing scheme over some field. Recall that the shares of a value are obtained by evaluating a random degree t polynomial at n points in the field, such that at (say) 0, the polynomial evaluates to the value being shared. Suppose n parties hold the shares of two values x and y under such a scheme. Let the shares be x_i and y_i for $i = 1, \ldots, n$.

   (a) **Addition.** Show how the parties can obtain shares z_i for $z = x + y$ (shared using the same secret-sharing scheme), without learning anything more.

   (b) **Multiplication (changing the threshold).** Show how the parties can obtain shares W_i for $w = xy$, but shared using an $(n, 2t+1)$ Shamir secret-sharing scheme, without learning anything more.

   [Hint: Given two polynomials f and g, what can you say about the polynomial h defined as $h(i) = f(i)g(i)$?]

   (c) **Degree reduction.** Suppose the parties are given shares r_i and R_i of a value r using the $(n, t+1)$ and $(n, 2t+1)$ secret-sharing schemes above. Show how they can convert their shares W_i of a value w under the latter scheme to shares w_i under the former scheme, such that any subset of t players learn nothing more about w, where r is uniformly randomly chosen. You can assume that all the parties follow the protocol honestly.

   [Hint: Use r to blind w before reconstructing it, and then re-share it using the lower degree scheme.]

5. **[Extra]** The Needham-Schroeder Public Key protocol was an early protocol (proposed in 1978) for "authenticated key exchange," using a public-key "encryption" scheme. (This was well before Goldwasser and Micali had developed the CPA security notion for encryption.)

The protocol uses a trusted server, S, to help two parties exchange secret keys with each other. A priori, there are no secrecy or authentication guarantees on the communication network, and the parties know only each other's identities and a public key of the server S. The server, S, knows public keys of all the users. The goal of the protocol is that at the end A and B should agree on random nonces N_A and N_B (chosen by A and B respectively). The protocol is described below.

Needham-Schroeder (Public Key) Protocol: The protocol is shown in Figure 1. It is described in terms of a public key "encryption" algorithm Enc. It is a *deterministic* encryption scheme with the property that $\mathsf{Enc}_{PK}(\mathsf{Enc}^{-1}_{SK}(M)) = M$. If M is sufficiently random, $\mathsf{Enc}^{-1}_{SK}(M)$ is assumed to behave like a signature on M (though it does not give existential unforgeability). PA, PB are Alice and Bob's public keys and SA, SB are their secret keys, respectively. Likewise, the server's public and secret keys are PS, SS.

$A \to S$:	A, B	(This is A requesting S to send B's public-key)
$S \to A$:	$\mathsf{Enc}^{-1}_{SS}(K_{PB}, B)$	(A will use Enc_{PS} to recover B's public key)
$A \to B$:	$\mathsf{Enc}_{PB}(N_A, A)$	(where N_A is a fresh nonce, picked by A)
$B \to S$:	B, A	(Now B requests S to send A's public-key)
$S \to B$:	$\mathsf{Enc}^{-1}_{SS}(K_{PA}, A)$	(B will use Enc_{PS} to recover A's public key)
$B \to A$:	$\mathsf{Enc}_{PA}(N_B, N_A)$	(where N_B is a fresh nonce picked by B)
$A \to B$:	$\mathsf{Enc}_{PB}(N_B)$	(A and B agree on N_A, N_B at this point)

Figure 1: The Needham-Schroeder public-key protocol.

(a) There is a (famous) man-in-the-middle attack on this protocol, whereby a party in the system can set up a shared key with B, while she thinks she has shared that key with A. Describe such an attack (without looking it up!).

[Hint: The adversary can run a concurrent session with A.]

(b) Suggest a (small) fix for the attack.

(c) If you were designing this protocol today, using public-key encryption and signatures, how would you do it?