Homework 3

Cryptography & Network Security
CS 406 : Fall 2018

Released: Tue October 30
Due: Mon November 12

Signatures, Random Oracles [Total 75 pts]

1. **Attacking a Signature Scheme** [20 pts]

   In this problem, we consider a seemingly minor modification of the Schnorr signature scheme, and show that it can be broken.

   Recall that, in the original scheme, the verification key is $(\mathbb{G}, g, Y)$, where $\mathbb{G}$ is a prime-order group with a generator g and $Y = g^y$ is a random group element, with $y \leftarrow \mathbb{Z}_{|\mathbb{G}|}$ being the signing key; the signature on a message M is produced as $\mathsf{Sign}_y(M) = (e, s)$, where $e = H(M||g^r)$ and $s = r - ye$, for a random $r \leftarrow \mathbb{Z}_{|\mathbb{G}|}$.

   In the modfied scheme the messages belong to $\mathbb{G}$, and $e = H(M||g^r)$ is replaced by $e = H(M \cdot g^r)$.

   Give an existential forgery attack on this modified scheme (in the random oracle model).

2. **CCA Secure PKE in the Random Oracle Model** [45 pts]

   Suppose $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure PKE scheme. We shall write $\mathsf{Enc}_{PK}(m; r)$ to indicate encryption of the message m using randomness r; suppose Enc requires $r \leftarrow \{0,1\}^k$ (k, as always, being the security parameter). Also, suppose H is a hash function modeled as a random oracle with k-bit outputs.

   Consider a new encryption scheme with the encryption algorithm defined as follows: $\mathsf{Enc}^*_{PK}(m; r) = (\mathsf{Enc}_{PK}(m||r; H(r)), H(m||r))$, where $r \in \{0,1\}^k$.

   (a) What should the corresponding decryption algorithm Dec^* be so that $(\mathsf{KeyGen}, \mathsf{Enc}^*, \mathsf{Dec}^*)$ is a CCA-secure encryption scheme?

   (b) Prove that with Dec^* as you defined above, $(\mathsf{KeyGen}, \mathsf{Enc}^*, \mathsf{Dec}^*)$ is indeed a CCA-secure encryption scheme in the random oracle model. Flesh out the details of the proof as much as you can, basing your arguments only on the CPA-security of the given scheme, and statistical properties.

   Hint: You should convert a CCA-adversary A^* for $(\mathsf{KeyGen}, \mathsf{Enc}^*, \mathsf{Dec}^*)$ into a CPA-adversary A for $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. A will need to simulate the random oracle and the decryption oracle that A^* expects. As such, A gets to see all random oracle queries that A^* makes.

   (c) Show that the scheme will not even be CPA secure if $H(m||r)$ is replaced by $H(m)$.

   (d) Show that, for some choice of a CPA-secure scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, the modified scheme will not even be CPA secure if $H(m||r)$ is replaced by $H(r)$. [Extra Credit]

3. **Needham-Schroeder Protocol.** [10 pts]

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 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 (very weak) signature on M: it is infeasible for an adversary who is *given a random* M to create the signature on M (note that this is weaker than the notion of existential unforgeability, which is not satisfied by this scheme). 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)
$S \to A$:	$\mathsf{Enc}^{-1}_{SS}(PB, B)$	(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)
$S \to B$:	$\mathsf{Enc}^{-1}_{SS}(PA, A)$	(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 E in the system can set up a shared key with B, such that B thinks that 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? [Extra Credit]