CSE 5852, Modern Cryptography: Foundations Fall 2016

Lecture 20, 2016

Prof. Benjamin Fuller *Scribe: Kirk Gardner*

1 Overview

Last class we introduced *public key signatures*. Our goal is to allow the reciever of a message to be assured it came from the intended sender. That is, the receiver should know the message was not changed as well as who created it. The general procedure is as follows

$(vk, sk) \leftarrow \text{Gen}(1^n)$	Generate a *verification key* vk and a *signing key* sk.
$\sigma \leftarrow \text{Sign}(m, sk)$	Sign a message m with sk.
$0, 1 \leftarrow \text{Vfy}(vk, m, \sigma)$	Use vk to check if σ is the correct signature of m.

We also showed the set $\mathbb{Z}_N^* = \{a \in \mathbb{Z} \mid a < N \text{ and } \gcd(a, N) = 1\}$ is a group under multiplication for an integer N, and introduced the *totient function* $\phi = |\mathbb{Z}_N^*|$.

Theorem 1. *Fix $N \in \mathbb{Z}$. Fix $e \in \mathbb{Z}$ such that $gcd(e, \phi(N)) = 1$. The function $f_e : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined as $f_e(x) = x^e \mod N$ is a permutation. For $d \in \mathbb{Z}$ such that $d = e^{-1}$ the function f_d is a permutation, and the inverse of f_e. That is*

$$f_d(f_e)) = x.$$

Lemma 2. *For $d \in \mathbb{Z}$ such that $d = e^{-1}$*

$$f_d(f_e)) = x.$$

Proof. We need to show $x^{ed} \equiv x \mod N$. We know that for some α

$$ed - 1 = \alpha\phi(N) \implies ed = \alpha\phi(N) + 1$$

so

$$x^{\alpha\phi(N)+1} = (x^{\phi(N)})^{\alpha} \cdot x$$

Last class we proved a generalization of Fermat's little theorem which states $x^{\phi(N)} = 1$, thus

$$x^{\alpha\phi(N)+1} \equiv x \mod N.$$

□

This class we will begin to provide assumptions in order to assure only the owner of the signing key can compute f_d while allowing anyone to compute f_e. In order to assure an attacker cannot compute f_e from f_d we need to satisfy the following minimal requirements

1. Can't figure out $\phi(N)$,
2. Can't factor (otherwise we can compute $\phi(N)$),
3. Can't figure out d.

2 Factoring is hard

The factoring problem requires choosing two prime numbers such that, when given to an attacker, their product cannot split them up.

1. Choose prime p, q using a "special algorithm,"
2. compute $N = p \cdot q$,
3. $p'q' \leftarrow \mathcal{A}(N)$,
4. output 1 if $p'q' = N$ and $p`, q' \neq 1$.

Before going in to detail about the so-called "special algorithm" used for selection the algorithm parameters we will introduce the RSA assumption.

$\texttt{RSA} - \texttt{inv}(1^n)$ (Rivest, Shamir, Adelman 1977)

1. Choose n-bit p, q using "special algorithm,"
2. compute N, e, d,
3. choose uniform $y \in \mathbb{Z}_N^*$,
4. $x = \mathcal{A}(N, e, y)$,
5. $\mathcal{A}$ wins if $x^e = y \mod N$.

Assumption 3. *If N, e, d are chosen properly for every PPT $\mathcal{A}$ there exists a negligible function $\epsilon(n)$ such that*

$$Pr[\texttt{RSA} - \texttt{inv}_{\mathcal{A}}(n) = 1] \leq \epsilon(n).$$

The "special algorithm" we have been referring to selects paramaters p, q, d, e by picking odd numbers at random and performing *probabilistic primality tests* in order to obtain two prime numbers p, q satisfying the following minimal assumptions.

1. p, q sould not be too close: $|p - q| > 2n^{1/4}$ (otherwise, Fermat's factorization gives p, q),
2. $p - q$ and $q - 1$ shouldn't have only small factors (otherwise, Pollard's factorization gives p, q),
3. d has to be big (otherwise Weiner (2009) to compute d if $d = n^{1/4}$),
4. e should be big enough so $x^e \neq x^e \mod N$.

3 Basic RSA

The main idea of the RSA problem is to use the permutations of f_e and f_d in such a way that makes it easy for anyone to apply the permutation f_e while preventing anyone from inverting without d.

The **basic RSA** signature scheme is as follows

1. $\text{Gen}(1^n)$:
   - Choose two n-bit prime integers p, q using the special algorithm,
   - compute $N = p \cdot q$ and compute e, d,
   - output $N, e = vk, d = sk$.
2. $\sigma = \text{Sign}(d, m) = f_d(m)$,
3. $\text{Vfy}(N, e, m, \sigma) = f_e(\sigma) = f_e(m^d) = m^{de} = m'$. Outout 1 iff $m' = m$.

Through the RSA assumption we can argue that it is difficult for the adversary to sign a random message. In order to talk about the security of the scheme we introduce the EU-CMA experiment

$\texttt{EU}-\texttt{CMA}_{\text{Gen,Sign,Vfy},\mathcal{A}}(1^n)$:

1. $sk, vk \leftarrow \text{Gen}(1^n)$,
2. give vk to $\mathcal{A}$,
3. for $i = 1, \ldots, k$
   - get m_i from $\mathcal{A}$
   - give $\sigma_i = \text{Sign}(sk, m_i)$ to $\mathcal{A}$
4. $\mathcal{A}$ outputs m', σ'.
5. Output 1 if $\text{Vfy}(vk, m, \sigma) = 1$ and $m' \neq m_i$ for all $i = 1, \ldots, k$.

Definition 4 (EU-CMA Security)**.** *A signature scheme is* ***existentially-unforgeable under chosen message attack (EU-CMA)*** *if for all PPT $\mathcal{A}$ there exists a negligible ϵ such that*

$$Pr[\texttt{EU}-\texttt{CMA}_{\mathcal{A}}(n) = 1] < \epsilon(n).$$

We can very easily break this scheme by selecting a signature σ at random and computing $\sigma^e = m$. Now, this message m with be uniformly distributed in the message space, but m, σ will still represent a valid message pair.

4 Hash Functions

Given some function H suppose we redefine our signature scheme as follows:

$\text{Sign}(d, N, m)$:

1. Compute $y = H(m)$,
2. Compute $\sigma = H(m)^d \mod N$.

$\text{Vfy}(e, N, m, \sigma)$:

1. Compute $y' = \sigma^e \mod N$,

2. output 1 iff $m' = H(m)$.

This function H will need to have some properties of a *hash function*. That is, we are looking for a function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ with the following properties.

1. **One-Way:** hard to invert.

2. **Collision-Resistant:** if m, m' are such that $H(m) = H(m')$ then $\text{Sign}(sk, m) = \text{Sign}(sk, m')$.

3. If $\text{Sign}(sk, m) = H(m)^d$ and $\text{Sign}(sk, m') = H(m')^d$ it should be hard to find a message m^* such that
$$H(m^*) = H(m)H(m').$$

In the next class we will investigate what additional properties we need from a hash function in order to prove security.