CSE 599b: Cryptography **(Winter 2006)**

Lecture 12: Public Key (Asymmetric) Encryption

10 February 2006

Lecturer: Paul Beame *Scribe: Paul Beame*

1 RSA

One way to use Diffie-Hellman's 1976 secret key exchange protocol is to create a key to be used in later rounds for symmetric encryption. This requires multiple rounds of communication. Rivest, Shamir, and Adleman in 1978 developed the one round communication scheme discussed earlier.

Namely, in order for Bob to receive messages Bob produces an integer $N = pq$ where p and q are primes of equal length and Bob also produces e relatively prime to $\varphi(N) = (p-1)(q-1)$ and computes $d = e^{-1} \mod (p-1)(q-1)$. Bob publishes the public key (N, e) and retains d as his private key.

Given message M, Alice can then compute $C = RSA_{(N,e)}(M) = M^e \mod N$ and send C to Bob. Bob decrypts by computing $C^d \mod N$ since

$$C^d \mod N = (M^e \mod N)^d \mod N = M^{de} \mod N = M^1 \mod N = M$$

where we used Euler's theorem that for M relatively prime to N (true for almost all M) we have $M^{\varphi(N)} \mod N = 1$. We call this scheme Plain RSA and we will see that it is not very secure.

The key observations about this scheme are that $RSA_{(N,e)}$ is a permutation and that knowledge of d acts like a 'trapdoor' that suddenly opens a way to invert $RSA_{(N,e)}$. We will give a more general definition of objects like this.

2 Collections of Trapdoor Permutations

Definition 2.1. *A* collection of trapdoor permutations *is a collection of functions* $\{f_i : D_i \to D_i\}_{i \in I}$ *such that*

- *Each f_i is a permutation of D_i.*
- *Sampling:*
  - *There is a PPT C_I^+ that on input 1^k produces a pair (i, t_i) where i is an element of $I \cap \{0,1\}^k$ and t_i is a string called the* trapdoor *for f_i.*
  - *There is a PPT S_D that on input $i \in I$ produces a (nearly) uniformly randomly chosen element x of D_i.*

- *Easy to Compute: There is a deterministic polynomial-time algorithm F that on input $i \in I$ and $x \in D_i$ computes $f_i(x)$.*
- *Hard to Invert: For all PPT A the function ϵ is negligible for*
$$\epsilon(k) = \Pr[A(f_i(x), i) \in f_i^{-1}(f_i(x)) \mid i \leftarrow C_I(1^k);\ x \leftarrow S_D(i)],$$
where $C_I(1^k)$ is the first component of $C_i^+(1^k)$.
- *Easy to Invert given Trapdoor: There is a deterministic polynomial-time algorithm F^{-1} that on input t_i for $i \in I$ and $y \in D_i$ computes $f_i^{-1}(y)$.*

Since, in particular, collections of trapdoor permutations are special cases of collections one-way functions, a collection of trapdoor permutations is guaranteed to have associated hard-core functions $B_i : D_i \to \{0,1\}^{b_i}$. (Hard-core bits if $b_i = 1$.)

We will see that these will be sufficient for IND-CPA secure public-key cryptosystems.

3 Public Key Encryption

Definition 3.1. *A* public key cryptosystem *is given by three algorithms:*

(i) Key generation: There is a PPT algorithm $\mathcal{G}$ that on input 1^k produces a pair of keys (e, d) where e is an encryption key and d is a decryption key.

(ii) Encryption: There is a PPT algorithm $\mathcal{E}$ that on input key e together with a message M produces $\mathcal{E}_e(M) = \mathcal{E}(e, M)$.

(iii) Decryption: There is a deterministic polynomial time algorithm D that on input d and $C = \mathcal{E}_e(M)$, satisfies $D_d(C) = D(d, C) = M$.

We now define IND-CPA security for public key encryption. Since the public key is indeed public, instead of given the adversary an encryption oracle it suffices to give the adversary the public key e. However, we must explicitly represent the choice of the two messages M^0 and M^1 in IND-CPA-style security.

Definition 3.2. *A public key encryption scheme $(\mathcal{G}, \mathcal{E}, D)$ is IND-CPA secure if and only if for every PPT A and every PPT $\mathcal{M}$, the function ϵ is negligible for*

$$\begin{aligned}\epsilon(k) &= \Pr[A(e, \mathcal{E}_e(M^0)) = 1 \mid (e, d) \leftarrow \mathcal{G}(\infty^{\|});\ (M^0, M^1) \leftarrow \mathcal{M}(e)] \\ &\quad - \Pr[A(e, \mathcal{E}_e(M^1)) = 1 \mid (e, d) \leftarrow \mathcal{G}(\infty^{\|});\ (M^0, M^1) \leftarrow \mathcal{M}(e)]\end{aligned}$$

Observe that as with other deterministic schemes, the original version of RSA encryption described above is not IND-CPA secure. Goldwasser and Micali, who first defined IND-CPA security for public-key encryption also defined a probabilistic encryption scheme that achieved it. The following method is similar in spirit to their construction although it is slightly different in details.

Probabilistic Encryption Let $\{f_i : D_i \to D_i\}_{i \in I}$ be a collection of trapdoor permutations and suppose that $\{B_i : D_i \to \{0,1\}^{b_i}\}$ is a collection of hard-core functions for this trapdoor collection (i.e., for random $x \in D_i$, given $f_i(x)$ it is hard to distinguish $B_i(x)$ from $\mathcal{U}_{b_i}$). The encryption scheme is defined as follows:

$\mathcal{G}(1^k)$ Run $C_I^+(1^k)$ to generate (e, d) and set $e = i$ and $d = t_i$.

$\mathcal{E}_e(M)$ Break M into blocks of $b = b_i$ bits, $M = \beta_1\beta_2 \ldots \beta_\ell$. For $j = 1$ to ℓ, set $r_j \leftarrow S_D(i)$ and set $C_j = (f_i(r_j), B_i(r_j) \oplus \beta_j)$, return $C = C_1 \ldots C_\ell$.

$D_d(C_1 \ldots C_\ell)$ For $j = 1$ to ell and $C_j = (y, z)$ compute $F^{-1}(t_i, y) = f_i^{-1}(y) = r_j$. Then return $\beta_j = z \oplus B_i(r_j)$.

For example, using the fact that for an n-bit $N = pq$ of n bits the $\log_2 n$ least significant bits of $x \in \mathbb{Z}_N^*$ is a hard-core function for $RSA_{(N,e)}$. Thus the probabilistic encryption scheme for $RSA_{(N,e)}$ encrypts each block β of $\log_2 n$ bits using $(r^e \mod N, \beta \oplus LSB_{\log_2 n}(r))$.

Theorem 3.3. *If $\{f_i : D_i \to D_i\}$ is a collection of trapdoor permutations with associated collection of hard-core functions $\{B_i : D_\to \{0,1\}^{b_i}\}$ then probabilistic encryption scheme above is IND-CPA secure.*

Proof. By our typical hybrid argument it suffices to show that IND-CPA security for two messages $M^0 = \beta^0$ and $M^1 = \beta^1$ that consist of a single blocks of b_i bits. Thus we are interested in the difference between the probability that A outputs 1 on input $(f_i(r), B_i(r) \oplus \beta^0)$ for random $r \in D_i$ versus $(f_i(r), B_i(r) \oplus \beta^1)$ for random $r \in D_i$. Since B_i is a hard-core function for f_i and f_i is a permutation the string $(f_i(r), B_i(r))$ is computationally indistinguishable from a distribution consisting of a random element of D_i and an independent random string of b_i bits. Therefore, $(f_i(r), B_i(r) \oplus \beta^1)$ and $(f_i(r), B_i(r) \oplus \beta^1)$ are also computationally indistinguishable from this same distribution. (Essentially we have used the fact that $f_i(r)B_i(r)$ is pseudorandom.) $\square$

For the probabilistic scheme above using RSA transmits $n + \log_2 n$ bits for every $\log_2 n$ bits of the message M since only the $O(\log n)$ least significant bits of RSA have been proved to be hard-core. This can be made substantially more efficient if the following conjecture is true.

Conjecture For $n = |N|$, $LSB_{n/2}$ is a hard-core for function for $RSA_{(N,e)}$.

A more efficient (PRNG-based) public key scheme The idea here is to use the observation that since f_i is a permutation, $f_i(r)$ for random r is also a random element of D_i and thus, just as in the case of pseudorandom generators, $f_i(r)$ can be used for the next blocks of bits:

$\mathcal{E}_e(M)$ If $|M| = b_i\ell$, i.e. M consists of ℓ blocks, then $C = \mathcal{E}_e(M) = (f_i^\ell(r), G_i^\ell(r) \oplus M)$ for random $r \in D_i$ where $G_i^\ell(r) = B_i(r)B_i(f_i(r)) \ldots B(f_i^{\ell-1}(r))$ is the standard PRNG applied to seed r.

D_d To decrypt $C = (y, z)$, write $z = z_1 \ldots z_\ell$ where each z_j has b_i bits. Then for $j = \ell$ down to 1 do: Compute $y' = F^{-1}(t_i, y) = f_i^{-1}(y)$, set $M_j = z_j \oplus B_i(y')$, and replace y by y'.

This scheme is significantly more efficient than the above scheme since the overhead is only one element of D_i per message.

Theorem 3.4. *Given a collection of trapdoor permutations and related collection of hard-core functions for the trapdoor permutations, the PRNG-based scheme above is IND-CPA secure.*

Proof Sketch. The argument is similar to the above proof except this time we use the fact that

$$G^{\ell+}(x) = B(x)B(f(x))\dots B(f^{\ell-1}(x))f^{\ell}(x) = G^{\ell}(x)f^{\ell}(x)$$

is PRNG. □

4 A Hybrid Encryption Scheme

Since public-key (asymmetric) encryption has more stringent requirements than symmetric encryption, symmetric encryption schemes in general will be more efficient than public-key encryption. Thus a natural scheme for public-key encryption will be to use the public-key aspects as little as possible, namely use a secure public-key scheme to send a randomly chosen key to use in a symmetric encryption scheme. This is the most common use of public-key encryption.

Suppose that $(\mathcal{G}, \mathcal{E}, D)$ is an IND-CPA secure public-key encryption scheme and $(\mathcal{K}, \mathcal{E}', D')$ is an IND-CPA secure symmetric encryption scheme then, provided that for $(e, d) \leftarrow \mathcal{G}(1^k)$ every element of $\mathcal{K}(1^k)$ is in the domain of $\mathcal{E}_e$ we obtain another IND-CPA secure public-key scheme H that is a hybrid of the two schemes as follows:

- The key generation algorithm $\mathcal{G}''$ for the scheme is $\mathcal{G}$ from the public-key scheme.
- The encryption algorithm $\mathcal{E}''$ encrypts M as follows: run $K \leftarrow \mathcal{K}(1^k)$ and send $C = (\mathcal{E}_e(K), \mathcal{E}'_K(M))$.
- To decrypt $C = (C_K, C_M)$ compute $K \leftarrow D_d(C_K)$ and $M \leftarrow D'_K(C_M)$.

Theorem 4.1. *Given an IND-CPA secure public-key encryption scheme and an IND-CPA secure symmetric encryption the above hybrid scheme is IND-CPA secure.*

Proof. Let $\mathcal{M}$ and A be PPT algorithms and consider

$$\begin{aligned} \epsilon(k) &= \Pr[A(e, \mathcal{E}''_e(M^0)) = 1 \mid (e, d) \leftarrow \mathcal{G}''(1^k);\ (M^0, M^1) \leftarrow \mathcal{M}(e)]. \\ &\quad - \Pr[A(e, \mathcal{E}''_e(M^1)) = 1 \mid (e, d) \leftarrow \mathcal{G}''(1^k);\ (M^0, M^1) \leftarrow \mathcal{M}(e)]. \end{aligned}$$

Choose $(e, d) \leftarrow \mathcal{G}''(1^k) = \mathcal{G}(1^k)$ and choose (M^0, M^1) by $\mathcal{M}(e)$. The encryptions $\mathcal{E}''_e(M^0)$ and $\mathcal{E}''_e(M^1)$ involve independent choices of symmetric encryption keys. Let $K_0 \leftarrow \mathcal{K}(1^k)$ and $K_1 \leftarrow \mathcal{K}(1^k)$. For $a, b \in \{0, 1\}$, let $\mathcal{H}_{ab}$ be the distribution of $(\mathcal{E}_e(K_a), \mathcal{E}_{K_0}(M^b))$. The probability distributions that A is trying to distinguish are $\mathcal{H}_{00}$ and $\mathcal{H}_{01}$.

Let p_{ab} be the probability that $A(e, \mathcal{H}_{ab}) = 1$ given $(e, d) \leftarrow \mathcal{G}(1^k)$ and $(M^0, M^1) \leftarrow \mathcal{M}(e)$. Thus

$$\epsilon(k) = p_{00} - p_{01} = (p_{00} - p_{10}) + (p_{10} - p_{11}) + (p_{11} - p_{01}).$$

Observe that for $\mathcal{H}_{00}$ and $\mathcal{H}_{10}$, A is comparing $(\mathcal{E}_e(K_0), \mathcal{E}_{K_0}(M^0))$ versus $(\mathcal{E}_e(K_1), \mathcal{E}_{K_0}(M^0))$. By the IND-CPA security of the public-key system, $p_{00} - p_{10}$ is negligible. (Using $\mathcal{M}$ and $\mathcal{K}$, given e a PPT algorithm $\mathcal{M}'$ can produce the pair (K_0, K_1) and a PPT A' can create $\mathcal{E}_{K_0}(M^0)$ and send $(C, \mathcal{E}_{K_0}(M^0))$ to A. The value of $p_{00} - p_{10}$ is precisely the advantage of A'.)

Similarly, for $\mathcal{H}_{11}$ and $\mathcal{H}_{01}$, A is comparing $(\mathcal{E}_e(K_1), \mathcal{E}_{K_0}(M^1))$ versus $(\mathcal{E}_e(K_0), \mathcal{E}_{K_0}(M^1))$. Again, by the IND-CPA security of the public-key system, $p_{11} - p_{10}$ is negligible.

Finally observe that for $\mathcal{H}_{10}$ and $\mathcal{H}_{11}$, A is comparing $(\mathcal{E}_e(K_1), \mathcal{E}_{K_0}(M^0))$ versus $(\mathcal{E}_e(K_1), \mathcal{E}_{K_0}(M^1))$. By the IND-CPA security of the symmetric encryption scheme, $p_{11} - p_{10}$ is negligible. The main idea of this argument is that it allows one to decouple the public-key and symmetric encryption schemes because K_0 and K_1 are statistically independent and thus $\mathcal{E}_e(K_1)$ has no reltionship with the symmetric key encryption. $\square$

We will consider more examples of public-key encryption schemes next class.

4.1 Malleability

It is quite clear that since the above schemes work using XOR with pseudorandom strings, the above schemes are completely malleable: One can convert the encryption of M to an encryption of M' by XOR-in the bits of $M \oplus M'$ in suitable bit positionas in the ciphertext.

5 Stronger Security Definitions

We have defined IND-CPA security already for public-key encryption. The other three levels of security to which we have restricted our consideration are NM-CPA, IND-CCA2, and NM-CCA2 security. Since any public-key security must involve access to the public encryption key, it is easy to see that the CCA2 versions of security are at least as strong as the corresponding CPA security; in fact by the above example, the CCA2 definitions are strictly stronger than IND-CPA security. In fact IND-CCA2 security and NM-CCA2 security are equivalent so we will end up with only three security notions of interest.

To give an idea of how one might define non-malleability formally, we give an example of a different way of presenting security definitions, namely in terms of a PPT simulator that does not have access to the relevant ciphertexts.

Definition 5.1. *A public-key encryption system $(\mathcal{G}, \mathcal{E}, D)$ is NM-CCA2 secure if and only if for all PPT A, R, and $\mathcal{M}$ there is a PPT S (a simulator) such that*

$$\begin{aligned}\epsilon(k) \;=\; & \Pr[R(M, D_d(A^{D_d(\cdot)}(e, \mathcal{E}_e(M), s))) = 1 \mid (e, d) \leftarrow \mathcal{G}(1^k);\ (M, s) \leftarrow \mathcal{M}^{D_d(\cdot)}(e)] \\ & - \Pr[R(M, D_d(S(e, s))) = 1 \mid (e, d) \leftarrow \mathcal{G}(1^k);\ (M, s) \leftarrow \mathcal{M}^{D_d(\cdot)}(e)]\end{aligned}$$

is a negligible function of k where the oracle $D_d(\cdot)$ may not be called on input $\mathcal{E}_e(M)$.

The intuition is that R is the known relation that is supposed to be satisfied between the message M and the message(s) corresponding to the ciphertext(s) that A produces; s is the state information that is passed from the algorithm $\mathcal{M}$ that makes queries before M is chosen to the algorithm A. What the NM-CCA2 security says is that seeing the encryption $\mathcal{E}_e(M)$ and all the future calls to the decryption oracle do not help in finding a ciphertext of some M' that is related to M. For NM-CPA security one simply removes the $D_d(\cdot)$ oracle from the above definition.

In general, NM security implies IND security since the relation R can be chosen to be equality. In that case the distribution $\mathcal{M}(e)$ can choose M by choosing M^0 or M^1 with equal probability.

We will discuss more details of this definition later when we consider the Cramer-Shoup encryption scheme which is based on the El Gamal scheme which we discuss next time.