Lecture 13 - Public Key Cryptography (continued).

Boaz Barak

November 10, 2005

Different types of permutations. It's important not to get confused between *pseudorandom permutations* (PRP), *one way permutations* (OWP) and *trapdoor permutations* (TDP).

One-way permutations. One way permutations are *unkeyed* - it's not a collection of functions but just a single function (or if we think of it as many functions then there's only one for every input length). The definition is the following:

Definition 1 (One-way permutations.)**.** A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ where for every n , f_n (the restriction of f to $\{0,1\}^n$) is a permutation on $\{0,1\}^n$, is called a *one-way permutation* if there exist super-polynomial functions T, n such that for every $T(n)$-sized circuit A

$$\Pr_{x \leftarrow_R \{0,1\}^n}[A(f(x)) = x] \leq \epsilon(n)$$

One-way permutations can be used to obtain *private key* encryptions by using their hardcore bit to a obtain pseudorandom generators, and from it obtaining pseudorandom functions and CPA (or even CCA) secure encryptions.

One way permutations are "cousins" of *one-way functions* defined as follows: (for every y we denote by $f^{-1}(y)$ the set $\{x \mid f(x) = y\}$)

Definition 2 (One-way functions.)**.** A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called a *one-way function* if there exist super-polynomial functions T, n such that for every $T(n)$-sized circuit A

$$\Pr_{x \leftarrow_R \{0,1\}^n}[A(f(x)) \in f^{-1}(f(x))] \leq \epsilon(n)$$

Like one-way permutations, one way functions are also unkeyed primitives. It's easy to see that any one-way permutations is a one-way function but not vice versa (a one-way function does not have to be a permutation). It's also known (through a much harder proof by Hastad, Impagliazzo, Luby and Levin) that one-way functions imply pseudorandom generators (which imply private-key encryption).

Pseudorandom permutations. Pseudorandom permutations (PRP) are *keyed*: there are many for each input length. The security they offer is incomparable to one-way permutations: if the adversary only has black-box access to a pseudorandom permutation, then it is indistinguishable form a random permutation. This implies that it is one-way (it's hard to invert a random input) but much more than that (for example its also hard to invert a non-random input). However, if the adversary knows the key then the permutation becomes in fact easy to invert.

Definition 3 (Pseudorandom permutations)**.** A collection of permutations $\{p_k\}_{k\in\{0,1\}^*}$ where for $k \in \{0,1\}^n$, p_k is a permutation over $\{0,1\}^{m(n)}$ isa *pseudorandom-permutation collection* if it satisfies:[1]

- *(Efficient computation)* The functions $(k,x) \mapsto p_k(x)$ and $(k,y) \mapsto p_k^{-1}(y)$ are efficiently computable (i.e., in polynomial time).
- *(Pseudorandomness)* There are super polynomial functions T, ϵ such that for every $T(n)$ time adversary Adv,

$$\left| \Pr_{k\leftarrow_R\{0,1\}^n}\left[\mathsf{Adv}^{p_k(\cdot),p_k^{-1}(\cdot)}(1^m)=1\right] - \Pr_{P\leftarrow_R\{0,1\}^{m\overset{1\text{-}1}{\to}\{0,1\}^m}}\left[\mathsf{Adv}^{P(\cdot),P^{-1}(\cdot)}(1^m)=1\right]\right| < \epsilon(n)$$

  (The notation $\mathsf{Adv}^{f,g}$ means that the adversary is given oracle access to the functions $f(\cdot)$, $g(\cdot)$ which naturally it can query for at most T times. If an event happens with probability at most $1/n^{\omega(1)}$ then we say it happens with *negligible* probability.)

Pseudorandom permutations yield almost directly (by some padding) CPA (and in fact even CCA) secure *private key* encryption schemes. They are closely related to pseudorandom functions in the sense that **(1)** every pseudorandom permutation collection is also a pseudorandom function collection **(2)** we can construct a pseudorandom permutation collection from a pseudorandom function collection (we did not see this but this is in Goldreich).

Trapdoor permutations. As far as we know, both one-way and pseudorandom permutations do not help us to get *public key* encryption schemes. The way we obtain these is by using *trapdoor permutations*. These are *keyed* collections with the following property: there are two keys for each function: one to compute it in the forward direction and one to compute it in the reverse direction (invert it). Now the key for the forward direction can be given to the adversary (not inside a black box but really given to him) and still this will not help him invert the function (that is, the function is a one-way permutation to someone not knowing the invertion key or "trapdoor").

Definition 4 (One-way permutations.)**.** A *trapdoor permutation* consists of three polynomial-time algorithms G, F, B where G is probabilistic and F, B deterministic satisfying the following:

- For every n and pair $(f,b) = G(1^n)$, the function $x \mapsto F_f(x)$ is a permutation over some set S. Furthermore, the function $B_b(\cdot)$ is the inverse of that permutation. That is, for every $x \in S$, $B_b(F_f(x)) = x$.
- The set S is *efficiently sampleable*: there is a polynomial-time algorithm to output a random element of S (or a distribution statistically close to a random element of S).
- For a random $(f,b) \leftarrow_R G(1^n)$, the function $x \mapsto F_f(x)$ is a one-way permutation even if the adversary knows f. That is, there exist super-polynomial functions T, n such that for every $T(n)$-sized circuit A

$$\Pr_{(f,b)\leftarrow_R G(1^n), x\leftarrow_R S}[A(f,F_f(x))=x] \leq \epsilon(n)$$

Examples of pseudorandom permutations.

[1] We assume that m and n are polynomially related. That is, $n^{1/c} < m(n) < n^c$ for some constant c.

RSA function RSA stands for Rivest, Shamir and Adelman this is the first trapdoor permutation suggested (in 1977) and is still the most widely used.

- Keys: choose p, g random primes of length ℓ, $n = p \cdot q$. Note that $\phi(n) = |Z_n^*| = (p-1)(q-1)$ choose e at random from $Z_{\phi(n)}^*$ (that is, $gcd(e, \phi(n)) = 1$. Note that $\phi(n)$ is *even* and hence, unlike in the Rabin case, e can not be two.
- Forward (public) key: n, e
- Backward (inversion/trapdoor) key: d such that $d = e^{-1} \pmod{()\phi(n)}$. That is, $ed = k\phi(n) + 1$. Note that d can be computed from $\phi(n)$ (which can be computed using the factorization p, q of n.
- Forward evaluation: $RSA_{n,e}(x) = x^e \pmod{()n}$.
- $RSA_{n,e}(x)$ is a permutation on $\mathbb{Z}_n^*$. We show this by giving the inverse: if $x \in \mathbb{Z}_n^*$ let $y = RSA_{n,e}(x) = x^e \pmod{n}$. Then, $y^d \pmod{n} = x$. Indeed, for every group G and element $a \in G$ we have that $a^{|G|} = 1$ and so in particular $x^{\phi(n)} = 1$. Hence $x^{ed} = x^{k\phi(n)+1} = x^{k\phi(n)}x = 1 \cdot x$.

Note that we can generate a random element of $\mathbb{Z}_n^*$ by choosing a random number x in $0, 1, \ldots, n-1$ and verifying that $gcd(x, n) = 1$. The probability for that is overwhelming since there are $(p-1)(q-1) = pq - p - q + 2$ elements in $\mathbb{Z}_n^*$ and so only a tiny fraction of the pq numbers between 0 and $n-1$ are not in $\mathbb{Z}_n^*$.

The **RSA Assumption** is that the RSA function is indeed a trapdoor permutation. It is known to be a stronger assumption than the assumption that factoring random integers is hard (by random I mean product of two large random primes). However, it is not known whether or not these assumptions are equivalent. That is, as far as we know, it may be the case that there is an efficient algorithm to invert the RSA function even if there is no efficient factoring algorithm.

Rabin trapdoor permutations. The Rabin function is not exactly a permutation (it is 4 to 1). However, as was shown by Blum and Williams it can be modified to be a trapdoor permutation assuming factoring random Blum integers is hard. A Blum integer is a number $n = pq$ where $p, q = 3 \pmod{()4}$.

- Keys: choose p, g random primes of length ℓ with $p, q = 3 \pmod 4$, $n = p \cdot q$. Note that $\phi(n) = (p-1)(q-1) = (4k+2)(4k'+2) = 4k$
- Forward (public) key: n
- Backward (inversion/trapdoor) key: p, q.
- Forward evaluation: $RABIN_n(x) = x^2 \pmod{()n}$.
- $RABIN_n(x)$ is a permutation on QR_n where QR_n is the set of quadratic residues modulu n. We show this by giving the inverse: if $x \in \mathbb{Z}_n^*$ let $y = RABIN_n(x) = x^2 \pmod{n}$.

  Our inverse will be the following: we'll compute $a = y \pmod p$ and $b = y \pmod q$. Recall that $p, q = 3 \pmod 4$ and so we can say $p = 4t + 3$ and $q = 4t' + 3$. We'll compute $x_1 = a^{t+1} \pmod p$ and $x_2 = b^{t'+1} \pmod q$ and invert $\langle x_1, x_2 \rangle$ using the chinese remainder theorem to get x'. If we prove $x' = x$ then we're done.

  Because Chinese remaindering is a one-to-one operation it is enough to prove that $x_1 = x \pmod p$ and $x_2 = x \pmod q$. We'll use here the fact that x was itself a quadratic residue and hence $x = s^2 \pmod n$ for some s.

We know that $x \pmod p = s^2 \pmod p$ and hence $x_1 = (x^2)^{t+1} = s^{4(t+1)} = s^{p-1+2} = s^2 \pmod p = x \pmod p$.

Similarly $x_2 = s^2 \pmod q$ and hence we're done.

Note that again we can sample from a distribution close to the uniform distribution over QR_n by choosing a random s in $\{1, \ldots, n-1\}$ and letting $x = s^2 \pmod n$.

The proof that inverting Rabin's function is equivalent to factoring can be extended to show that inverting Rabin's function on QR_n is equivalent to factoring n for n a random Blum integer. (This is your homework.)

Thus if factoring such integers is hard we have a trapdoor permutation collection which we can then use to obtain a public key encryption scheme.

Key exchange and the Diffie-Hellman protocol. Alice and Bob can communicate securely over a line eavesdropped by Eve by having Alice generate a keypair (e, d) for a public-key encryption scheme, send to Bob e, and then Bob can send messages to Alice by encrypting them with e.

However, this is not necessarily the only way to do so. A different approach is using a *key exchange protocol*. The first (and still most used) such protocol was given in the same paper by Diffie and Hellman where they first suggested the "crazy" notion of public key cryptography. We'll first present the protocol and then talk about its security goals.

They use the fact that the group $\mathbb{Z}_p^*$ for a prime p is *cyclic*. This means that there is some number $g \in \mathbb{Z}_p^*$ such that $\mathbb{Z}_p^* = \{1, g, g^2, g^3, \ldots, g^{p-2}\}$. g is called a *generator* for the group. In other words, for every element $x \in \mathbb{Z}_p^*$, there is an $i \in \{0, \ldots, p-2\}$ such that $x = g^i \pmod p$. This number i is called the *discrete log* of x with respect to g.

It is known how to efficiently find a generator g for $\mathbb{Z}_p^*$ given a prime p. It is not known how to compute the discrete logarithm and this problem is believed to be hard.

The Diffie-Hellman protocol:

- Alice chooses prime p at random and finds a generator g.
- Alice chooses $x \leftarrow_{\mathrm{R}} \{0, 1, \ldots, p-2\}$ and sends p, g and $\hat{x} = g^x \pmod p$ to Bob.
- Bob chooses $y \leftarrow_{\mathrm{R}} \{0, 1, \ldots, p-2\}$ and sends $\hat{y} = g^y \pmod p$ to Alice.
- Alice and Bob both compute $k = g^{xy} \pmod p$. Alice does that by computing $\hat{y}^x$ and Bob does this by computing $\hat{x}^y$.
- They then use k as a key to exchange messages using a private key encryption scheme.

Clearly, if Eve can compute the discrete log and obtain x from $\hat{x}$ or y from $\hat{y}$ then this protocol is insecure. Thus the assumption that DH key exchange is secure is stronger than the assumption that the discrete log function is hard to compute (or in other words, that the exponentiation function is a one-way permutation). However, as far as we know, this assumption is *not* sufficient for the security of Diffie-Hellman protocol. We need a stronger assumption which is the following:

Decisional Diffie Hellman (DDH) assumption — Take 1. For every prime p and generator g of $\mathbb{Z}_p^*$, the following two distributions A and B over triplets are computationally indistinguishable: $A = \langle g^x, g^{\cdot} g^{xy} \rangle$ for random x and y in $\{1, \ldots, p-2\}$ and $B = \langle g^x, g^y, z \rangle$ for random x and y in $\{1, \ldots, p-2\}$ and $zin\mathbb{Z}_p^*$.

This assumption implies that as far as Eve is considered, the key k is a random element in Z_p^* (i.e., a random number between 1 and $p-1$) and hence can be safely used as a key for any private key encryption scheme. For example, to send a message m of length ℓ, Bob can send Alice $k \oplus m$.

Unfortunately, this assumption is not true (although as far as we know it is "morally true") for a very simple reason: given a number $\hat{y} \in \mathbb{Z}_p^*$, we can check if it has a square root modolu p (i.e., whether it is a quadratic residue). It is known that g^x is a quadratic residue if and only if x is even. Thus, given g^x and g^y we can test whether x and y are even (which happens with probability $1/4$) and in this case g^{xy} will be also a quadratic residue, while a random element in $\mathbb{Z}_p^*$ will only be in QR_p with probability $1/2$.

Fortunately, the assumption can be made for other groups in which it is believed to be true. One such group is the subgroup of quadratic residues mod p, for p of the form $p = 2q+1$. See `http://crypto.stanford.edu/~dabo/abstracts/DDH.html` for more about this assumption.