Lecture 14 - Public Key Cryptography.

Boaz Barak

March 24, 2010

Public key cryptography In the mid 1970's, Diffie and Hellman, and (independently) Merkle, began to challenge the conventional wisdom of $\sim$ 3000 years of cryptography, namely that two parties must exchange some secret information before they can begin to communicate confidentially. Merkle suggested the notion of a *key exchange protocol* in his 1974 class project. This is a protocol in which Alice and Bob can interact over a public channel and as a result obtain a secret key k that is known only to them. He gave a candidate protocol that had non-trivial but rather weak security— the protocol takes T computational steps and an attacker needs to spend roughly T^2 time to break it. ("Time" here is really the number of invocations of a hash function / block cipher, in today's processor speeds, one can perhaps think of $T = 10^9$, $T^2 = 10^{18}$.)

Diffie and Hellman considered the notion of a *trapdoor function.* They conjectured that such a creature exists, and showed that if so, it can be used to obtain both *public key encryption* and *digital signatures*, thus achieving confidentiality and integrity of communication over an open channel. (Merkle also had some thoughts on how to solve the confidentiality issue.)

My understanding of the history is that Diffie and Hellman searched for functions that are easy to compute and hard to invert, and Hellman's colleague John Gill suggested modular exponentiation. The problem was that it's not known if exponentiation has a *trapdoor* that allows to invert it. But then Diffie and Hellman realized that it *can* be used to achieve a potentially exponentially secure version of Merkle's key exchange protocol, hence the protocol known today as Diffie-Hellman key exchange. (The Diffie-Hellman protocol also immediately yields a probabilistic public-key encryption scheme, known today as El-Gamal encryption, though at the time people didn't realized that a secure public-key encryption must be probabilistic.)

After Diffie and Hellman published their 1976 paper, the search for a trapdoor function began, and in 1977 Rivest, Shamir and Adleman (RSA) gave a trapdoor function candidate closely related to the factoring problem. Diffie-Hellman key exchange and public key encryptions and signatures based on RSA are still the most popular public key cryptographic implementations today. A year later in 1978, Rabin gave a trapdoor function inverting which is provably as hard as factoring. Both the RSA and Rabin trapdoor functions are components that need to be instantiated in a proper way with padding etc.. to yield a CPA or CCA secure public key encryption, and many natural instantiations can be *insecure*, as people discovered over the years.

Interestingly, a sequence of works (by Lamport, Goldwasser-Micali-Rivest, Goldreich-Goldwasser-Micali, Goldreich, Naor-Yung, Rompel and others) culminated in showing that digital signatures can be constructed from any one-way function, and hence trapdoor functions are not inherent to their constructions.

Trapdoor functions. 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 functions* (also known as 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 1 (Trapdoor functions.)**.** A *trapdoor function collection* is a collection $\mathcal{F}$ of finite functions such that every $f \in \mathcal{F}$ is a one-to-one function from some set S_f to a set T_f. We require the following properties:

Efficient generation, computation, and inversion There is an probabilistic polynomial-time algorithm G that on input 1^n outputs a pair (f, f^{-1}), where these are two poly(n) size strings that describe the functions f, f^{-1}. That is, the mapping $(x, f) \mapsto f(x)$ and $(y, f^{-1}) \mapsto f^{-1}(y)$ can be computed in polynomial time.

Efficient sampling There is a probabilistic polynomial-time algorithm that given f can output a random element of S_f (or a distribution statistically close to a random element of S_f).

One-wayness The function f is hard to invert. That is, for every polynomial-time A there is a negligible function ϵ such that

$$\Pr_{(f,f^{-1})\leftarrow_{\mathrm{R}} G(1^n), x\leftarrow_{\mathrm{R}} S_f}\left[A(1^n, f, f(x)) = x\right] < \epsilon(n)$$

We remark that this is a slight difference from the Boneh-Shoup definition that does not allow the set S to depend on f, the RSA and Rabin trapdoor functions can be massaged to fit the latter definition, and in fact even to ensure that the set S is $\{0,1\}^n$. Thus, later in the course we will often assume **The Trapdoor Permutation Axiom** that there exists such a trapdoor permutation family with domain and range being $\{0,1\}^n$. (The technical term for such a collections is a doubly enhanced trapdoor permutation collection", see `http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/nizk-tdp.ps`).

Examples of pseudorandom function.

Rabin trapdoor function. The primes and integer factorization have been studied by mathematicians for thousands of years. Despite this, we still don't know of a poly(n)-time algorithm to factor n-digit numbers. This suggests the conjecture that such an algorithm *does not exist.* For cryptography, we need a stronger, average-case, form, and we'll assume that factoring random Blum integers is hard. (A Blum integer is a number $n = pq$ where $p, q = 3 \pmod{()4}$.) Let $\mathcal{B}_n$ denote the set $\{P \in [1..2^n] : P \text{ prime and } p = 3 \pmod 4\}$.

The Factoring Axiom. For every polynomial-time algorithm A there is a negligible function ϵ such that

$$\Pr_{P,Q\leftarrow_{\mathrm{R}}\mathcal{B}_n}[A(P \cdot Q) = \{P, Q\}] \leq \epsilon(n)$$

The following family of is known as Rabin's trapdoor function (we describe below actually a variant due to Blum and Williams).

- Keys: choose P, Q random primes of length n with $P, Q = 3 \pmod 4$, $N = P \cdot Q$. Note that $\phi(N) \pmod 4 = (P-1)(Q-1) \pmod 4 = 2 \cdot 2 \pmod 4 = 0 \pmod 4$.
- 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 $\{1, \dots, N-1\}$ and letting $X = S^2 \pmod N$.

One-wayness of Rabin's function. The key to showing that the function is one-way is the following lemma:

Lemma 1. *Let X, Y be such that $X \neq \pm Y \pmod N$ but $X^2 = Y^2 \pmod N$. Then $gcd(X - Y, N) \notin \{1, N\}$.*

Proof. Since $X \neq \pm Y \pmod N$, $N \not| X - Y$ and $N \not| X + Y$ and so in particular $gcd(X - Y, N) \neq N$. Moreover we know that $X^2 - Y^2 = 0 \pmod N$ and hence $N | (X - Y)(X + Y)$. This implies that $gcd(X - Y, N) \neq 1$, since otherwise we'd have $N | X + Y$. □

This implies that given such X, Y, if $N = PQ$ then we can compute $gcd(X - Y, N)$ to find either P or Q (and then find the other factor by computing $N/gcd(X - Y, N)$). By the same argument we saw last class, if there is an invertor A for the Rabin function that succeeds with probability ϵ , if we choose a random X and let $Y = A(X^2)$, then we have probability at least $\epsilon/2$ that Y will satisfy that $Y^2 = X^2 \pmod N$ but $X \neq \pm Y \pmod N$. Thus Rabin's function is a trapdoor function family under the factoring axiom.

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

- Keys: choose P, Q random primes of length ℓ, $N = P \cdot Q$. Note that $\varphi(N) = |Z_N^*| = (P-1)(Q-1)$. Choose e at random from $Z_{\varphi(N)}^*$ (that is, $gcd(e, \varphi(N)) = 1$.[1] Note that $\varphi(N)$ is *even* and hence, unlike in the Rabin case, e can not equal 2.

[1]Choosing e at random is just one possibility, one can also fix e to be any number in $\mathbb{Z}_{\varphi(N)}^* \setminus 1$, and different choices have been considered in the literature, in particular people often choose $e = 3$ or e that is a prime of the form $e = 2^i + 1$, to get faster exponentiation.

- Forward (public) key: N, e
- Backward (inversion/trapdoor) key: d such that $d = e^{-1} \pmod{\varphi(N)}$. That is, $ed = k\varphi(N) + 1$. Note that d can be computed from $\varphi(N)$ (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 $X^{\varphi(N)} = 1$. Hence $X^{ed} = X^{k\varphi(N)+1} = X^{k\varphi(N)}X = 1 \cdot X$.

Note that we can generate a random element of $\mathbb{Z}_N^*$ by choosing a random number X in $0, 1, \dots, N-1$ 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 function. 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.

Using trapdoor functions for public key encryption Exercise...

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 $\mathbb{Z}_P^* = \{1, g, g^2, g^3, \dots, g^{P-2}\}$. g is called a *generator* for the group. In other words, for every element $X \in \mathbb{Z}_P^*$, there is an $i \in \{0, \dots, P-2\}$ 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 $X \leftarrow_{\mathrm{R}} \{0, 1, \dots, P-2\}$ and sends P, g and $\hat{X} = g^X \pmod P$ to Bob.
- Bob chooses $Y \leftarrow_{\mathrm{R}} \{0, 1, \dots, P-2\}$ 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.

Different types of permutations. It's important not to get confused between *pseudorandom permutations* (PRP), *one way permutations* (OWP) and *trapdoor permutations* (TDP). Both *one-way permutations* and *pseudorandom permutations* are symmetric primitives, that are related to *private key* cryptography. A one-way permutation is just one function rather than a collection of functions, and it has the property that it's easy to compute but hard to invert. A pseudorandom permutation collection has the property that it's indistinguishable from a random permutation for an adversary that *doesn't know the key.* The crucial difference in a trapdoor permutation is that an adversary that *is given the (forward) key* still cannot invert it, even though this is easy to do using the backward (i.e. trapdoor) key.