CS 4501-6501 Topics in Cryptography 23 Mar 2018

Lecture 9, Trapdoor Permutations: RSA and Digital Signatures

Lecturer: Mahmoody *Scribe: Caleb Smith, Danny McNamara, Ahmadreza Rahimi*

1 Introduction

Last time, we defined public-key encryption and covered the Diffie-Hellman key agreement protocol. Here, we will instantiate public-key encryption, in particular we will see the RSA encryption (invented by Rivest, Shamir and Adlemant in late 80s). We will also cover digital signatures in the public-key setting. The core idea behind RSA's encryption is a notion called "trapdoor permutations".

2 Trapdoor Permutations

The main idea behind a trapdoor permutation, π : {0, ..., N-1} $\rightarrow$ {0, ..., N-1}, is that π is a bijection that is easy to compute and hard to invert without the trapdoor. Formally, we have a key generation algorithm (similar to PKE) that works as follows. When we run Gen(1^n) (for security parameter $n \approx \log N$) we obtain the following.

Find $ek = \pi(\cdot)$ and $dk = \pi^{-1}(\cdot)$ for permutations $\pi(\cdot),\ \pi^{-1}(\cdot)$ such that:

- $\forall\ x \in \{0, ..., N-1\}\ \pi^{-1}(\pi(x)) = x$
- $\forall$ poly-time adversaries, A, $Pr_{x \leftarrow^{\$} \{0,...N-1\}}[A(\pi(x)) = x] \leq \text{negl}(n)$.

It seems natural to use Trapdoor Permutations for a public-key encryption scheme by publicly revealing π for encryption and keeping π^{-1} for decryption, but the naive approach does not quite work to give us CPA secure encryption. This is because the most straightforward way of doing this does not involve any randomness in encryption, therefore it cannot be CPA secure. We will now show how to incorporate randomness to encrypt and decrypt a single bit b.

- To encrypt: pick $r, s \in \{0, ..., N-1\}$ at random and send $[\pi(r), s, \langle r, s\rangle \oplus\ b]$, where $\langle r, s\rangle$ is defined to be $\oplus_{i=1}^{n} r_i \cdot s_i$
- To decrypt: first apply π^{-1} to $\pi(r)$ to get back r, then compute $\langle r, s\rangle$ and xor the result with $\langle r, s\rangle \oplus\ b$ to get back the bit b.

The Goldreich-Levin theorem (Theorem 7.5 in Katz and Lindell) states that for the above scheme, if π is a one-way permutation, then for all poly-time adversaries, A, $\Pr_{r,s}[A(s, \pi(r)) = \langle r, s\rangle] \leq \frac{1}{2} + neg(n)$. Therefore, the above method of encrypting one bit is actually CPA

secure. Based on what we proved in the problem set 2, (we did it for private key encryption, but the same proof holds for public key encryption as well) if we encrypt a message bit by bit using fresh randomness, it will allow us to encrypt large messages in CPA secure way.

As you can see, there is a large overhead for sending a single encrypted bit using that technique, so now we will present a more efficient scheme proposed by Bellare and Rogaway [BR95] using "ideal" hash functions (also known as random oracle model).

- Let $h : \{0,1\}^n \to \{0,1\}^n$ be an "ideal" hash function and have the key generation algorithm produce $ek = \pi(\cdot)$ and $dk = \pi^{-1}(\cdot)$.
- To encrypt $m \in \{0,1\}^n$: randomly pick $r \leftarrow \{0,1\}^n$ and output $c = \pi(r), h(r) \oplus m$
- To decrypt $c = (y, z)$, output $h(\pi^{-1}(y)) \oplus z$

The trade off for this scheme is that the security is based on a heuristic, namely they showed that if $(\pi(\cdot), \pi^{-1}(\cdot))$ are a secure trapdoor permutation and $h(\cdot)$ is a truly random function, then the above scheme is provably CPA secure. The obvious problem here is that we cannot have a hash function which is a truly random function. But the good thing is this unrealistic modeling of hash functions still capture what we can really do with them, namely, they look random to use till we compute them on a point, which is also the case for a random oracle. Now that we have seen what trapdoor permutations are and how they can be used for public-key encryption, we will now show how we can instantiate the object using number theoretic arguments, which will ultimately be the crux of RSA encryption.

3 RSA Trapdoor Permutation

3.1 Number Theory Review

Before we continue to describe RSA algorithm we need to brief review to Number Theory. For $x, y \in \mathbb{Z}$ lets define $\gcd(x, y)$ as the greatest number $z \in \mathbb{Z}$ that $z|x$ and $z|y$. And we say x and y are relatively prime if $\gcd(x, y) = 1$.

For any $n \in \mathbb{N}$ lets define $\varphi(n)$ as a function that counts the number of positive integers that are less than n and are relatively prime to n, this function is also called Euler's totient function or Euler's phi fucntion.

Example 3.1. $\varphi(6) = 2$ since the numbers that are less than 6 and are relatively prime to 6 are $1, 5$.

Example 3.2. For any prime number p, $\varphi(p) = p - 1$ since all the numbers $1, \cdots, p-1$ are relatively prime to p.

Example 3.3. If p, q are prime numbers, $\varphi(pq) = (p-1)(q-1)$. proving this is easy by just counting the numbers that are relatively prime to pq. The same argument applies to $\varphi(p_1 \cdots p_n) = \prod_{i=1}^{n}(p_i - 1)$.

Theorem 3.4 (Euler's Theorem)**.** *If* $\gcd(a, N) = 1$ *then*

$$a^{\varphi(N)} \equiv 1 \ (mod\ N)$$

RSA Trapdoor permutation Now we want to use this theorem to come up with a trapdoor permutation. Now we know if $\gcd(a, N) = 1$ then $a^{k\cdot\varphi(N)} \equiv 1 \pmod{N}$ where k is an integer, so

$$a \times a^{k\varphi(N)} = a^{1+k\cdot\varphi(N)} \equiv a \pmod{N}$$

so if there are two numbers e, d such that $e \cdot d = 1 + k\varphi(N)$ (or equivalently $e \cdot d \equiv 1 \pmod{\varphi(N)}$ then $a^{e\cdot d} \equiv a \pmod{N}$. Therefore, we can pretend the computation of $a^{e\cdot d}$ is the same as applying the "trapdoor permutation" π and its inverse π^{-1} on a, where $\pi(a) = a^e$ and $\pi^{-1}(b) = b^d \pmod{N}$.

Here the trapdoor key for the inverse function is d (and N) and the public parameter for trapdoor function is e (and N). This is the description of the trapdoor permutation proposed for RSA where they (carefully) choose N to be product of two prime numbers p, q. Informally speaking the security that it is hard to invert π *requires* that it is also hard to calculate $\varphi(N)$ by just knowing N when N is the product of two large prime numbers, because if we calculate $\varphi(N)$ then it is easy to compute d knowing $\varphi(N)$ and e (Euclid's algorithm for gcd allows computing "multiplicative inverse" of e mod $\varphi(N)$, which d).

What about a that is not prime relative to N? The argument above shows that if a is prime relative to N, then $a^{ed} = \pi^{-1}(\pi(a)) = 1$, but for a trapdoor permutation, we need this to hold for all $a \in \{0, 1\}0, \ldots, N-1$. Interestingly, it can be shown that for the case of $N = pq$, and if $gcd(a, N) = p$ or q, then $a^{ed} = \pi^{-1}(\pi(a)) = 1$ holds as well. (In problem set we will discuss why we would be fine even if this would *not* happen for such a that is not prime relative to N, simply because there are few of them, and picking a at random will not be like that for $1 - \text{negl}(n)$ probability).

How to actually encrypt? As we discussed above, the most native way of using RSA trapdoor permutations for encryption and decryption, which uses $m^e \mod N$ as encryption of $m \in \{0, \ldots, N-1\}$ and $c^d \mod N$ for decryption of c. But as we also saw above, using Goldreich-Levin's method or ideal hash functions, we can still use RSA as a trapdoor permutation, in a randomized way, and do the encryption and decyprtion properly.

4 Public Key Authentication: Digital Signatures

Public key is an example of asymmetric-key cryptography. This made a huge difference, as two parties who have not met could communicate securely over a public channel.

Now, we study the same thing for authentication. Instead of using the same key for both encryption and decryption, digital signatures introduce a pair of keys: one to encrypt (the signature) and one to verify the message.

Example 4.1. Consider a scenario in which Alice is sending a message m to Bob. In order to prove to Bob that she wrote the message, Alice 'signs' the message by encrypting it with her private signing key *sk*. Using *sk*, Alice signs m with $\pi^{-1} = \text{Sign}_{sk}(m)$. Bob receives the message and uses Alice's verification key *vk* (which is broadcast by Alice, just like how we

broadcast public encryption keys in the context of encryption) to verify the message m and its signature.

The goal of an adversary in a security game designed around Example 4.1 would be to forge a signature such that they could generate a pair (m, t) that passes the $\mathsf{Vrf}_{vk}(m, t) = 1$ equation. This could happen while the adversary gets to see a bunch of correct signatures on various messages signed by Alice, so to model this, we design a security game such that, the Challenger possesses a private signature key that is unknown to the adversary. The adversary has access to a public verification key and the ability to request keys from a Signing Oracle for a message. The probability of an adversary win should be negligible.

Security Game of Signature Schemes. Suppose $\mathcal{S} = (\text{Gen}, \text{Sign}, \mathsf{Vrf})$ is a signature scheme. In the security game we do as follows.

1. $\text{Gen}(1^n)$ is executed and sk, vk keys are generated.
2. Adversary receives vk, and can ask polynomially many messages to be signed $m_1, \ldots, m_k$ and it receives $t_1 = \text{Sign}_{sk}(m_1) \ldots, t_k = \text{Sign}_{sk}(m_k)$.
3. The adversary outputs some $m \notin \{m_1, \ldots, m_k\}$ and some signature t.
4. Adversary wins if $\mathsf{Vrf}_{vk}(m, t) = 1$.

Definition 4.2. We call a signature scheme $\mathcal{S} = (\text{Gen}, \text{Sign}, \mathsf{Vrf})$ secure, if for all poly(n)-time adversary n, the probability of winning in the security game above is at most negl(n).

4.1 Using trapdoor permutations for signatures

As in the case of public key encryption, there is a naive way of using trapdoor permutations to do the signing which is not secure. The idea is to use the public permutation π for the public job of verification, and keep the private trapdoor permutation π^{-1} for the private task of signing. Let's see how it works:

Naive way to use TDPs for signing:

- To generate the keys, we generate a pair of permutation and its inverse (trapdoor) using the generation algorithm for TDPs. Namely, we obtain π, π^{-1}, and let the public verification key vk be $\pi(\cdot)$ (description of permutation) and the private key signing key sk be $\pi^{-1}(\cdot)$ (the descryption fo the trapdoor).
- To sign a message m in permutation's domain, publish signature t where t is the tag
- To verify the message and tag, *(m, t)*, accept if and only if $\pi(t) = m$

The issue. This signature scheme is *not* secure according to our definition of security. Namely, the adversary can find a pair of (m, t) that passes the test of signature verification. In particular, the adversary could choose any value for t (in the range of the permutation) and *then* generate a corresponding message m using the public verification key that would pass the verification test, namely, adversary lets $m = \pi(t)$ which is efficiently computable. However, in practice, this does not matter to many people because the message perhaps will not contain any meaningful content. This is still bad practice, because for some contexts it *might* become a relevant message m, and it is not a surprise, because it is insecure by the formal definition.

4.1.1 Hash and Sign

By applying an ideal hash to the original message, it is easy to make the previous signature method secure. First, we apply the hash to the message and then apply the π^{-1}. So the signature would be $t = \pi^{-1}(h(m))$. This is called the "hash and sign" method, but note that the whole thing is actually signing! The intuition behind the security is that, now in order to win, an adversary would need to begin with t and invert the hash function in order to determine the original message. This is not possible efficiently if we use a secure Trapdoor Permutation. We will talk about this more next time.

References

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. pages 92–111, 1995.