Lecture 15 - Digital Signatures

Boaz Barak

March 29, 2010

Reading KL Book Chapter 12.

Review
- Trapdoor permutations - easy to compute, hard to invert, easy to invert with trapdoor.
- RSA and Rabin signatures.

Definition of digital signatures. Recall that we had the following picture:

	Private Key	**Public Key**
Secrecy	Private Key Encryption	Public key Encryption
Integrity	Message Authentication Codes (MAC)	??

Digital signatures complete this picture by giving a public key analog of message authentication codes. Digital signatures were suggested by Diffie and Hellman in their seminal paper, but unlike the case of public key encryption schemes (where they had a key exchange protocol that could be made into a probabilistic encryption scheme) they did not have a reasonable candidate for such signatures until the RSA system was invented a year later by Rivest, Shamir and Adelman. However, even the RSA system was quickly seen not to have sufficient security and only later Goldwasser, Micali and Rivest gave what is now considered to be the "right" definition for digital signatures, and also a factoring-based construction meeting this definition. This definition is called *existential forgery under chosen-message attack* but we'll simply call it secure signature schemes.

Definition 1 (Digital Signatures)**.** A triplet of algorithms $\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}$ is called a (T, ϵ)-*secure signature scheme* if it satisfies the following properties:

Validity For every pair $(s, v) \leftarrow \mathsf{Gen}(1^n)$, and every $m \in \{0,1\}^n$, we have that

$$\mathsf{Ver}_v(m, \mathsf{Sign}_s(m)) = 1$$

Security For every T-time circuit A, we have that

$$\Pr[A^{\mathsf{Sign}_v(\cdot)}(v) = (m, \sigma) \text{ st } m \text{ wasn't queried by } A \text{ and } \mathsf{Ver}_v(m, \sigma) = 1] < \epsilon$$

Again, a scheme is simply secure if it is (T, ϵ)-secure for super-polynomial T and ϵ.

Applications. Digital signatures have many applications and are widely used today in the world. Some practical application include verifying websites such as **amazon.com** and verifying code such as drivers for Windows and upgrades for embedded devices. In fact, signature keys tend to be much "longer-lived" than encryption keys: they are often hardwired into

various devices and there's no mechanism to replace them. Thus, there are fixed and well known public verification keys today whose corresponding secret keys are worth many millions of dollars. For this reason protecting the private signature keys is often more critical than protecting the private decryption keys, and time permitting, we may discuss some techniques for such protection in this class.

History For some time, people thought that it will be impossible to achieve signature schemes with a proof of security. The reasoning was that the proof of security will have to be an efficient way to transform a forged signature into, say, a factoring of n, where $n = pq$ is the public key. However, if there is such a way then the scheme cannot be secure under a chosen message attack. However, this reasoning is flawed and in 1984 this was demonstrated by Goldwasser, Micali and Rackoff who gave the first (albeit stateful) signature scheme based on the hardness of factoring. This was later improved by Goldreich to a stateless signature scheme. Surprisingly, results of Naor and Yung, and Rompel show that secure signature schemes can be constructed based on much weaker assumptions without using any number theory - The PRG (or OWP) Axiom suffices.

All these signature schemes are still not quite efficient enough for practical use. More efficient constructions were given by Gennaro, Halevi and Rabin , and by Cramer and Shoup, under some stronger variants of the Diffie-Hellman and RSA assumptions.

Efficient signatures with heuristic analysis We now describe an efficient construction a stateless, many-time signature scheme. This construction (or closely related variants) is widely used. However, there is no known proof of security for this construction under any reasonable assumption. Rather, we'll only give a *heuristic argument* (given by Bellare and Rogaway) why this scheme may be secure.

Plain trap-door based signatures To get some intuition for the scheme, let's recall the original suggestion of Diffie and Hellman for a signature scheme: Their idea was to use a trapdoor permutation (which they thought should exist but didn't have a candidate for, until RSA came up with one). To obtain a signature for a message m, one treats m as an image/ciphertext and inverts or "decrypts" it to obtain the signature σ.

Plain Rabin signatures More concretely, say, for the Rabin trapdoor collection this signature scheme is the following:

Key generation Choose two random p, q that are equal to 3 (mod 4), these are the secret/signing key. The public key is $n = p \cdot q$.

Signing To sign a message m, output $\sigma = \sqrt{m} \pmod{n}$ (fix some choice for one of the four possible roots, for example, we can always output the one root that is itself a quadratic residue).

Verification To verify that σ is a valid signature for m, check that $\sigma^2 = m \pmod{n}$.

Assuming the factoring is hard, if m is chosen at random then it should be hard to come up with a signature for m. However, it's clear that the performance of this scheme under chosen message attack is catastrophic: if you can make a single query for a message of your choice, you'll choose a random $x \in \mathbb{Z}_n^*$ and let $m = x^2 \pmod{n}$. Given $\sigma = \sqrt{m} \pmod{n}$ there's probability $1/2$ that $\sigma \neq \pm x \pmod{n}$ in which case $gcd(\sigma - x, n)$ will yield a non-trivial factor of n.

(As a side note, for some time many people thought that similar problems will happen for any factoring-based signature scheme, and so they believed that the existence of a reduction to a hard problem like factoring is incompatible with being secure against a chosen-message attack.)

Fixing the problem Despite this problem, similar trapdoor-permutation based schemes are widely used in practice (the underlying permutation is typically RSA, but as you saw in the exercises, it has similar problems). Of course some change must be made in the scheme, and it is the following one: we use the hash and sign paradigm, but now we hope that not only the hash function lets us sign long messages, but it is also "crazy" enough to foil such attacks.

For example, we hope that it is infeasible, given the hash function h, for an attacker to find an input m and a value x such that

$$x^2 = h(m) \pmod{n} \tag{1}$$

Note that the fact that h is collision resistent does not tell us anything about the hardness of this question. For example, think of h as the identity function from $\mathbb{Z}_n^*$ to $\mathbb{Z}_n^*$: this function does not have any collisions at all. However, clearly we can choose $m = x^2$ and then since $h(m) = m$ we can find a solution for Equation 1.

This property is only an example: it is not sufficient to ensure security of the scheme. For example, the scheme will also be broken if one can do the following: find m, and a quadratic residue c such that

$$h(c \cdot m) = c \cdot h(m) \tag{2}$$

the reason is that if this holds then we can ask for a signature σ on m, and then that $\sqrt{c} \cdot \sigma$ is a signature for $c \cdot m$.

Making this provably secure. Ideally we should be able to proceed at this point in a similar way to what we did in the past:

1. Give a precise definition for a hash function collection $\{h\}$ being "sufficiently crazy".
2. Show a construction of "sufficiently crazy" hash functions based on some reasonable assumptions (e.g., factoring, DDH, PRG Axiom 1....)
3. Deduce that with the right choice of hash functions, the Rabin hash-and-sign signature scheme is secure.

Unfortunately, we don't know how to do that. In fact, we don't know even how to get step 1 (except for the uninteresting definition that a hash function is defined to be "sufficiently crazy" if it makes the Rabin scheme secure against chosen-message attacks, which will make us stuck in step 2).

This means we're in an uncomfortable situation: we have an efficient and attractive construction, that people use in practice, we don't know that it is *insecure* (we don't have an attack on it), but we also don't have anything intelligent to say about its security.

The Bellare-Rogaway analysis. Bellare and Rogaway (building on work by Fiat and Shamir) suggested an approach to try to justify the security of such schemes, and also a methodology to design such schemes. There idea was to analyze the scheme as if h was a completely random function, that is given to the adversary as a black-box, and to see if it is secure in

this setting. If it is, we say that it is secure in the *random oracle model*. Since intuitively, we think of crazy hash functions as having random-like behavior, if the scheme is not secure even if h is a random function then it's probably insecure (and we can find an attack) for any instantiation of that with a particular hash function.

The question is what happens if the scheme *is* secure when h is a random function (as [BR] proved is the case for the trapdoor-permutation hash-and-sign scheme). The first idea that comes to mind is that then we can make it secure by using a function from a *pseudorandom function collection* instead of h. (Indeed, in the original paper of Fiat and Shamir suggesting a use of this paradigm they (mistakenly) claimed that this will work.) However, there's a big problem here: the definition of pseudorandom functions says that an adversary can not tell apart a black-box computing a random function from a black-box computing a random function from the collection. It says *nothing* about what happens if the adversary is actually given the seed/key/description of the pseudorandom function, as is the case in the hash-and-sign signatures. In fact it is clear that if the adversary is given the seed s then it can trivially distinguish between a black box computing $f_s(\cdot)$ to a black-box computing a random function.

We'd like to define something like "public-key/publicly evaluatable pseudorandom functions" that remain pseudorandom even to someone that knows the key, but it's not known how to make such a definition that is both useful and not impossible to achieve.

Nevertheless, Bellare and Rogaway argued that if one proves that a scheme is secure in the random oracle model then it says something positive about the security of the real scheme, where h is replaced by a cryptographic hash function such as, say, $SHA-256$. (In fact, one have to be more careful than that, see Section 8.8.3 in the Boneh-Shoup book.) Their argument was that even if we can't pinpoint what's exactly the security properties of the hash functions that we need, the existence of a proof in the random oracle model implies that sufficiently crazy hash functions will be good enough, and that any attack on the scheme will necessarily have to find some weakness in the design of the cryptographic flaw. Thus, roughly speaking, they put forward the following conjecture/thesis (this is my phrasing of the thesis in their paper):

The Random-Oracle Thesis: If a protocol has a proof of security in the random-oracle model, then it will be secure when instantiated with a "sufficiently crazy" hash function.

In the 12 years that passed since their paper, this thesis has been experimentally verified and mathematically refuted.

It was experimentally verified in the sense that no one has yet found an attack against the schemes suggested in their paper. There are also many other schemes that were proven secure in the random oracle model and so far have not been broken.

It was mathematically refuted by Canetti, Goldreich and Halevi in 1998, showing that there in fact exist protocols that are secure in the random oracle model, but can be attacked *no matter what* hash function collection is used. There were further results on this topic, see for example `http://www.cs.ut.ee/~lipmaa/crypto/link/rom/` for more info.

Signatures without random oracles There are efficient number-theoretic based constructions of signatures not using random oracles, see Section 13.5 in the Boneh Shoup book.

Signatures from one-way functions It turns out that signatures can be constructed from The PRG (or OWP/OWF) Axioms only. We now sketch the construction, although we'll use collision resistant hash functions as a component.

One time signature scheme We start by presenting a *one-time* signature scheme (due to Lamport) that remains secure if the attacker can only make a single query to the signing oracle. In fact, we'll consider an even simpler variant: a signature for a single bit. Thus, the attack is that the adversary chooses a bit $b \in \{0,1\}$, gets a signature for b and needs to forge a signature for $\overline{b} = 1 - b$. We'll base this on the OWP Axiom: the existence of a *one-way permutation* $f(\cdot)$ that is a one-to-one function $f : \{0,1\}^n \to \{0,1\}^n$ such that for every polynomial-time A, $\Pr_{x \leftarrow_R \{0,1\}^n}[A(f(x)) = x] < n^{-\omega(1)}$.

Key generation $\mathsf{Gen}(1^n)$ chooses $x^0, x^1 \leftarrow_R \{0,1\}^n$ and computes $y^b = f(x^b)$ for $b = 0, 1$. The private signing key is $s = (x^0, x^1)$ and the public verification key is $v = (y^0, y^1)$.

Signing algorithm To a sign a bit $b \in \{0,1\}$, $\mathsf{Sign}_s(b) = x^b$.

Verification $\mathsf{Ver}_v(b, x) = 1$ iff $f(x) = y^b$.

It is a simple exercise to verify that this scheme is secure under a single-query chosen message attack.

Extending to longer messages It is clear how to extend a single bit scheme into a scheme for signing ℓ bits: just generate ℓ independent public/private key pairs.

Signing messages longer than the key length. One drawback of that scheme (other than it is one-time) is that to sign a message of length ℓ, we need a key of length $n \cdot \ell$. This turns out to be a serious bottleneck in converting a one-time signature scheme into a standard (many-times) scheme. To overcome this, we'll need the notion of a *collision resistant hash function*. Recall that that this is a collection of functions H such that each function maps say $2n$ bit long strings into n bit long strings and so it's definitely *not* one-to-one but given such a function it is infeasible to demonstrate that it is not one-to-one (i.e., to find a *collision*: two values $x \neq x'$ such that $h(x) = h(x')$). The formal definition is the following:

Definition 2 (Collision-resistant hash functions)**.** A collection of functions $\{h_k\}_{k \in \{0,1\}^*}$, with $h_k : \{0,1\}^{2n} \to \{0,1\}^n$ for $k \in \{0,1\}^n$, is called (T, ϵ)-*collision resistant* if the function $(k, x) \mapsto h_k(x)$ is polynomial-time computable and for every T-time A we have that

$$\Pr_{k \leftarrow_R \{0,1\}^n}[A(k) = (x, x') \text{ st } h_k(x) = h_k(x')] < \epsilon(n)$$

Collision resistant hash functions are known to exist based on the assumption that factoring is hard (Goldwasser, Micali and Rivest) and there are also several efficient candidates for collision resistant functions (e.g., SHA-256). In recent years there had been some non-trivial attacks on some of these candidates (namely MD-5 and ShA-1), but there are still others that are believed to be secure.

Naor and Yung defined a weaker notion of collision resistant that can be sufficient for the application of signature schemes, and showed that it can be achieved from one way permutations. Rompel then improved this to use only one-way functions.

A scheme with message size > key size : It is not hard to show that given a collision resistant hash function h mapping $\{0,1\}^{2n}$ to $\{0,1\}^n$ we can extend it to a function mapping say $\{0,1\}^{n^3}$ to $\{0,1\}^n$. Therefore we can have the following scheme:

Components: A signature scheme $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ that uses n^2 long keys to sign n long messages. A hash function collection $\{h_k\}_{k\in\{0,1\}^*}$ where for $k \in \{0,1\}^n$, $h_k : \{0,1\}^{n^3} \to \{0,1\}^n$. For convenience we use h to denote both the function itself and its key k, thus we think of h as both a function from $\{0,1\}^{n^3}$ to $\{0,1\}^n$ and an n bit string.

Key generation $\mathsf{Gen}(1^n)$ chooses a pair (s', v') of a signature scheme for messages of length n, and a hash function $h : \{0,1\}^{n^3} \to \{0,1\}^n$. The public key is (h, v') while the private key is s'.[1] Note that the length of the keys is $n^2 + n \ll n^3$.

Signing algorithm To a sign a message $m \in \{0,1\}^{n^3}$, compute $m' = h(m)$ and output $\sigma = \mathsf{Sign}'_{s'}(m')$.

Verification $\mathsf{Ver}_v(m, \sigma) = 1$ iff $\mathsf{Ver}_{v'}(h(m), \sigma) = 1$.

This transformation applies equally well to one time and many time schemes. In both cases the new scheme inherits the security of the old scheme. The idea of the proof is that as long as the adversary doesn't find a collision in the hash function, we can convert an attack on the new scheme to an attack on the underlying scheme.

From one-time to many-times scheme. We do this in two steps: (1) first get a *stateful* signature schemes using a tree of signatures/certificates, and (2) make it stateless using pseudorandom functions. See Section 14.5.

Thoughts for Wednesday We used MACs to transform CPA-secure private key encryption to CCA-secure encryption. Can we do the same for *public key* encryption? In such a construction, who will need to know the secret signing key - the encryptor or decryptor?

Also, do you think it's reasonable to expect a consumer to maintain a signing key?

What would be worse - if Amazon's decryption key is compromised or its signing key?

[1] When specifying a private key we always ignore the parts that are already present in the corresponding public key.