Lecture 17 - Digital Signatures (continued)

Boaz Barak

November 17, 2005

Plan Last time we saw (up to missing details of the construction of collision-resistant hash functions) how to construct a secure digital signature scheme based on the assumption that factoring large Blum integers is hard.

However, this construction, with its use of roughly n one-time signatures to generate one signature, is not the most efficient known construction. As mentioned in the previous lecture notes, there are more efficient constructions that are provably secure based on number theoretic assumptions, but their analysis is a bit complicated so we may not get to see them in this course.

In this lecture I will discuss an efficient construction a stateless, non-interactive 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.

In addition, if we have time, we'll show how one can transform a zero knowledge proof system into an *interactive* signature scheme. We can also transform zero knowledge systems into standard (non-interactive) signatures with a heuristic argument for security (as was done first by Fiat and Shamir). Such schemes are also used often in practice.

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'll see 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, Axiom 1, Axiom 2,...)
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, $SHA-256$. 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.

To be completed