Luca Trevisan

Notes for Lecture 20

Summary

Today we begin to talk about *signature schemes*.

We describe various ways in which "textbook RSA" signatures are insecure, develop the notion of *existential unforgeability under chosen message attack*, analogous to the notion of security we gave for authentication, and discuss the difference between authentication in the private-key setting and signatures in the public-key setting.

As a first construction, we see Lamport's *one-time signatures* based on one-way functions, and we develop a rather absurdly inefficient stateful scheme based on one-time signatures. The scheme will be interesting for its idea of "refreshing keys" which will be used next time to design a stateless, and reasonably efficient, scheme.

1 Signature Schemes

Signatures are the public-key equivalents of MACs. The set-up is that Alice wants to send a message M to Bob, and convince Bob of the authenticity of the message. Alice generates a public-key/ secret-key pair (pk, sk), and makes the public key known to everybody (including Bob). She then uses an algorithm $Sign()$ to compute a *signature* $\sigma := Sign(sk, M)$ of the message; she sends the message along with the signature to Bob. Upon receiving M, σ, Bob runs a verification algorithm $Verify(pk, M, \sigma)$, which checks the validity of the signature. The security property that we have in mind, and that we shall formalize below, is that while a valid signature can be efficiently generated given the secret key (via the algorithm $Sign()$), a valid signature cannot be efficiently generated without knowing the secret key. Hence, when Bob receives a message M along with a signature σ such that $Verify(pk, M, \sigma)$ outputs "valid," then Bob can be confident that M is a message that came from Alice. (Or, at least, from a party that knows Alice's secret key.)

Syntactically, a signature scheme is a collection of three algorithms $(Gen, Sign, Verify)$ such that

- $Gen()$ takes no input and generates a pair (pk, sk) where pk is a public key and sk is a secret key;

- Given a secret key sk and a message M, $Sign(sk, M)$ outputs a signature σ;
- Given a public key pk, a message M, and an alleged signature σ, $Verify(sk, M, \sigma)$ outputs either "valid" or "invalid", with the property that for every public key/ secret key pair (pk, sk), and every message M,

$$Verify(pk, M, Sign(sk, M)) = \text{“valid''}$$

The notion of a signature scheme was described by Diffie and Hellman without a proposed implementation. The RSA paper suggested the following scheme:

- Key Generation: As in RSA, generate primes p, q, generate e, d such that $ed \equiv 1 \bmod (p-1) \cdot (q-1)$, define $N := p \cdot q$, and let $pk := (N, e)$ and $sk := (N, d)$.
- Sign: for a message $M \in \{0, \ldots, N-1\}$, the signature of M is $M^d \bmod N$
- Verify: for a message M and an alleged signature σ, we check that $\sigma^e \equiv M \pmod{N}$.

Unfortunately this proposal has several security flaws.

Ideally, we would like the following notion of security, analogous to the one we achieved in the secret-key setting.

Definition 1 *A signature scheme (G, S, V) is (t, ϵ) existentially unforgeable under a chosen message attack if for every algorithm A of complexity at mot t, there is probability $\leq \epsilon$ that A, given a public key and a signing oracle, produces a valid signature of a message not previously sent to the signing oracle.*

2 One-Time Signatures and Key Refreshing

We begin by describing a simple scheme which achieves a much weaker notion of security.

Definition 2 (One-Time Signature) *A signature scheme (G, S, V) is a (t, ϵ)-secure one-time signature scheme if for every algorithm A of complexity at mot t, there is probability $\leq \epsilon$ that A, given a public key and one-time access to a signing oracle, produces a valid signature of a message different from the one sent to the signing oracle.*

We describe a scheme due to Leslie Lamport that is based on one-way function.

Theorem 3 *Let $f : \{0,1\}^n \to \{0,1\}^n$ be a (t, ϵ) one way function computable in time r. Then there is a one-time signature scheme (G, S, V) that signs messages of length ℓ and that is $(t - O(rl), \epsilon \cdot 2\ell)$ secure.*

A disadvantage of the scheme (besides the fact of being only a one-time signature scheme) is that the length of the signature and of the keys is much bigger than the length of the message: a message of length ℓ results in a signature of length $\ell \cdot n$, and the public key itself is of length $2 \cdot \ell \cdot n$.

Using a collision resistant hash function, however, we can convert a one-time signature scheme that works for short messages into a one-time signature scheme that works for longer messages. (Without significantly affecting key length, and without affecting the signature length at all.)

Theorem 4 *Suppose (G, S, V) is a (t, ϵ) secure one-time signature scheme for messages of length ℓ, which has public key length kl and signature length sl. Suppose also that we have a (t, ϵ) secure family of collision-resistant hash functions $H : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^\ell$. Suppose, finally, that H, G, S all have running time at most r.*

Then there exists a $(t - O(r), 2\epsilon)$ secure one-time signature scheme (G', S', V') with public key length $kl + k$, signature length sl and which can sign messages of length m.

In particular, given a one-way function and a family of collision-resistant hash functions we can construct a one-time signature scheme in which the length of a signature plus the length of the public key is less than the length of messages that can be signed by the scheme.

If (G, S, V) is such a one-time signature scheme, then the following is a stateful scheme that is existentially unforgeable under a chosen message attack.

Initially, the signing algorithm generates a public key/ secret key pair (pk, sk). When it needs to sign the first message M_1, it creates a new key pair (pk_1, sk_1), and generates the signature $\sigma_0 := S(sk, M_1 || pk_1)$. The signature of M_1 is the pair (σ_0, pk_1). When it, later, signs message M_2, the signing algorithm generates a new key pair (pk_2, sk_2), and the signature $\sigma_1 = S(sk_1, M_2 || pk_2)$. The signature of M_2 is the sequence

$$M_1, pk_1, \sigma_0, pk_2, \sigma_1$$

and so on. Of course it is rather absurd to have a signature scheme in which the signature of the 100th message contains in its entirety the *previously signed* 100 messages along with their signatures, but this scheme gives an example of the important paradigm of *key refreshing*, which will be more productively employed next time.