Abstract. In this lecture we introduce the notion of Digital Signatures and present relevant constructions. Readings from Chapter 7 of D. Stinson.

1 Notion of Digital Signatures

A method for signing a document that is stored in electronic form. A way to uniquely identify the signer by anyone, and attribute the signature to him/her.

Definition 1. A digital signature scheme is a five-tuple \((P, A, K, S, V)\), where

1. \(P\) is a finite set of possible messages.
2. \(A\) is a finite set of possible signatures.
3. \(K\) is a finite set of possible keys, where keys are randomly generated thus creating a pair of keys \((p, s)\).
4. \(\text{sig}_K(m) \in S\) is a signing algorithm corresponding to key \(K\), that takes as an input \(K, m\) and outputs a signature \(\sigma\) often called a tag. The algorithm \(\text{sig}_K : P \rightarrow A\) may be randomized (i.e., a different signature can be generated for the same message), or stateful (i.e., it takes into account some internal state such as clock or counter), or both.
5. \(\text{ver}_K \in V\) is a verification algorithm that takes as an input \(K, \sigma\) and outputs true, false (or \((1,0))\). The verification algorithm \(\text{ver}_K : P \times A \rightarrow \{1, 0\}\) satisfies that for every message \(m \in P\) and for every signature \(\sigma \in A\),

\[
\text{ver}_K(m, \sigma) = 1, \quad \sigma = \text{sig}_K(m), \quad (1)
\]
\[
\text{ver}_K(m, \sigma) = 0, \quad \sigma \neq \text{sig}_K(m). \quad (2)
\]

Both functions \(\text{sig}_K, \text{ver}_K\) need to be computable in polynomial time, with function \(\text{ver}_K\) being public, while function \(\text{sig}_K\) is private.

1.1 The RSA Signature Scheme

Let, \(n = pq\) with \(p, q\) being large primes. Let also \(P = A = \mathbb{Z}_n\), and define,

\[
\mathcal{K} = \{(n, p, q, a, b) : n = pq, \ p, q \text{ prime, } ab \equiv 1 \pmod{\phi(n)}\}. \quad (3)
\]

Let \(n, b\) be the public key and \(p, q, a\) be the private key. Then we can define

\[
\text{sig}_K(m) = m^a \mod n \quad (4)
\]

and

\[
\text{ver}_K(m, \sigma) = \text{true} \iff m \equiv \sigma^b \pmod{n}. \quad (5)
\]

We saw that RSA encryption is secure as long as one cannot factor \(n\). Does that automatically translate to digital signatures also being secure?
1. Pick \( x = 1 \). Since \( 1^b \equiv 1 \mod n \) the signature for this message is \( \sigma = 1 \).

2. Choose a random signature \( \sigma \) and compute \( m = e_K(\sigma) \equiv \sigma^b \mod n \); Then \( y = sig_K(m) \) is a valid signature on message \( m \).

**Question:** What is the problem with the second attack. Is RSA insecure?

A way to eliminate the above problem is to make messages look random before they are signed. Possible application of a hash function to randomize the message to be signed.

**Combining encryption with signing:** Which one should come first?

1. Compute signature on message \( m \), attach signature to message and encrypt \( m || \sigma \).
2. Encrypt message \( m \) and produce \( y \). Sign \( y \) and attach signature \( \sigma \) on \( y \).

**Question:** Which method is better and why?

## 2 What does it Mean for a Signature Scheme to be Secure?

### 2.1 Adversarial Models

1. Key-only attack—only public key is known.
2. Known message attack—think of known plaintext.
3. Chosen message attack—think of chosen plaintext
4. Total break—recovery of the private key.
5. Selective forgery: The adversary is given a message \( m \) and is able to find a signature \( \sigma \) such that \( \text{ver}_K(m, \sigma) = \text{true} \).
6. Existential forgery: the adversary is able to find at least one valid \((m, \sigma)\) pair.

### 2.2 Security Analysis for RSA Scheme

We have already shown that the RSA scheme is not existentially secure since Eve can choose a random signature \( \sigma \) and find \( m \) such that \( \sigma \) is a valid signature for \( m \). This would be an existential forgery using a key only attack.

Existential forgery is also feasible utilizing a known message attack and the multiplicative property of the RSA cryptosystem. Let Eve obtain two message-signature pairs \((m_1, \sigma_1), (m_2, \sigma_2)\). Then Eve can generate a valid signature for message \( m_1 m_2 \mod n \) by computing \( \sigma_1 \sigma_2 \mod n \).

\[
\text{sig}_K(m_1 m_2) = (m_1 m_2)^a \mod n = m_1^a m_2^a \mod n \equiv \sigma_1 \sigma_2 \mod n. \tag{6}
\]

Selective forgery can be performed with a chosen plaintext attack as follows. Assume that Eve wants to compute the signature on \( m \). It is fairly easy to find \( m_1, m_2 \in \mathbb{Z}_n \) such that \( m \equiv m_1 m_2 \mod n \). Then it can ask Alice for her signatures on messages \( m_1, m_2 \) and then compute the signature for \( m \).
2.3 Combination of Signatures with Hashes

Idea is that each message \( m \) is first hashed using a publicly known hash function \( h \). Then the message digest \( h(m) \) is signed.

- **Existential forgery using a chosen message attack**
  Eve finds a collision, i.e., \( m, m' \) such that \( h(m) = h(m') \). Eve gives \( m \) to Alice and obtains \( \text{sig}_K(h(m)) \). Then \( (m', \text{sig}_K(h(m))) \) is a valid signed message.

- **Existential forgery using a known message attack**
  Eve obtains a pair \( (m, \sigma) \), where \( \sigma = \text{sig}_K(h(m)) \). Eve computes \( h(m) \) and tries to find \( m' \) such that \( h(m') = h(x) \).

- **Existential forgery using a key-only attack**
  Eve computes the signature on some message digest (remember RSA, where Eve picks signature and then finds \( m \) corresponding to the signature). Then given the message digest, it tries to invert the hash function and hence obtain a valid message signature pair.

3 ElGamal Based Signature Scheme

Recall that the public key is \((p, \alpha, \beta)\) where \( \beta = \alpha^q \). Alice’s private key is \( q \in \mathbb{Z}_p^* \), and \( \alpha \) is a primitive element of the group \( \mathbb{Z}_p^* \).

1. Choose a random number \( k \) such that \( 1 \leq k \leq p - 2 \), i.e., \( k \in \mathbb{Z}_{p-1}^* \).
2. \( \text{sig}_K(m, k) = (\gamma, \delta); \gamma = \alpha^k \mod p; \delta = (m - q\gamma)k^{-1} \mod (p - 1). \)
3. Alice sends \( (m, \text{sig}_K(m, k)) \) to Bob.
4. Bob verifies if: \( \beta^\gamma \gamma^\delta \equiv \alpha^m \mod p. \)

**Why does the verification work?**

During verification

\[ \beta^\gamma \gamma^\delta \equiv \alpha^{q\gamma} \alpha^{k\delta} \mod p \equiv \alpha^m \mod p, \]  

where we used that \( q\gamma + k\delta \equiv m \pmod{p-1} \). We can also find the signing algorithm constructively. Let \( \alpha^m \equiv \beta^\gamma \gamma^\delta \pmod{p} \). Then set \( \gamma = \alpha^k \pmod{p} \) and \( \beta = \alpha^q \). Then we obtain

\[ \alpha^m \equiv \alpha^{q\gamma} \alpha^{k\delta} \pmod{p}. \]  

Since \( \alpha \) is a primitive element modulo \( p \) the above congruence is true only if the exponents are congruent modulo \( p - 1 \). That is,

\[ m \equiv q\gamma + k\delta \pmod{p - 1}. \]  

Then we can solve for \( \delta \) and obtain \( \delta = (m - q\gamma)k^{-1} \pmod{p - 1}. \)

**Example of an ElGamal signature computation (ex. 7.1 in Stinson)**—Let \( p = 467, \alpha = 2, q = 127 \)

\[ \beta = \alpha^q \pmod{p} \]
\[ = 2^{127} \pmod{467} \]
\[ = 132. \]
Let Alice sign the message $m = 100$. She first chooses a random number $k \in \mathbb{Z}_{p-1}^*$. Let this number be $k = 213$. Then she computes
\[
\gamma = \alpha^k \mod p = 2^{213} \mod 467 = 29.
\] (13)
and
\[
\delta = (m - q\gamma)k^{-1} \mod (p - 1) = (100 - 2 \times 29)431 \mod 466 = 51.
\] (14)
The $\gamma, \delta$ is the signature for message $m$. Anyone can verify that
\[
\beta^\gamma \gamma^\delta \equiv \alpha^m \pmod{p}.
\] (15)