Lecture 7: Signatures

Chris Brzuska

October 24, 2022

The current lecture notes contain

- a discussion of applications of signature schemes
- the syntax of signature schemes
- security of signature schemes
- Lamport's theorem mentioned in the lecture which establishes that one-way functions imply one-time-secure signature scheme.

1 Applications

Signatures protect the authenticity and integrity of a message, i.e., if a message is signed, then I know that this message is authorized by the owner of the signing key. For example, trusted developers sign the code of the applications that they develop, and the operating system then verifies that an application is trusted (i.e., the signature verifies under the key of a trusted developer) before allowing the application to be installed. In this case, the use of a signature scheme is quite similar to an ordinary signature scheme, i.e., a person explicitly authorizes a piece of text as their own—except that a digital signature is verifiable by a machine, which is, of course, convenient. Due to this convenience, digital signatures are used, e.g., also in e-commerce, e-governance and other trusted applications. E.g., some universities digitally sign the diplomas which they issue.[1]

Moreover, signatures are an essential tool to prevent *person-in-the-middle* (PITM) attacks on key exchange protocols. Namely, a PITM attack on a key exchange protocol intends to trick to parties to use a key which the PITM attacker knows. To be able to mount such an attack, the PITM attacker needs to modify the messages which the two parties transmit, whence the usefulness of using signatures which prevent such changes. Strictly speaking, signatures, of course, do not *prevent* such changes, but rather, they only make such changes detectable.

An important issue w.r.t. to signing keys is the question how to link a signing key to a person or to a URL, since it is easy to verify that a signature is valid under a certain public-key, but how can I be sure that it is indeed the verification key of the intended URL and not an attacker's verification key? The way in which this is usually achieved is via *certificates*. A *certificate*, interestingly, again

[1] Note that in practice, this might not necessarily be extremely convenient, since it is not clear that the receiving university has the required technological know-how to verify the signature.

consists of a signature! Namely, it is a signature of a *name* and a *public-key* and it links these two together. However, this certificate could, potentially, also be adversary-generated. Thus, we might also want to see a certificate for the certificate issuer (and indeed, this is done in practice) which links the name of the issuer to the signing key. In this way, we can create so-called *certificate chains*. In the end, however, we need a root certificate. And such root certificates are typically pre-installed in a system. The security of the entire system hinges on the trustworthiness of the pre-installed root certificates.
After discussing these applications of signatures, let us now turn to the definition of their syntax and security.

2 Syntax and Security

Signature schemes are used for protect the authenticity and integrity of messages and are the *public-key* analogue of message authentication codes (MACs). Namely, while MACs can only be verified when knowing the secret MAC key, signatures can be verified when knowing the *public* verification key. I.e., a signature scheme relies on a *key pair*.

Definition 2.1 (Signature scheme)**.** A *signature* scheme s consists of three probabilistic polynomial-time (PPT) algorithms

$$\begin{aligned}(pk, sk) &\leftarrow\!\!\$ \mathsf{s.kgen}(1^\lambda)\\ \sigma &\leftarrow\!\!\$ \mathsf{s.sig}(sk, x)\\ \{0,1\} &\leftarrow\!\!\$ \mathsf{s.ver}(pk, x, \sigma)\end{aligned}$$

Correctness criterion: $\forall x \in \{0,1\}^* :$

$$\Pr_{(pk,sk)\leftarrow\!\!\$ \mathsf{s.kgen}(1^\lambda), \sigma\leftarrow\!\!\$ \mathsf{s}.sig(sk,x)}[\mathsf{s.ver}(pk, x, \sigma) = 1] = 1.$$

i.e. when generating a key pair $(pk, sk) \leftarrow\!\!\$\ \mathsf{s.kgen}(1^\lambda)$ and signing $\sigma \leftarrow\!\!\$\ \mathsf{sig}(sk, x)$ for some message x, then $\mathsf{ver}(pk, x, \sigma) = 1$.

Definition 2.2 (UNF-CMA for Signatures)**.** Let the games $\texttt{Gunf-cma}_{\mathsf{s}}^0$ and $\texttt{Gunf-cma}_{\mathsf{s}}^1$ be defined as

$\texttt{Gunf-cma}_{\mathsf{s}}^0$	$\texttt{Gunf-cma}_{\mathsf{s}}^1$
Package Parameters	Package Parameters
λ: security parameter	λ: security parameter
s: signature scheme	s: signature scheme
Package State	Package State
pk: public key	pk: public key
sk: secret key	sk: secret key
	$\mathcal{L}$: list
GETPK()	GETPK()
if $pk = \bot$:	**if** $pk = \bot$:
$(pk, sk) \leftarrow\!\$\ \mathsf{s.kgen}(1^\lambda)$	$(pk, sk) \leftarrow\!\$\ \mathsf{s.kgen}(1^\lambda)$
return pk	**return** pk
SIG(x)	SIG(x)
if $pk = \bot$:	**if** $pk = \bot$:
$(pk, sk) \leftarrow\!\$\ \mathsf{s.kgen}(1^\lambda)$	$(pk, sk) \leftarrow\!\$\ \mathsf{s.kgen}(1^\lambda)$
$\sigma \leftarrow\!\$\ \mathsf{s.sig}(sk, x)$	$\sigma \leftarrow\!\$\ \mathsf{s.sig}(sk, x)$
	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, \sigma)\}$
return σ	**return** σ
VERIFY(x, σ)	VERIFY(x, σ)
if $pk = \bot$:	**if** $(x, \sigma) \in \mathcal{L}$:
$(pk, sk) \leftarrow\!\$\ \mathsf{s.kgen}(1^\lambda)$	**return** 1
$d \leftarrow \mathsf{s.ver}(pk, x, \sigma)$	
return d	**return** 0

A signature scheme s is UNF-CMA-secure if the real game $\texttt{Gunf-cma}_{\mathsf{s}}^0$ and the ideal game $\texttt{Gunf-cma}_{\mathsf{s}}^1$ are computationally indistinguishable, that is, for all PPT adversaries $\mathcal{A}$, the advantage

$$\begin{aligned}&\mathbf{Adv}_{\mathcal{A}}^{\texttt{Gunf-cma}_{\mathsf{s}}^0, \texttt{Gunf-cma}_{\mathsf{s}}^1}(\lambda)\\ :=&\left|\Pr\left[1 = \mathcal{A} \to \texttt{Gunf-cma}_{\mathsf{s}}^0\right] - \Pr\left[1 = \mathcal{A} \to \texttt{Gunf-cma}_{\mathsf{s}}^1\right]\right|\end{aligned}$$

is negligible in λ.

Remark. Note that we denote $\texttt{Gunf-cma}_{\mathsf{s}}^b$ the unforgeability game for signatures when s is a *signature* scheme and $\texttt{Gunf-cma}_{m}^b$ the unforgeability game for MACs when m is a MAC scheme.

$s_f.\mathsf{kgen}(1^\lambda)$

for $i = 1..\lambda$
  $x_0^i \leftarrow\$ \{0,1\}^\lambda$
  $x_1^i \leftarrow\$ \{0,1\}^\lambda$
  $y_0^i \leftarrow f(x_0^i)$
  $y_0^i \leftarrow f(x_1^i)$
$sk \leftarrow \begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$
$pk \leftarrow \begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$
return (sk,pk)

$s_f.\mathsf{sig}(sk, m)$

parse sk as
$\begin{pmatrix} x_0^1, .., x_0^\lambda \\ x_1^1, .., x_1^\lambda \end{pmatrix}$
for $i = 1..\lambda$
  $z^i \leftarrow x^i_{m[i]}$
  // m[i] is ith bit of m
$\sigma \leftarrow (z^1, .., z^\lambda)$
return σ

$s_f.\mathsf{ver}(pk, m, \sigma)$

parse pk as
$\begin{pmatrix} y_0^1, .., y_0^\lambda \\ y_1^1, .., y_1^\lambda \end{pmatrix}$
$(z^1, .., z^\lambda) \leftarrow \sigma$
for $i = 1..\lambda$
  if $f(z^i) \neq y^i_{m[i]}$:
    return 0
return 1

Figure 1: Lamport's one-time signature scheme s_f for messages of length λ.

3 Constructions

In the lecture, we constructed the above signature scheme s_f based on an injective one-way function f, which signs messages of exactly length λ.
We then stated the following theorem (The proof is one of the exercises on Exercise Sheet 6):

Theorem 1 (Signature scheme from OWF)**.** If f is a one-way function, then the signature scheme s_f is a one-time secure signature scheme for messages of length λ, i.e., for all PPT adversaries $\mathcal{A}$ that make only a single SIG query of length ℓ, the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathtt{Gpunf\text{-}cma}_s^0, \mathtt{Gpunf\text{-}cma}_s^1}(\lambda)$ is negligible.