CS 120/ E-177: Introduction to Cryptography

Salil Vadhan and Alon Rosen Nov. 22, 2006

Lecture Notes 17:

Digital Signatures

Recommended Reading.

- Katz-Lindell 10

1 Signatures vs. MACs

Digital signatures are the public-key version of message authentication codes:*anybody* can verify. Can be thought of as digital "analogue" of handwritten signatures (but are in fact stronger). Unlike MACs signatures are:

1. *Publicly verifiable* - anybody can verify their validity.
2. *Transferable* - recipient can show the signature to another party who can then verify that the signature is valid (this follows from public verifiability).
3. *Non-repudiable* - If Alice digitally signs a document, then Bob can prove to a third party (e.g. a court) that she signed it, by presenting the document and her signature. By definition, only Alice could have produced a valid signature.

   Notice that MACs *cannot* have this property. None of the parties holding the key can claim the other one has signed. This is because it might be the case that the other party has actually signed.

MACs are more efficient in practice.

2 Syntax

Definition 1 *A* digital signature scheme *consists of three algorithms* (G, S, V) *such that:*

- *The* key generation *algorithm* G *is a randomized algorithm that returns a* public key PK *and a* secret key SK*; we write* $(PK, SK) \stackrel{R}{\leftarrow} G(1^n)$.
- *The* signing *algorithm* S *is a (possibly) randomized algorithm that takes the secret key* SK *and a* message m *and outputs a signature* σ*; we write* $\sigma \stackrel{R}{\leftarrow} \mathrm{S}_{SK}(m)$.
- *The* verification *algorithm* V *is a deterministic algorithm that takes the public key* PK*, a message* m*, and a signature* σ*, and outputs* $V_{PK}(m, \sigma) \in \{\texttt{accept}, \texttt{reject}\}$.

We require $V_{PK}(m, \mathrm{S}_{SK}(m)) = \texttt{accept}$ *for all* $(PK, SK) \stackrel{R}{\leftarrow} G(1^n)$ *and* $m \in \{0,1\}^*$.

2.1 Comments

1. The *sender* needs secret key, *opposite* from public-key encryption. Alice will send a message encrypted with Bob's public key but signed with Alice's secret key. However, digital signatures and public key encryption are *not* "duals" of each other (as one might be tempted to think).

2. It is conceivable that the sender keeps state between signatures and we will allow this in some cases.

3. Similarly to MAC, randomization is not necessary.

4. Note that we do not require any formatting on the messages and they could be arbitrary strings. Sometimes it is required that messages obey some pre-specified "format" (possibly depending on PK). In such a case, it is required to explicitly specify how to map arbitrary strings into a string that obeys this format.

3 Security

Definition 2 (existential unforgeability under adaptive chosen message attack) *A signature scheme* (G, S, V) *is* secure *if for every PPT* A*, there is a negligible function* ε *such that*

$$\Pr\left[A^{\mathrm{S}_{SK}(\cdot)}(PK) \text{ forges}\right] \leq \varepsilon(k) \qquad \forall k,$$

where the probability is taken over $(PK, SK) \stackrel{R}{\leftarrow} G(1^k)$ *and the coin tosses of* A*. "*A *forges"* $\equiv A$ *produces a pair* (m, σ) *for which (a)* $V_{PK}(m, \sigma) =$ `accept`*, and (b)* m *is different from all of* A*'s queries to the* S_{SK}*-oracle.*

3.1 Comments

1. Definition is strong:

   (a) A gets access to signatures on messages of its choice.

   (b) A forges even if m it has produced is "meaningless."

   These are indeed strong requirements. However, if we can satisfy them, we can certainly satisfy weaker requirements. Also, this will give us signature schemes which are *application independent* (in particular, will be suitable for use regardless of the formatting/semantics of the messages being signed). As for Item (1), in practice this can happen. For example, notary would conceivably sign on any document regardless of its contents.

4 Applications

Here are some applications of signatures.

1. Can be used for public-key infrastructure (without public directory):

   - One trusted party (*certificate authority*, e.g. Verisign) has a public key known to everyone, and signs individual's public keys, e.g. signs statement like

$$m = \texttt{`Verisign certifies that } PK_{\text{Alice}} \texttt{ is Alice's public key'}$$

When starting a communication with a new party, Alice sends the *certificate*

$$(PK_{\text{Alice}}, m, \mathrm{S}_{SK_{\text{Verisign}}}(m))$$

- Many variants: Can be done hierarchically (Verisign signs Harvard's PK, Harvard signs individual students' public keys).

2. Can be used by any trusted organization/individual to certify any property of a document/file, e.g. that a piece of software is safe.

3. Useful for obtaining chosen ciphertext security (for private key encryption MACs are used).

5 Examples

Here are some examples of insecure signatures.

1. Let f be a *one-way function*.

   (a) $y = f(x)$ for $x \xleftarrow{r} \{0,1\}^n$. $SK = x, PK = y$.

   (b) $\mathrm{S}_{SK}(m) = x$.

   (c) $V_{PK}(m, \sigma) = 1$ if and only if $f(\sigma) = y$.

   Analog of handwritten signature. Verifies that signer is able to sign.

   Not secure. Can "cut and paste". Signature is independent of message. Adversary A can simply take σ an append it to message of its choice. Demonstrates why our definition of security is stronger than handwritten signatures.

2. Let $\mathcal{F} = \{f_i : D_i \to D_i\}_{i \in \mathcal{I}}$ be a family of *trapdoor permutations*.

   (a) $PK = i$, $SK = t$

   (b) $\mathrm{S}_{SK}(m) = f_i^{-1}(m)$.

   (c) $V_{PK}(m, \sigma)$: check that $f_i(\sigma) = m$.

   Not secure. f_i might be easy to invert on a particular m, e.g. for RSA, if $m = 1$ then $\mathrm{S}_{N,d}(1) = 1$. More generally, the adversary can choose $\sigma \in D_i$, compute $m = f_i(\sigma)$ and send the forgery (m, σ).

3. A special case of the trapdoor permutation example is "textbook RSA"

   (a) $PK = (N, d)$, $SK = (N, e)$

   (b) $\mathrm{S}_{SK}(m) = m^e \bmod N$.

   (c) $V_{PK}(m, \sigma) = 1$ if and only if $\sigma^d = m \bmod N$.

   Not secure. Given any message m and $PK = (N, d)$ can ask for signatures σ_1, σ_2 on messages m_1 and $m_2 = m_1^{-1} \cdot m$, where $m_1 \xleftarrow{r} Z_N^*$. Then $\sigma_1 \cdot \sigma_2 \bmod N$ is a valid signature on m (why?). In addition m is very likely to be different than m_1, m_2.

   Demonstrates how one can attack signature scheme even without retrieving SK.

6 Constructions

Digital signatures can be constructed from one-way functions (no trapdoors!). Signing is deterministic. This result is beyond the scope of this class. Instead we will outline a provably secure construction from collision-free hash functions:

1. *One-time* signature for fixed-length messages ($\mathcal{P}_k = \{0,1\}^k$) from any one-way function.
2. Use *hash-then-sign* to convert this into a one-time signature for unbounded-length messages.
3. Use *key refreshing* to convert one-time signature into many-use signature.

6.1 One-time signature

We start by showing how to construct a *one-time* signature from any one-way function f.

Single bit message. Let f be any one-way function.

1. The secret key is $SK = \{x_0, x_1\}$ where x_0 and x_1 are chosen at random in $\{0,1\}^k$. The public key is $PK = \{y_0, y_1\}$ where $y_0 = f(x_0)$ and $y_1 = f(x_1)$.
2. The signature of the bit b is $\mathrm{S}_{SK}(b) = x_b$.
3. The verification is $V_{PK}(b, x) = \texttt{accept}$ iff $f(x) = y_b$.

This scheme is not very efficient and gives away half of the secret key. But it is secure if the adversary asks only one query. Assume that he obtains the signature of 0, i.e. x_0. Then the adversary cannot produce the signature of 1, i.e. invert y_1, because x_0 and x_1 are chosen independently so seeing x_0 does not help him to invert y_1.

ℓ-bit messages.

- $G(1^k)$: For $i = 1, \cdots, \ell$, choose $x_{i,0} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^k$, $x_{i,1} \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^k$. Let $y_{i,0} = f(x_{i,0})$, $y_{i,1} = f(x_{i,1})$ PK=all y's, SK=all x's.
- $\mathrm{S}_{SK}(m)$ for $m \in \{0,1\}^\ell$: For $i = 1, \ldots, \ell$, let $\sigma_i = x_{i,m_i}$. Output $\sigma = (\sigma_1, \ldots, \sigma_\ell)$.
- $V_{PK}(m, \sigma)$: Check that $f(\sigma_i) = y_{i,m_i}$ for all $in\{1, \cdots, \ell\}$.

Theorem 3 *Above is a secure one-time signature (i.e. adversary can make only one query) for message space $\{0,1\}^\ell$ if f is a one-way function.*

Proof: Reducibility argument. Suppose there is a PPT forger A that succeeds with nonnegligible probability ε. We will build PPT B that inverts f with nonnegligible probability.

Intuitively, the query (m, σ) reveals half of the x's. But $m \neq m'$ so in at least one position, A has to guess the corresponding x, i.e. invert f on a point y. B will guess the position where A has successfully inverted f.

Inverter $B(y)$:

1. Choose $j \stackrel{R}{\leftarrow} \{1, \ldots, \ell\}, c \stackrel{R}{\leftarrow} \{0,1\}$. Let $y_{j,c} = y$.
2. For all $(i,b) \neq (j,c)$, choose $x_{i,b} \stackrel{R}{\leftarrow} \{0,1\}^k$, let $y_{i,b} = f(x_{i,b})$.
3. Let PK = all $y_{i,b}$'s, and run $A(PK)$.
4. A asks for signature on some m. If $m_j \neq c$, B can answer correctly A's query, otherwise output `fail`.
5. A produces forgery (m', σ'). If $m'_j = c$, output $x = \sigma'_j$. Otherwise output `fail`.

Whenever A would successfully forge, B will succeed w.p. $\geq 1/2\ell$ (whenever (j,c) satisfy $m_j \neq c$ and $m'_j = c$, σ'_j is an inverse of y under f) Thus B succeeds w.p. $\varepsilon/2\ell$, still nonnegligible. ■

If the message is of length ℓ, the public key is of length $2k\ell$ so this scheme is not very efficient. Also, to construct signatures for multiple messages we will need that the size of PK is shorter than the size of message.

How to sign long messages? Sign a short *hash* of the message. This is called *hash-then-sign*. See lecture notes on collision-free hashing.

6.2 One-time signatures to many-time signatures

How do we obtain a signature scheme for multiple messages?

- Idea: generate new keys "on the fly" and authenticate them with previous key.
- Start with a one-time signature scheme (G, S, V), and construct general signature scheme as follows:
  - Public key PK_0, secret key SK_0 for original scheme.
  - To sign first message m_1: Generate $(PK_1, SK_1) \stackrel{R}{\leftarrow} G(1^k)$. Let $\sigma_1 \stackrel{R}{\leftarrow} \mathrm{S}_{SK_0}(m_1, PK_1)$. Output $\sigma'_1 = (1, \sigma_1, m_1, PK_1)$.
  - To sign second message m_2: Generate $(PK_2, SK_2) \stackrel{R}{\leftarrow} G(1^k)$. Let $\sigma_2 \stackrel{R}{\leftarrow} \mathrm{S}_{SK_1}(m_2, PK_2)$. Output $\sigma'_2 = (2, \sigma'_1, \sigma_2, m_2, PK_2)$.
  - etc.
- This works, but very inefficient.
- Improvements:
  - Use a tree — each key authenticates two new keys $\Rightarrow$ signature length, verification time, signing time no longer grow linearly w/ number of signatures.
  - Can also make it stateless.
  - Can construct secure signatures based on any one-way function (very complicated!).
  - See Katz-Lindell.

7 Digital Signatures in Practice

- Hash-then-sign with plain RSA
  - $PK = (N, e)$, $SK = (N, d)$ s.t. $ed \equiv 1 \pmod{\phi(N)}$.
  - $\mathrm{S}_{SK}(m) = (H(m))^d \bmod N$ where H is SHA-1 or MD5.
  - Intuition: adversary has "no control" over $H(m)$, so cannot forge by choosing signature first, or by exploiting special inputs on which RSA is easy to invert.
  - RSA PKCS #1: $H(m) = 001FF \cdots FF00 \circ \text{SHA-1}(m)$.
    - No justification.
    - Doesn't seem related to one-wayness of RSA since $H(m)$ is always of very special form.
  - Random Oracle Model
    - If model H as a "public" truly random function $H : \{0,1\}^* \to \mathbb{Z}_N^*$, then can justify signature based on one-wayness of RSA.
    - But no H is a truly random function. Heuristically, it is hoped that some cryptographic hash functions (e.g. SHA-1) "behave like" a truly random function.
- El Gamal signatures
  - $PK = (p, g, \hat{x})$, $SK = (p, g, x)$ where $\hat{x} = g^x \bmod p$.
  - $\mathrm{S}_{SK}(m)$:
    1. Choose $y \xleftarrow{\mathrm{R}} \mathbb{Z}_{p-1}$, let $\hat{y} = g^y \bmod p$.
    2. Find s s.t. $g^m \equiv g^{x\hat{y}} \cdot \hat{y}^s \pmod{p}$.
    3. Output $\sigma = (\hat{y}, s)$.

    How does the signer compute s? We have $m \equiv x\hat{y} + ys \pmod{(p-1)}$ and the signer knows $X, y, \hat{y}, m$ so he can compute $(m - x\hat{y})y^{-1}$ modulo $(p-1)$.
  - $V_{PK}(m, (y, s))$: Check that $g^m \equiv \hat{x}^y \cdot y^s \pmod{p}$.
  - Not secure! (Similar attacks as plain trapdoor functions.)
  - Digital Signature Standard (1991, NIST+NSA): hash-then-sign, using SHA-1 and DSA (variant of El Gamal).
- Bridging the gap...
  - Heuristic proofs in Random Oracle Model (as mentioned above).,
  - More efficient, truly provable signatures based on specific hard problems: Strong RSA (given N, z, find $y, e > 1$ s.t. $y^e \equiv z \pmod{N}$).