CS 127/CSCI E-127: Introduction to Cryptography

Prof. Salil Vadhan — Fall 2013

Lecture Notes 17:

MAC Domain Extension & Digital Signatures

Reading.

- Katz-Lindell §4.3–4.4 (2nd ed) and §12.0-12.3 (1st ed).

1 Domain Extension for MACs

Last time we saw a MAC construction for message space $\{0,1\}^\ell$, where $\ell = \ell(n)$ is the input length of a PRF family. What about long messages? Suppose that we have a MAC secure for messages of length $\ell(n) = n$. How can we construct a secure MAC for longer messages? Given $m = m_1 \| m_2 \| \cdots \| m_d$, where $|m_i| = n$, consider the following constructions:

- Attempt 1: Define $\mathsf{Mac}'_k(m) = \mathsf{Mac}_k(m_1 \oplus m_2 \oplus \ldots \oplus m_d)$.

- Attempt 2: Define $\mathsf{Mac}'_k(m) = \mathsf{Mac}_k(m_1), \cdots, \mathsf{Mac}_k(m_d)$.

- Attempt 3: For, say $|m_i| = n/2$, define $\mathsf{Mac}'_k(m) = \mathsf{Mac}_k(m_1, 1), \cdots, \mathsf{Mac}_k(m_d, d)$ (here we assume that an index i is encoded by $n/2$ bits).

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a MAC for message space $\{0,1\}^n$. Create a MAC for all messages of length $\{0,1\}^{d \cdot (n/3)}$ for $d = d(n) = \mathrm{poly}(n)$ by defining $\mathsf{Mac}'_k(m)$ as follows:

- Write $m = m_1 \| m_2 \| \cdots \| m_d$, where each $m_i \in \{0,1\}^{n/3}$.
- Choose $r \xleftarrow{R} \{0,1\}^{n/3}$ (a random "identifier")
- For $i = 1, \ldots, d$, compute $t_i \xleftarrow{R} \mathsf{Mac}_k(m_i \| i \| r)$, where i is padded to $n/3$ bits. (Note that $\|i\| = O(\log n) \leq n/3$ for sufficiently large n.)
- Output $(r, t_1, t_2, \cdots, t_d)$.

$\mathsf{Vrfy}'_k(m_1\|\cdots\|m_d,(r,t_1,t_2,\cdots,t_d))$ accepts if for all $i\in\{1,\cdots,d\}$, $\mathsf{Vrfy}_k(m_i\|i\|r,t_i)=\texttt{accept}$.

Theorem 1 *If* $(\mathsf{Gen},\mathsf{Mac},\mathsf{Vrfy})$ *is secure for message space* $\{0,1\}^n$*, then* $(\mathsf{Gen}',\mathsf{Mac}',\mathsf{Vrfy}')$ *is secure for message space* $\{0,1\}^{d\cdot n/3}$.

Proof Sketch: Suppose there were a forger $\mathcal{A}'$ for $(\mathsf{Gen}',\mathsf{Mac}',\mathsf{Vrfy}')$ with nonnegligible success probability ε. After obtaining MACs of polynomially many messages, $\mathcal{A}'$ produces a forgery (m,t). If this forgery is successful, then $m=m_1,\cdots,m_d$, $t=(r,t_1,\cdots,t_d)$, and for all i, $\mathsf{Vrfy}_k(m_i\|i\|r,t_i)=\texttt{accept}$. At a high level, the analysis can be divided into cases:

1. The forgery $t=(r,t_1,\cdots,t_d)$ contains a "new" r (i.e. one that has not appeared in the previous MACs $\mathcal{A}'$ has seen).

2. The forgery $t=(r,t_1,\cdots,t_d)$ does *not* contain a "new" r. Here we have two sub-cases:

   (a) r has appeared in only one MAC $m'=m'_1,\cdots,m'_d$, $t'=(r,t'_1\cdots t'_d)$ that $\mathcal{A}'$ has seen.

   (b) r has appeared more than once.

□

Note: This construction is *not* secure for message space $\bigcup_d(\{0,1\}^n)^d$, i.e. for messages with a variable number of blocks. (Why not?) How can we make it secure?

CBC MAC : The previous construction involves d applications of $(\mathsf{Gen},\mathsf{Mac},\mathsf{Vrfy})$ and the size of the resulting tag is dn. For practical purposes it would be desirable to have a shorter tag. One example for a more efficient MAC is the CBC MAC, which we describe next.

Let $\mathcal{F}=\{f_k:\{0,1\}^n\to\{0,1\}^n\}$ be a family of pseudorandom functions (or pseudorandom permutations). Define a MAC over message space $\{0,1\}^{d\cdot n}=(\{0,1\}^n)^d$ by defining $\mathsf{Mac}_k(m_1,\cdots,m_d)=y_d$, where $y_i=f_k(m_i\oplus y_{i-1})$ and $y_0=0^n$.

Theorem 2 *CBC MAC is secure for message space* $\{0,1\}^{d\cdot n}$.

- Also needs a modification to be secure for variable-length message spaces.
- DES CBC MAC used extensively in practice.
- There are other ways to convert fixed-length MACs into arbitrary-length MACs, e.g. "hash-then-MAC" which we'll see next time.

2 Digital Signatures

2.1 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.

2.2 Syntax

Definition 3 *A* digital signature scheme *consists of three algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *such that:*

- *The* key generation *algorithm* Gen *is a randomized algorithm that returns a* public key pk *and a* secret key sk*; we write* $(pk, sk) \xleftarrow{R} \mathsf{Gen}(1^n)$.
- *The* signing *algorithm* Sign *is a (possibly) randomized algorithm that takes the secret key* sk *and a* message m *and outputs a signature* σ*; we write* $\sigma \xleftarrow{R} \mathsf{Sign}_{sk}(m)$.
- *The* verification *algorithm* Vrfy *is a deterministic algorithm that takes the public key* pk*, a message* m*, and a signature* σ*, and outputs* $\mathsf{Vrfy}_{pk}(m, \sigma) \in \{\texttt{accept}, \texttt{reject}\}$.

We require $\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m)) = \texttt{accept}$ *for all* $(pk, sk) \xleftarrow{R} \mathsf{Gen}(1^n)$ *and* $m \in \{0,1\}^*$.

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.

2.3 Security

Definition 4 (existential unforgeability under adaptive chosen message attack) *A signature scheme* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* secure *if for every PPT A, there is a negligible function* ε *such that*

$$\Pr\left[A^{\mathsf{Sign}_{sk}(\cdot)}(pk) \text{ forges}\right] \leq \varepsilon(k) \qquad \forall k,$$

where the probability is taken over $(pk, sk) \xleftarrow{R} \mathsf{Gen}(1^k)$ *and the coin tosses of A. "A forges"* $\equiv$ *A produces a pair* (m, σ) *for which (a)* $\mathsf{Vrfy}_{pk}(m, \sigma) = \texttt{accept}$*, and (b) m is different from all of A's queries to the* Sign_{sk}*-oracle.*

Comments This definition is strong:

1. A gets access to signatures on messages of its choice.
2. 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.

2.4 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*

     $$(pk_{\text{Alice}}, m, \mathsf{Sign}_{sk_{\text{Verisign}}}(m))$$

   - 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).

2.5 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) $\mathsf{Sign}_{sk}(m) = x$.

(c) $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$ if and only if $f(\sigma) = y$.

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

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

   (a) $pk = i$, $sk = t$

   (b) $\mathsf{Sign}_{sk}(m) = f_i^{-1}(m)$.

   (c) $\mathsf{Vrfy}_{pk}(m, \sigma)$: check that $f_i(\sigma) = m$.

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

   (a) $pk = (N, d)$, $sk = (N, e)$

   (b) $\mathsf{Sign}_{sk}(m) = m^e \bmod N$.

   (c) $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$ if and only if $\sigma^d = m \bmod N$.