Katz, Lindell
Introduction to Modern Cryptrography
Slides Chapter 4

Markus Bläser, Saarland University

Message authentication

How can you be sure that a message has not been modified?

Encyrption is not enough ...

- Recall our prg-based scheme: $c := m \oplus G(k)$
- We can modify the message sent: $c' = c \oplus s$
- Decryption of c' yields $m \oplus s$

Message authentication codes (MAC)

Definition (4.1)

A triple $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ of ppt algorithms is a *MAC* if

1. Gen on input 1^n outputs a key k with $|k| \geq n$.
2. The tag-generation algorithm Mac takes a key k and a message m and produces a tag $t \leftarrow \mathsf{Mac}_k(m)$.
3. The deterministic verifier Vrfy takes a key k, a message m, and a tag t and outputs a bit $b := \mathsf{Vrfy}_k(m, t)$. ($b = 1$ means valid, $b = 0$ means invalid.)

For all keys k and messages m: $\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = 1$.
If Mac is only defined on messages of length $\ell(n)$, then this scheme is called *fixed length MAC*.

- Gen usually chooses key uniformly at random
- Canonical verification for deterministic Mac: Compute $\tilde{t} := \mathsf{Mac}_k(m)$ and compare with t.

The message authentication experiment

$\text{Mac-Forge}_{\mathcal{A},\Pi}(n)$:

1. k is generated by $\text{Gen}(1^n)$
2. The adversary $\mathcal{A}$ is given 1^n and oracle access to $\text{Mac}_k(.)$.
   $\mathcal{A}$ eventually outputs (m, t)
   Let $\mathcal{Q}$ denote the set of queries made by $\mathcal{A}$.
3. $\mathcal{A}$ succeeds if
   3.1 $\text{Vrfy}_k(m, t) = 1$ and
   3.2 $m \notin \mathcal{Q}$

$\mathcal{A}$ has to produce one valid nontrivial pair (m, t)!!

Secure message authentication

Definition (4.2)

$\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is *existentially unforgeable under an adaptive chosen message attack* or just *secure* if for all ppt $\mathcal{A}$

$$\Pr[\text{Mac-Forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

- Does not prevent replay attacks, i.e., sending (m, t) several times.
- Solution: counters/time stamps

Strongly secure MAC

Secure MACs do not prevent $\mathcal{A}$ to create a new tag $t' \neq \mathrm{Mac}_k(m)$ for some message $m \in \mathcal{Q}$.

Modified experiment Mac-sForge$_{\mathcal{A},\Pi}$

- Record pairs $(m, t) \in \mathcal{Q}$.
- $\mathcal{A}$ has to output a pair $(m, t) \notin \mathcal{Q}$.

$\Longrightarrow$ *strongly secure MAC* (Definition 4.3)

Proposition (4.4)

If Π is a secure MAC using canonical verification, then Π is strongly secure.

A timing attack (a la MacGyver)

- To verify a tag t using canonical verification, $\mathrm{t}' := \mathrm{Mac}_\mathrm{k}(\mathrm{m})$ is computed and compared to t.
- If character-wise comparision is used, rejection time depends on position of first unequal letter.
- Assume you know a prefix s of t.
- Then send $(\mathrm{m}, s||\mathrm{a}), \ldots, (\mathrm{m}, s||\mathrm{z})$ one after another (padded to the right length).
- One message will take longer, this is the correct next character

(Of course, computers use bytes instead of letters.)

A fixed length MAC

Construction (4.5)

Let F *be a prf.*

- Mac*: On key* $\mathrm{k} \in \{0,1\}^n$ *and* $\mathrm{m} \in \{0,1\}^n$*, output* $\mathrm{t} := \mathrm{F}_k(\mathrm{m})$*.*
- Vrfy*: On key* k*, message* m*, and tag* t*, output* 1 *if* $\mathrm{t} = \mathrm{F}_k(\mathrm{m})$*. Otherwise, output* 0*. (If* $|\mathrm{k}| \neq |\mathrm{m}|$*, then output* 0*.)*

Theorem (4.6)

If F *is a prf, then Construction 4.5 is a secure fixed-length MAC.*

Domain extension

Construction (4.7)

Let $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ *be a fixed-length.*

- Mac*: on key* $k \in \{0,1\}^n$ *and message* $m \in \{0,1\}^*$ *of length* $\ell < 2^{n/4}$*, divide* m *into blocks* $m_1, \ldots, m_d$ *of length* $n/4$*. (Pad the last block.) Choose* $r \in \{0,1\}^{n/4}$ *uniformly at random.*
  Let $t_i := \mathsf{Mac}'_k(r \| \ell \| i \| m_i)$, $i = 1, \ldots, d$*. Output* $\langle r, t_1, \ldots, t_d \rangle$*.*
- Vrfy*: On input* k*, message* m *of length* $< 2^{n/4}$*, and tag* $\langle r, t_1, \ldots, t_{d'} \rangle$*, divide* m *into messages* $m_1, \ldots, m_d$ *of length* $n/4$*. Output* 1 *if* $d = d'$ *and* $\mathsf{Vrfy}_k(r \| \ell \| i \| m_i, t_i) = 1$ *for all* $1 \leq i \leq d$*. Otherwise, output* 0*.*

Domain extension (2)

Theorem (4.8)

If Π' is a secure fixed-length MAC, then Construction 4.7 is a secure MAC for arbitrary length messages.

- Construction 4.7 is not practical, since the tag is four times as long as the message.
- In practice, CBC-MAC can be used (see Section 4.4).

Chosen-ciphertext attacks

The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(n)$:

1. A key is generated using $\mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given 1^n and oracle access to $\mathsf{Enc}_k(.)$ and $\mathsf{Dec}_k(.)$. It outputs a pair of messages m_0 and m_1 of the same length.
3. $b \in \{0,1\}$ is chosen uniformly at random and $c \leftarrow \mathsf{Enc}_k(m_b)$ is given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_k(.)$ and $\mathsf{Dec}_k(.)$ but is not allowed to query $\mathsf{Dec}_k(.)$ on c. Eventually, $\mathcal{A}$ outputs a bit b'.
5. The result of the experiment is 1 if $b' = b$ and 0 otherwise.

Adversary can decrypt any message except the challenge message.

CCA-security

Definition (3.33)

A scheme Π is *CCA-secure* if for all ppt $\mathcal{A}$,

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n).$$

- (without proof) CCA-secure implies CCA-secure for multiple encryptions
- implies "non-malleability": small modifications on the ciphertext result in ciphertexts that have "no relation" to the message.

Authenticated Encryption

The unforgeable encryption experiment $\text{Enc-Forge}_{\Pi,\mathcal{A}}(n)$:

1. A key k is generated using $\text{Gen}(1^n)$
2. $\mathcal{A}$ is given 1^n and oracle access to $\text{Enc}_k(.)$. $\mathcal{A}$ outputs a ciphertext c.
3. Let $m := \text{Dec}_k(c)$ and let $\mathcal{Q}$ denote the set of all queries asked by $\mathcal{A}$.
   The output of the experiment is 1 if
   3.1 $m \neq \perp$ (c is a valid ciphertext)
   3.2 $m \notin \mathcal{Q}$

Message integrity for encryption schemes: $\mathcal{A}$ cannot produce a nontrivial valid ciphertext.

Authentcated Encryption (2)

Definition (4.16)

A private-key encryption scheme Π is *unforgeable* if for all ppt adversaries $\mathcal{A}$,

$$\Pr[\text{Enc-Forge}_{\Pi,\mathcal{A}}(n) = 1] \leq \text{negl}(n).$$

Definition (4.17)

A private-key encryption scheme Π is an *authenticated encryption scheme* if it is CCA-secure and unforgeable.

Generic constructions

- $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ CPA-secure encryption scheme
- $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ secure MAC
- Choose two independent (!) keys k_E and k_M.

Encrypt-and-authenticate Send $\langle c, t \rangle$ where
$$c \leftarrow \mathsf{Enc}_{k_E}(m) \text{ and } t \leftarrow \mathsf{Mac}_{k_M}(m)$$

Authenticate-then-encrypt Send c computed by
$$t \leftarrow \mathsf{Mac}_{k_M}(m) \text{ and } c \leftarrow \mathsf{Enc}_{k_E}(m \| t).$$

Encrypt-then-authenticate Senc $\langle c, t \rangle$ where
$$c \leftarrow \mathsf{Enc}_{k_E}(m) \text{ and } t \leftarrow \mathsf{Mac}_{k_M}(c).$$

First two approaches fail.

Encrypt-then-authenticate

Construction (4.18)

Let

- $\Pi_E = (\mathsf{Enc}, \mathsf{Dec})$ *encryption scheme*
- $\Pi_M = (\mathsf{Mac}, \mathsf{Vrfy})$ *MAC*

Define a private-key encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$*:*

- Gen'*: on input* 1^n*, choose* $k_E, k_M \in \{0,1\}^n$ *independently and uniformly at random.*
- Enc'*: on input* k_E, k_M *and message* m*, compute* $c \leftarrow \mathsf{Enc}_{k_E}(m)$ *and* $t \leftarrow \mathsf{Mac}_{k_M}(c)$*. Return* $\langle c, t \rangle$*.*
- Dec'*: on input* k_E, k_M *and* $\langle c, t \rangle$*, first check whether* $\mathsf{Vrfy}_{k_M}(c, t) = 1$*. If yes, then output* $\mathsf{Dec}_{k_E}(c)$*. Otherwise, output* $\perp$*.*

Encrypt-then-authenticate (2)

Theorem (4.19)

If Π_E is CPA-secure and Π_M is a strongly secure MAC, then Construction 4.18 is a authenticated encryption scheme.

Proof

Adversary $\mathcal{A}_M$ (oracle access to $\mathrm{Mac}_{k_M}(.)$)

1. Choose $k_E \in \{0,1\}^n$ and $i \in \{1,\ldots,q(n)\}$ uniformly at random.
2. Run $\mathcal{A}$ on 1^n.

   When $\mathcal{A}$ makes an encryption oracle query for a message m:

   2.1 Compute $c \leftarrow \mathrm{Enc}_{k_E}(m)$

   2.2 Compute a tag t for c using $\mathrm{Mac}_{k_M}(.)$ and give $\langle c, t\rangle$ to $\mathcal{A}$.

   When $\mathcal{A}$ produces messages m_0, m_1, choose b uniformly at random and encrypt m_b as above as the challenge.

   When $\mathcal{A}$ makes a decryption oracle query, do the following:

   2.1 If this is the ith decryption query, then output (c, t).

   2.2 If $\langle c, t\rangle$ was a reponse to a previous encryption oracle query for some message m, then return m

   2.3 Otherwise, return $\perp$

Proof (2)

Adversary $\mathcal{A}_E$ (oracle access to $\mathrm{Enc}_{k_E}(.)$)

1. Choose uniform $k_M \in \{0,1\}^n$.
2. Run $\mathcal{A}$ on 1^n.

   When $\mathcal{A}$ makes an encryption oracle query for m, do the following:

   2.1 Compute ciphertext c of m using $\mathrm{Enc}_{k_M}(.)$.
   2.2 Compute $t \leftarrow \mathrm{Mac}_{k_M}(c)$ and give $\langle c, t \rangle$ to $\mathcal{A}$.

   When $\mathcal{A}$ makes a decryption oracle query for $\langle c, t \rangle$, answer it as follows:

   2.1 If $\langle c, t \rangle$ was a reponse to a previous encryption oracle query for m, then return m. Otherwise, return $\perp$.
3. When $\mathcal{A}$ outputs m_0, m_1, output these messages and recieve the challenge c. Compute $\mathrm{Mac}_{k_M}(c)$ and return $\langle c, t \rangle$ to $\mathcal{A}$. Answer further queries of $\mathcal{A}$ as above.
4. Output the same bit b' as $\mathcal{A}$.