Authentication

Debdeep Mukhopadhyay
IIT Kharagpur

Encryption vs Message Authentication

- Does ciphers provide authentication?
  - Stream Ciphers: Flipping a bit of the ciphertext, results in the same bit being flipped in the message.
  - Block Ciphers: OFB and counter modes are like stream ciphers.

    Even for ECB mode, changing a block affects only the block.

    For CBC mode, changing the jth bit of IV, changes the jth bit of the first message block.

Message Authentication Codes (MAC)

A MAC is a tuple of PPTalgorithms (Gen,Mac,Vrfy) st:
1. The key generation algorithm Gen takes as input the security parameter n, and outputs a key k, $|k| \geq n$.
2. The tag generation algorithm Mac, takes as input a key k, a message $m \in \{0,1\}^*$, and outputs a tag t.
We write this as $t \leftarrow \text{Mac}_k(m)$.
3. The verification algorithm Vrfy takes as input a key k, a message m, and a tag t. It outputs, b=1, to indicate Valid, and b=0 to indicate invalid.
We assume wlog. Vrfy is deterministic, and thus, b=Vrfy(m,t)

Fixed length MAC

It is required for every n, every k output by Gen, and every message m, $\text{Vrfy}_k(m, Mac_k(m)) = 1$.
If (Gen,Mac,Vrfy) is such that for every k output by Gen, algorithm Mac is only defined for messages of length l(n) (and Vrfy_k outputs 0 for any message $m \notin \{0,1\}^{l(n)}$, then we say that (Gen,Mac,Vrfy) is a fixed length Mac for messages of arbitrary length.

MAC-forge experiment

The message authentication experiment Mac-forge$_{A,\Pi}(n)$:

1. A random key k is generated by running Gen(n).
2. The adversary A is given n, and oracle access to Mac$_k$(.).
Let Q denote the set of all the oracle accesses.
The adversary finally produces an (m,t).
3. The adversary is successful (indicated by the experiment returning 1), if and only if:

i) $\text{Vrfy}_k(m,t) = 1$

ii) $m \notin Q$

Secure MAC-*formally*

A MAC is existentially unforgeable under an adaptive chosen message attack, or just secure if for all PPT adversaries A, there exists a negligible function negl such that:

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

Is this definition strong?

- Formalism says that if the adversary is able to generate the MAC of *any* message it suffices:
  - But the message may not be valid.
- We show a demonstration to see why this definition is needed.
- Further the definition makes security of MAC independent of applications.

Constructing a Fixed length MAC

Let F be a pseudorandom function. Define a fixed length MAC for messages of length n as follows:

1. Gen: on input n, choose k $\leftarrow \{0,1\}^n$ uniformly at random.
2. Mac: Compute, tag t=F_k(m). If $|m| \neq$ n, then output nothing.
3. Vrfy: Check t=F_k(m). If $|m| \neq$ n, then output 0.

Theorem

If F is a pseudorandom function, then the above scheme is a fixed-length Mac for messages of length n, that is secure under an adaptive chosen message attack.

Proof Outline

- *Replace the pseudorandom function with a random function.*
- *If the MAC is insecure when the function is replaced by a pseudorandom function, another PPT adversary D can use this fact to distinguish the pseudorandom function from a random function.*
- *D who is provided with an oracle with the task of distinguishing from a random function, employs the MAC-adversary, A.*
- *For all messages which A sends, D uses its oracles to generate the tags.*
- *Finally, when A provides (m,t), where m is new, D checks whether its oracle also produces the same output. Then it produces a 1, else 0.*

Extension to variable lengths

- Split the message into d blocks, pad the last by 0's so that each is of size n bits.
- Apply the fixed length MAC for messages of size n on each block.
  - XOR all the blocks and then authenticate
  - Authenticate each block separately.
  - Authenticate each block along with a sequence number: $t_i = MAC_k(i||m_i)$
- but none of them works.

The final MAC construction

Let Π'=(Gen',Mac',Vrfy') be a fixed length MAC for messages of length n. Define a MAC as follows:

1. Gen: Same as Gen'
2. On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^*$ of length $l < 2^{n/4}$, parse m into d blocks $m_1, ..., m_d$, each of length n/4. (Final block is padded if needed). Choose a random identifier, $r \leftarrow \{0,1\}^{n/4}$.

The final MAC construction

For i=1,...,d, compute $t_i = Mac'_k(r \,\|\, l \,\|\, i \,\|\, m_i)$,
where i and l are uniqely coded strings of length n/4. Finally, output the tag, $t=(r, t_1,...,t_d)$.

3. Vrfy: on input a key k, and a message m of length $l < 2^{n/4}$, and a tag $t=(r, t_1,...,t_{d'})$, parse m into d blocks $m_1,...,m_d$, each of length n/4.
(Final block is padded if needed).
output 1 iff d'=d, and $vrfy_k(r \,\|\, l \,\|\, i \,\|\, m_i, t_i) = 1$ for $1 \le i \le d$

Theorem

If Π' is a secure fixed length MAC for messages of length n, then the above construction is a MAC that is secure under an adaptive chosen message attack.

Proof

Let Π denote the MAC. Let *A* be a PPT algorithm, and define:

$$\Pr[\text{Mac-forge}_{A,\Pi}(\text{n})=1]$$

Repeat: Same message identifier appears in two of the tags returned by MAC oracles.

Forge: At least one of the blocks $r \| l \| i \| m_i$ was never previously authenticated by the MAC oracle, yet $\text{Vrfy}'(r \| l \| i \| m_i) = 1$

Proof (Contd.)

$$\begin{aligned}\Pr[\text{Mac-forge}_{A,\Pi}(n)=1] = {} & \Pr[\text{Mac-forge}_{A,\Pi}(n)=1 \wedge \text{Repeat}] \\ & + \Pr[\text{Mac-forge}_{A,\Pi}(n)=1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{Forge}}] + \\ & + \Pr[\text{Mac-forge}_{A,\Pi}(n)=1 \wedge \overline{\text{Repeat}} \wedge \text{Forge}]\end{aligned}$$

Claim 1

There is a *negl* function, ε such that:

$$\Pr[\text{Repeat}] \leq \varepsilon(n)$$

Proof: Let q(n) be the number of MAC oracle queries made by *A*.

In the *ith* query, oracle chooses, $r_i \leftarrow \{0,1\}^{n/4}$ uniformly.

Thus, $\Pr[\text{Repeat}] \leq \frac{q(n)^2}{2^{n/4}}$

Claim 2

$$\Pr[\text{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{Forge}}] = 0$$

$$\therefore \text{ If, Mac-forge}_{A,\Pi}(n) = 1 \text{ and Repeat} = 0 \Rightarrow \text{Forge} = 1$$

Let, (m,t) be the final output of A [the forged message].

Let its length be l, and the identifier is r.

Thus, $t = < r, t_1, ..., t_d >$.

Parse $m = (m_1, ..., m_d)$, each of length n/4. Last block may be padded with 0s.

Case 1

Case 1: Identifier r is different from all the identifiers used by the MAC oracles.

$\Rightarrow r \| l \| 1 \| m_1$ was never previously authenticated by the MAC oracle.

Since, Mac-forge$_{A,\Pi}(n) = 1 \Rightarrow Vrfy'_k(r \| l \| 1 \| m_1, t_1) = 1$.

Thus, Forge occurs.

Case 2

Identifier r was used in exactly one of the MAC tags obtained by A from its oracles.

Denote by (m', t') the query-response pair, when the identifier r occurred.

$\because m \notin Q \Rightarrow m \neq m'$.

Let l' be the length of m'.

Case 2a

$Case 2a : l \neq l'$

This implies, $r \| l \| 1 \| m_1$ was never previously authenticated by the MAC oracle.

This is because all MAC oracle responses used a different identifier, and the one oracle that used the same identifier, has a different length value.

Since, Mac-forge$_{A,\Pi}(n) = 1 \Rightarrow Vrfy'_k(r \| l \| 1 \| m_1, t_1) = 1$.

Thus, Forge occurs.

Case 2b

Case 2b: $l = l'$

Parse $m' = (m'_1, ..., m'_d)$.

Note: since $l = l'$, the number of blocks in m and m' are same.

Since, $m \neq m'$, $\exists$i, st. $m'_i \neq m_i$.

But, then $r \| l \| i \| m_i$ was never authenticated.

All previous oracles, except one had different identifiers.

The one with the same identifier, had different sequence numbers, $i' \neq i$ in all the blocks except one; in this remaining block it used $m'_i \neq m_i$.

Since, Mac-forge$_{A,\Pi}(n) = 1 \Rightarrow Vrfy'_k(r \| l \| 1 \| m_1, t_1) = 1$.

Thus, Forge occurs.

Thus,

$$\Pr[\text{Mac-forge}_{A,\Pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \text{Forge}] \geq \varepsilon(n) - \frac{q(n)}{2^{n/4}}$$

Adversary A' against fixed length MAC

- A' runs A as a subroutine.
- Whenever A requests for a tag, it generates an identifier r, and makes queries appropriately to its own fixed length MAC.
- When A outputs, (m,t), A' parses m and sees any m_i which did not occur in its previous oracle queries (to the fixed MAC).
- If it finds such it outputs, $(r||l||i||m_i,t_i)$ as a valid MAC. If not, it outputs nothing.

Success Probability of A'

$$\begin{aligned}\Pr[\text{Mac-forge}_{A',\Pi'}(n)=1] &\geq \Pr[\text{Mac-forge}_{A,\Pi}(n)=1 \wedge \text{Forge}] \\ &\geq \Pr[\text{Mac-forge}_{A,\Pi}(n)=1 \wedge \text{Forge} \wedge \overline{\text{Repeat}}] \\ &\geq \varepsilon(\text{n}) - \frac{q(n)}{2^{n/4}}\end{aligned}$$

CBC-MAC

- Previous construction is inefficient.
- Large number of block cipher calls required.
- Message tag also large in length.
  - for message length = l.n, block cipher needs tp be applied 4l times.
  - Message tag length also more than 4l.n

CBC-MAC for fixed length messages

Let E be a pseudorandom function, and fix a length l. The basic CBC-MAC for fixed length messages is:

1. Gen: On input n, choose $k \leftarrow \{0,1\}^n$ uniformly at random.
2. Mac: on input $k \in \{0,1\}^n$ and a message of length $l.n$ and repeat the following steps:
   1. Parse $m = m_1 ... m_l$ where each m_i is of length n, and set $t_0 = 0^n$.
   2. For $i = 1$ to l, $t_i = F_k(t_{i-1} \oplus m_i)$.

   Output t_l as the tag.
3. Vrfy: on input a key $k \in \{0,1\}^n$, a message of length $l.n$, and a tag of length n, output 1 if and only if $t = Mac_k(m)$.

Note that the IV is set to 0, and not random as for CBC encryptionn.

Security

- Let l be a polynomial in n. If F is a pseudorandom function, then the above construction is a fixed length MAC for messages of length l.n and is existentially unforgeable under an adaptive chosen message attack.

Not secured if used for messages of arbitrary length

Not secured if used for messages of arbitrary length

- Adversary can forge in that case.
- Consider a message m_1, and a tag $t_{1.}$
  - Thus, $t_1=MAC_k(m_1)$
- Likewise, $t_2=MAC_k(m_2 \text{ xor } t_1)$
- Thus the MAC for the message $m_1||m_2$ can be forged as t_2.

What if the IV is random?

- If the IV is random, it is a part of the tag.
- Consider a message m of one block length.
- Let the tag for m be (IV,t).
- Thus, a valid tag for IV is (m,t).
- **So, in CBC-MAC the IV is not used.**
  - **this shows that it is dangerous to change cryptographic primitives without proper analysis!**

Another difference with CBC-encrypt

- **In CBC-encrypt, we export each block encryption, but not so in CBC-MAC.**
- Consider a message made of two blocks, $m_1||m_2$, and the corresponding tag as $t_1||t_2$.
- Thus, $t_1=F_k(m_1)$, and $t_2=F_k(t_1 \text{ xor } m_2)$.
- How can you forge a tag using this?

Another difference with CBC-encrypt

- **In CBC-encrypt, we export each block encryption, but not so in CBC-MAC.**
- Consider a message made of two blocks, $m_1||m_2$, and the corresponding tag as $t_1||t_2$.
- Thus, $t_1=F_k(m_1)$, and $t_2=F_k(t_1 \text{ xor } m_2)$.
- How can you forge a tag using this?
  - Consider a message t_1 xor $m_2||t_2$ xor m_1
  - Its valid tag is $t_2||t_1$.

CBC-MAC for arbitrary length

- Prepend the message with its length |m| (encoded as n-bit string), and then compute the basic CBC-MAC.
- **What if we append the message with the length?**

CBC-MAC for arbitrary length

- Change the key generation to choose two keys k_1 and k_2 of length n.
- Thus to authenticate a message of length n, first compute a basic CBC-MAC using k_1: $t=MAC_{k1}(m)$
- Then output tag, $t'=F_{k2}(t)$
  - in this case one does not need to know the length of the message before the MAC computation.

CCA-secured encryption scheme

- We have seen that the previous encryption schemes are vulnerable to CCA attacks.
- We show here that message authentication codes along with CPA secured schemes are CCA-secured.

The CCA-secure encryption scheme

Let $\Pi_E = (Gen_E, Enc, Dec)$ be a private key encryption scheme, and let $\Pi_M = (Gen_M, Mac, Vrfy)$ be a message authentication code.

Define an encryption scheme $\Pi' = (Gen', Enc', Dec')$ as follows:

1. Gen': on input n, run $\text{Gen}_E(n)$ and $\text{Gen}_M(n)$ to obtain keys k_1, k_2.
2. Enc': on input a key (k_1, k_2) and a plaintext m, compute: $c = Enc_{k_1}(\text{m})$, and $t = Mac_{k_2}(\text{c})$, and output ciphertext <c,t>
3. Dec': on input a key (k_1, k_2) and a ciphertext <c,t>, first check $\text{Vrfy}_{k_2}(\text{c,t})=1$, and then output $\text{Dec}_{k_1}(\text{c})$, if Vrfy returns 1, else output $\perp$.

Note that no ciphertext generated by Enc' will be decrypted to $\perp$

Security Proof

- Before we go into the proof, we impose an additional requirement of the MAC, that the MACs have to be unique.
  - ie. for every k and m, there is a unique value t st. $\text{Vrfy}_k(\text{m,t})=1$.
- This is not problematic, as we have seen CBC-MAC as to be unique.

Theorem

If Π_E is a CPA-secure private key encryption scheme and Π_M is a secure message authentication code with unique tags, then the construction is a CCA-secure private key encryption scheme.

Proof Idea

- **The adversary is a CCA adversary and hence can make decryption queries.**
- The queries to the decryption oracle can be of two types:
  - ciphertexts that are generated from its encryption oracles:
    - adversary already knows that the message is m.
  - those that are not generated from encryption oracle, but which are valid (pass the verification):
    - this event is called ValidQuery
    - when it occurs the MAC is forged
- **Thus if the CCA adversary has to win the challenge, then it ask queries of both these types:**
  - first one does not give any extra information and hence is not useful.
  - second one occurs with a very small probability as the MAC is secure.
  - thus the adversary is reduced to the CPA setting.

Detailed Proof

Let A be a PPT adversary attacking the scheme in a CCA attack.

Let ValidQuery be the event that A submits a query <c,t> to its decryption oracle that was not previously obtained from its encryption oracle, but for which $\mathrm{Vrfy}_{k_2}(c,t)=1$.

Thus, $\Pr[\mathrm{PrivK}^{cca}_{A,\Pi'}(n)=1]$

$$= \Pr[\mathrm{PrivK}^{cca}_{A,\Pi'}(n)=1 \wedge \mathrm{ValidQuery}] + \Pr[\mathrm{PrivK}^{cca}_{A,\Pi'}(n)=1 \wedge \overline{\mathrm{ValidQuery}}]$$

$$\leq \Pr[\mathrm{ValidQuery}] + \Pr[\mathrm{PrivK}^{cca}_{A,\Pi'}(n)=1 \wedge \overline{\mathrm{ValidQuery}}]$$

ValidQuery occurs with negligible probability

A is the PPT adversary attacking the scheme in a CCA attack.

Let q(.) be a polynomial that upper bounds the number of dcryption oracle queries made by A.

Consider the following adversary A_M attacking the MAC Π_M through $\text{Mac-forge}_{A_M,\Pi_M}(n)$:

Define the MAC adversary

Adversary A_M has access to oracle $\text{Mac}_{k_2}(.)$:

1.Choose $k_1 \leftarrow \{0,1\}^n$

2. Choose $i \leftarrow \{1,...,q(n)\}$

3.Run A on input n. A makes encryption and decryption queries.

4. The encryption queries are answered as follows:

    1. Compute $c = Enc_{k_1}(\text{m})$

    2. Query c to the MAC oracle, and receive t in response.

  Return <c,t> to A.

It also creates the challenge ciphertext in the usual way, by randomly choosing a bit $b \leftarrow \{0,1\}$, and encrypting m_b.

Define the MAC adversary

5. The decryption queries are answered as follows:

    When A makes a decryption query to $< c,t >$,

  A_M answers as follows:

    1. If <c,t> was a response to a previous encryption oracle for a message m, return m.

    2. If this is the *ith* decryption oracle query using a new value of c, output (c,t) and stop.

    3. Otherwise output $\perp$.