4.1 Message Integrity

4.1.1 Secrecy vs Integrity

密码学最基本的目标是在开放信道实现安全交流（通信双方）

在第三章，我们通过加密的方式展现了如何在未经保护的信道实现双方的安全通信，并且敌手没有获得消息的任何信息。

网络通讯过程中，为了保证信息安全，需要考虑多方面的因素。比较重要的关键点：

- 完整性（Integrity）：确保信息在传输过程中，没有被篡改。

书中也叫消息认证，in the sense that each party should be able to identify when a message it receives was sent by the party **claiming** to send it, and was **not modified** in transit.

举个例子说明消息验证的重要性：

Consider the case of a user communicating with their bank over the Internet. When the bank receives a request to transfer $1,000 from the user's account to the account of some other user X, the bank has to consider the following:

1. Is the request authentic? That is, did the user in question really issue this request, or was the request issued by an adversary (perhaps X itself) who is impersonating the legitimate user?
2. Assuming a transfer request was issued by the legitimate user, are the details of the request as received exactly those intended by the legitimate user? Or was, e.g., the transfer amount modified?

然后就是一些错误相关的编码技术可能并不足以用作消息验证，因为它只能纠错（海明码）或者检错（CRC）出一些随机的传输上的错误。并不足以对抗敌手可能对你的消息造成大量的一个错误。

本章讲的是如何使用**MAC**（message authentication code）检测*message*是否被篡改。

4.1.2 Encryption vs. Message Authentication

实质上就目标来说，保密和消息完整性是不同的。

大多数人存在一个普遍的**误区**：加密算法能够解决消息完整性问题。这就是把**Encryption**和**Message Authentication**混为一谈，实际上加密并不能够保证消息的完整性，也不能实现消息验证，除非是有特定的需要4.5节会讲。

好像加密过后密文完全隐藏了消息的内容，然后敌手就不能去修改内容了。但是事实上，后面会证明所有的加密方案都远远不能保证消息的完整性。

Encryption using stream ciphers

考虑一种简单的加密方案，加密算法$Enc_k(m)$

计算密文$c := G(k) \oplus m$,这种情况下，敌手翻转密文中的第i位，明文中的第i也会相应的得到翻转。比如前面银行的例子，$1000是以二进制传输的，所以反转这个数字的一位，就会对转钱的用户造成很大的损失。

4.2 Message Authentication Codes – Definitions

Encryption一般并不能解决**Message Integrity**问题，对于这个问题最常用的工具是**Message Authentication Code, MAC**，**MAC**的目标就是检测**message**是否被敌手篡改。

The Syntax of a Message Authentication Code

这部分主要讲述MAC的定义以及如何使用。在对称加密中，通信双方首先共享密钥k，**sender**发送(m,t)给接收方，其中t是**MAC tag**,计算方法为$t \leftarrow Mac_k(m)$ **receiver**接收(m,t), 并通过**Vrfy**验证t是否有效，即m 是否被敌手篡改（通过共享密钥来完成）。
正式描述如**DEFINITION 4.1**所示。

> ***DEFINITION 4.1*** *A* message authentication code *(or* MAC*) consists of three probabilistic polynomial-time algorithms* (Gen, Mac, Vrfy) *such that:*
>
> 1. *The* key-generation algorithm Gen *takes as input the security parameter* 1^n *and outputs a key* k *with* $|k| \geq n$.
> 2. *The* tag-generation algorithm Mac *takes as input a key* k *and a message* $m \in \{0,1\}^*$*, and outputs a tag* t*. Since this algorithm may be randomized, we write this as* $t \leftarrow \mathsf{Mac}_k(m)$.
> 3. *The deterministic* verification algorithm Vrfy *takes as input a key* k*, a message* m*, and a tag* t*. It outputs a bit* b*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid*. We write this as* $b := \mathsf{Vrfy}_k(m,t)$.

Canonical verification

Canonical verification的基本思想是再计算一遍m的Mac值，计算出的t' 和接收到的t相等则通过验证。该方法隐含着$Mac_k()$ 是一个确定算法。

Security of Message Authentication Codes

MAC安全最直观的定义是：不存在**Efficient Adversary**对**任何未经发送或者验证的信息**生成**Valid tag**。

为了形式化定义安全性，首先对敌手的能力进行限制。

在消息鉴别码的情形中。敌手的能力：

- 能够看到通信双方的消息以及对应的MAC标记
- 敌手可能直接或者间接地影响消息的内容。（比如敌手就是一个通信方的一个私人助理，就能够控制所发的消息）

所以说为了适应这个场景，允许敌手去访问$MAC_k()$并且能够得到提交$message$的标记t。

接着就是对一个MAC方案进行攻击。

如果敌手能够输出任意消息的m和它的标记t，而且满足下面两种情况：

1. t是m的有效标记，即$Vrfy(m,t) = 1$
2. 敌手之前并没有获得过m的标记t

就说明该MAC方案被攻破了。

第二个条件的限制就是说，敌手可能复制发送方之前发送过的m和t，但是这种情况并不算做攻破了MAC安全。这种通常被称作**重放攻击**。

重放攻击就是说敌手获得了两个通信双方的信道上的（m，t）。就可以伪装成发送方，一直向接收方发送。

Consider again the scenario where a user (say, Alice) sends a request to her bank to transfer $1,000 from her account to some other user (say, Bob). In doing so, Alice can compute a MAC tag and append it to the request so the bank knows the request is authentic. If the MAC is secure, Bob will be unable to intercept the request and change the amount to $10,000 because this would involve forging a valid tag on a previously unauthenticated message. However, nothing prevents Bob from intercepting Alice's message and replaying it **ten times** to the bank.

正如前面所述，MAC并不能抵抗Replay attack，因为Replay attack并没有破坏MAC的安全性。
那么如何抵抗Replay attack呢?

方法	sequence numbers (or counters)	time-stamps
描述	序列号(计数)，详细描述在4.5.3节	sender发送当前时间 T ,并通过 T ||m 计算**tag** t
条件	通信双方保持同步状态	通信双方保持相近的同步时钟
缺点	概率性的丢包可能会产生一些问题	**Replay attack**足够快时也有可能攻击成功

满足前面定义的安全级别的MAC称为：在适应性选择攻击下的存在性不可伪造。

存在性不可伪造就是：敌手无法伪造任何消息的标记。

适应性选择消息攻击：敌手可以获得任何消息的MAC标记。这些标记会在攻击中被适应性地选择。

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

1. *A key* k *is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary* $\mathcal{A}$ *is given input* 1^n *and oracle access to* $\mathsf{Mac}_k(\cdot)$. *The adversary eventually outputs* (m, t). *Let* $\mathcal{Q}$ *denote the set of all queries that* $\mathcal{A}$ *asked its oracle.*
3. $\mathcal{A}$ succeeds *if and only if (1)* $\mathsf{Vrfy}_k(m, t) = 1$ *and (2)* $m \notin \mathcal{Q}$. *In that case the output of the experiment is defined to be* 1.

所以一个方案是MAC安全的就是说在上面这个实验成功的概率是可忽略的。

也就是下面定义4.2.

DEFINITION 4.2 *A message authentication code* $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* existentially unforgeable under an adaptive chosen-message attack, *or just* secure, *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there is a negligible function* negl *such that:*

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

Strong MACs

secure MAC保证敌手不能对**New Message**伪造有效的tag，但并没有限制敌手不能够对**验证过**的message伪造有效的tag：假设$Mac_k()$是一个随机函数，即$Mac_k(m_i)$可能存在多个有效的tag t_i，那么敌手就能够对验证过得message m_i求得一个新的有效tag $\tilde{t}_i != t_i$.这种情况在现实中可能会对通信双方造成某些损害。为了排除这种情况，需要定义**Strong MAC**。

DEFINITION 4.3 *A message authentication code* $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* strongly secure, *or a* strong MAC, *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$, *there is a negligible function* negl *such that:*

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

A potential timing attack

上面没有提到的一个问题是对MAC verification进行timing attack。敌手发送(m,t)给接收方，不仅能够了解到是accepts或者rejects的状态，还能了解到接收方作出这个决定花费了多长的时间；由这一特性，提出了timing attack：已知敌手伪造的前i 位的tag是对的，现在需要验证第 i+1位是否正确；如果第i+1位正确，那么rejects的时间会稍长，否则和前i位正确时所花费的时间一样。

对于timing attack的应对措施就是：再进行验证时比较tag的all bytes。

4.3 Constructing Secure Message Authentication Codes

4.3.1 A Fixed-Length MAC

伪随机函数是用来构建安全的MAC方案的常用工具。

CONSTRUCTION 4.5

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

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$. (If $|m| \neq |k|$ then output nothing.)
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^n$, and a tag $t \in \{0,1\}^n$, output 1 if and only if $t \stackrel{?}{=} F_k(m)$. (If $|m| \neq |k|$, then output 0.)

A fixed-length MAC from any pseudorandom function.

这个构造方法4.5是为了构造一个固定长度安全MAC。本节想要讨论如何将处理固定长度的MAC转化为能过够处理任意长度的消息的MAC方案。

首先有一个理论4.6，说明这是一个安全的固定长度的MAC方案。

THEOREM 4.6 *If* F *is a pseudorandom function, then Construction 4.5 is a secure fixed-length MAC for messages of length* n.

之后是对这个4.6的一个证明。（通过与真随机的不可区分来说明这个方案是安全的）

$$\left|\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\text{Mac-forge}_{\mathcal{A},\tilde{\Pi}}(n) = 1]\right| \leq \mathsf{negl}(n); \qquad (4.2)$$

4.3.2 Domain Extension for MACs

构造方法4.5构造了安全的固定长度的MAC，然而在许多应用当中，固定长度的消息是不可以接受的。所以下面说明通过定长消息的MAC来构建一个可用于变长消息的MAC。

对于变长的消息来说，采取的分块的方式，给定一个能处理的消息长度，将变长的消息分块成多个定长的来进行处理，并且运用定长的方案。

具体方案如下4.7

CONSTRUCTION 4.7

Let $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ be a fixed-length MAC for messages of length n. Define a MAC as follows:

- Mac: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ of (nonzero) length $\ell < 2^{n/4}$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Choose a uniform identifier $r \in \{0,1\}^{n/4}$.
  For $i = 1, \ldots, d$, compute $t_i \leftarrow \mathsf{Mac}'_k(r \| \ell \| i \| m_i)$, where i, ℓ are encoded as strings of length $n/4$.† Output the tag $t := \langle r, t_1, \ldots, t_d \rangle$.
- Vrfy: on input a key $k \in \{0,1\}^n$, a message $m \in \{0,1\}^*$ of length $\ell < 2^{n/4}$, and a tag $t = \langle r, t_1, \ldots, t_{d'} \rangle$, parse m into d blocks $m_1, \ldots, m_d$, each of length $n/4$. (The final block is padded with 0s if necessary.) Output 1 if and only if $d' = d$ and $\mathsf{Vrfy}'_k(r \| \ell \| i \| m_i, t_i) = 1$ for $1 \le i \le d$.

† Note that i and ℓ can be encoded using $n/4$ bits because $i, \ell < 2^{n/4}$.

A MAC for arbitrary-length messages from any fixed-length MAC.

然后跟定长方案一样，有一个理论4.8，说明只要用到的固定长度的MAC方案是安全的，那么这个变长的MAC方案也是安全的。

THEOREM 4.8 *If Π' is a secure fixed-length MAC for messages of length n, then Construction 4.7 is a secure MAC (for arbitrary-length messages).*

之后又是对该定理的一个证明。

$$\begin{aligned}
\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] &= \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \mathsf{Repeat}] \\
&\quad + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\mathsf{Repeat}} \wedge \mathsf{NewBlock}] \\
&\quad + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\mathsf{Repeat}} \wedge \overline{\mathsf{NewBlock}}] \\
&\le \Pr[\mathsf{Repeat}] \qquad (4.3) \\
&\quad + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \mathsf{NewBlock}] \\
&\quad + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1 \wedge \overline{\mathsf{Repeat}} \wedge \overline{\mathsf{NewBlock}}].
\end{aligned}$$

主要是证明敌手成功的概率是可忽略的。

Repeat代表两个tag t的r是相同的

NewBlock代表敌手生成了未经Mac'_k验证过的m的t

首先对于**Repeat**这个场景来说。假设存在

$$m =< m_1, m_2, \ldots, m_d > \qquad t =< r, t_1, \ldots, t_d >$$

$$m' =< m'_1, m'_2, \ldots, m'_d > \qquad t' =< r, t'_1, \ldots, t'_d >$$

因为r相同，所以敌手就可以根据这两个（m，t）对构造**新的消息**

$$m'' =< m_1, m'_2, m_3, \ldots, m'_d > \qquad t'' =< r, t_1, t'_2, t_3, \ldots, t'_d >$$

这个（m'',t''）是一定能通过验证的。

然后就是**NewBlock**就代表攻破了固定长度的MAC安全。

对于随机生成的r相同的概率是可忽略的，所以$Pr[Repeat] = negl(n)$

然后因为前面已经说明固定长度的MAC是一个安全的方案，所以

$$Pr[Mac - forge_{A,II}(n) = 1 \cap NewBlock] = negl(n)$$

最后对于$Pr[Mac - forge_{A,II}(n) = 1 \cap \overline{NewBlock} \cap \overline{Repeat}] = 0$

因为综合考虑了所有情况，有Repeat事件发生则NewBlock必然发生。

也就是说，如果Mac-forge$_{\mathcal{A},\Pi}(n) = 1$且Repeat事件并未发生，就意味着Forge事件发生。下面证明这一点。还记得$\mathcal{A}$最后输出(m,t)，l表示m的长度。将m分成d个分块：$m_1, \cdots, m_d$，每个长度是$n/4$（必要时用 0 填充最后一个分块）。有$t = \langle r, t_1, \cdots, t_d \rangle$；既然Mac-forge$_{\mathcal{A},\Pi}(n) = 1$，知道$t$包括$d+1$个部分。有如下几种情况：

(1) 标识码r与 MAC 预言机所使用的标识码不同。这意味着$r\|l\|1\|m_1$从未被 MAC 预言机鉴别过。因为Mac $-$ forge$_{\mathcal{A},\Pi}(n) = 1$，有Vrfy$'_k(r\|l\|1\|m_1, t_1) = 1$，因此，Forge事件在这种情况下发生。

(2) 标识码r使用在$\mathcal{A}$从 MAC 随机预言机获得的多个 MAC 标记中的一个标记中。用m'代表$\mathcal{A}$查询随机预言机的消息，该消息的应答t'中包含标识码r。因为$m \notin Q$，有$m' \neq m$。用l'表示m'的长度。有两种情况：

① 情况 1：$l' \neq l$。则依旧有$r\|l\|1\|m_1$从未被 MAC 预言机鉴别过。（MAC 预言机应答有两种情况：一个使用不同的标识码$r' \neq r$，以及使用相同标识码但长度不同$l' \neq l$）。因为Mac-forge$_{\mathcal{A},\Pi}(n) = 1$，所以Forge事件在这种情况下发生。

②情况 2：$l' = l$。将m'分成d个分块$m'_1, \cdots, m'_d$（既然$l' = l$，那么m和m'中分块数量是相同的）。因为$m' \neq m$，肯定存在某些i使得$m'_i \neq m_i$。于是存在$r\|l\|i\|m_i$从未被 MAC 预言机鉴别过。（预言机应答分下述情况：标识码不同$r' \neq r$；使用相同标识码r，但序列号不同$i' \neq i$，阶段$m'_i \neq m_i$。）因为Mac-forge$_{\mathcal{A},\Pi}(n) = 1$，Forge事件在这种情况下发生。

(3) 标识码r使用在多个 MAC 标记中，这些标记是$\mathcal{A}$从预言机得到的。这种情况是不可能发生的，因为Repeat事件不发生。