第七讲
密码算法
Lecture 7
Cryptographic Algorithms

明玉瑞 Yurui Ming

yrming@gmail.com

声明
Disclaimer

- 本讲义在准备过程中由于时间所限，所用材料来源并未规范标示引用来源。所引材料仅用于教学所用，作者无意侵犯原著者之知识版权，所引材料之知识版权均归原著者所有；若原著者介意之，请联系作者更正及删除。

  The time limit during the preparation of these slides incurs the situation that not all the sources of the used materials (texts or images) are properly referenced or clearly manifested. However, all materials in these slides are solely for teaching and the author is with no intention to infringe the copyright bestowed on the original authors or manufacturers. All credits go to corresponding IP holders. Please address the author for any concern for remedy including deletion.

密码系统
Cryptosystem

- 一个密码系统是满足以下条件的五元组（$\mathcal{P}$，$\mathcal{C}$，$\mathcal{K}$，$\mathcal{E}$，$\mathcal{D}$）：
  - $\mathcal{P}$表示所有可能的明文组成的有限集；
  - $\mathcal{C}$表示所有可能的密文组成的有限集；
  - $\mathcal{K}$代表秘钥空间，由所有可能秘钥组成的有限集；
  - 对每一个$K \in \mathcal{K}$，都存在一个加密规则$e_K \in \mathcal{E}$和相对应的解密规则$d_K \in \mathcal{D}$，对每对e_K：$\mathcal{P} \to \mathcal{C}$，$d_K$：$\mathcal{C} \to \mathcal{P}$，满足条件：对每一个明文$x \in \mathcal{P}$，均有$d_K(e_K) = x$。

代换密码
Substitution Cipher

- 代换密码系统
  - 令$\mathcal{P}=\mathcal{C}=\mathbb{Z}_{26}$，$\mathcal{K}$是由26个数字0到25的所有可能的置换组成的集合，对任意的置换$\pi\in\mathcal{K}$，定义$e_\pi(x)=\pi(x)$，$d_\pi(y)=\pi^{-1}(y)$，这里π^{-1}表示π的逆置换。
- 在代换密码的情形下，可以认为$\mathcal{P}$和$\mathcal{C}$是26个英文字母（区分大小写的情况下为52个字母），将加密与解密过程看作一个字母表上的置换。
- 下面展示了一个代换密码的例子，其中小写字母表示明文，大写字母表示密文：

a	b	c	d	e	f	g	h	i	j	k	l	m
X	N	Y	A	H	P	O	G	Z	Q	W	B	T

n	o	p	q	r	s	t	u	v	w	x	y	z
S	F	L	R	C	V	M	U	E	K	J	D	I

加密

A	B	C	D	E	F	G	H	I	J	K	L	M
d	l	r	y	v	o	h	e	z	x	w	p	t

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	g	f	j	q	n	m	u	s	k	a	c	i

解密

置换密码
Permutation Cipher

- 置换密码系统
  - 令m为一正整数，令$\mathcal{P}=\mathcal{C}=(\mathbb{Z}_{26})^m$，$\mathcal{K}$是由所有定义在集合$\{1,2,\cdots,m\}$上的所有可能的置换组成的集合，对任意的置换$\pi\in\mathcal{K}$，定义：

$$e_\pi(x_1,x_2,\cdots,x_m)=\pi\left(x_{\pi(1)},x_{\pi(2)},\cdots,x_{\pi(m)}\right)$$
$$d_\pi(y_1,y_2,\cdots,y_m)=\pi^{-1}\left(y_{\pi^{-1}(1)},y_{\pi^{-1}(2)},\cdots,y_{\pi^{-1}(m)}\right)$$

  这里π^{-1}表示π的逆置换。
- 下面展示了一个置换密码的例子。

x	1	2	3	4	5	6
$\pi(x)$	3	5	1	6	4	2

加密

x	1	2	3	4	5	6
$\pi^{-1}(x)$	3	6	1	5	2	4

解密

置换密码
Permutation Cipher

- 假设需要加密的明文是shesellsseashellsbytheseashore

1. 则首先将明文以6个字母为单位进行分组：

   shesel | lsseas | hellsb | ythese | ashore

2. 然后根据选定的置换π，对每组内的字母进行置换，得到如下结果：

   EESLSH | SALSES | LSHBLE | HSYEET | HRAEOS

3. 最后，得到密文： EESLSHSALSESLSHBLEHSYEETHRAEOS

分组密码
Block Cipher

- 分组密码或块密码是每次只能处理特定长度的一块数据的一类密码算法，这里的一个数据块（block）就称为一个分组。
- 一个分组的比特数就称为分组长度（block lenght）。例如，DES和3DES的分组长度为64比特，AES的分组长度为128比特。
- 分组密码常用的一种设计是迭代密码的设计，每次迭代都包含一系列置换和替换操作，称为轮函数，明文加密将经过 N 轮类似的过程。因此，基于迭代机制的分组密码需要指定一个轮函数和一个密钥表。
- 分组密码处理完一个分组就结束了，因此不需要通过内部状态来记录加密的进度，分组密码算法只能加密固定长度的分组，但是我们需要加密的明文长度可能会超过分组密码的分组长度，这时就需要对分组密码算法进行迭代，以便将一段很长的明文全部加密。而迭代的方法就称为分组密码的模式（mode）。

分组密码
Block Cipher

- 下面从整体上讲述一下分组密码的加密过程。
- 设K是某个指定长度的随机二进制密钥，一般通过固定的公开算法，由K构造N个轮密钥（也称为子密钥），记为$(K^1,\cdots,K^N)$。该密钥列表称为密钥编排方案。
- 假定已经选定轮函数g，其第r轮迭代以轮密钥K^r和上次迭代输出（亦称为当前状态）w^{r-1}两个变量作为输入，其输出w^r定义为下一个状态。初态w^0被定义为明文x，密文y定义为经过N后的状态，如下所示：

$$w^0 \leftarrow x$$
$$w^1 \leftarrow g(w^0, K^1)$$
$$\cdots$$
$$w^{N-1} \leftarrow g(w^{N-2}, K^{N-1})$$
$$w^N \leftarrow g(w^{N-1}, K^N)$$
$$y \leftarrow w^N$$

分组密码
Block Cipher

- 为了能够解密，轮函数g必须在其第二个自变量固定，即同一个密钥的条件下，是单射函数，这等价于存在函数g^{-1}，对所有的w和K，有$g^{-1}(g(w,K),K)=w$，其解密过程如下：

$$
\begin{gathered}
w^N \leftarrow y \\
w^{N-1} \leftarrow g^{-1}(w^N, K^N) \\
\cdots \\
w^1 \leftarrow g^{-1}(w^2, K^2) \\
w^0 \leftarrow g^{-1}(w^1, K^1) \\
x \leftarrow w^0
\end{gathered}
$$

分组密码
Block Cipher

- 上面我们从定义的角度，引入了迭代密码设计并从形式上展示了加密与解密过程。为了具化迭代密码设计，我们引入代换-置换网络的概念。
- 设l和m都是正整数，代换-置换网络一般是从如下两个操作π_S与π_P构建：

$$\pi_S: \{0,1\}^l \to \{0,1\}^l$$
$$\pi_P: \{1,\cdots,lm\} \to \{1,\cdots,lm\}$$

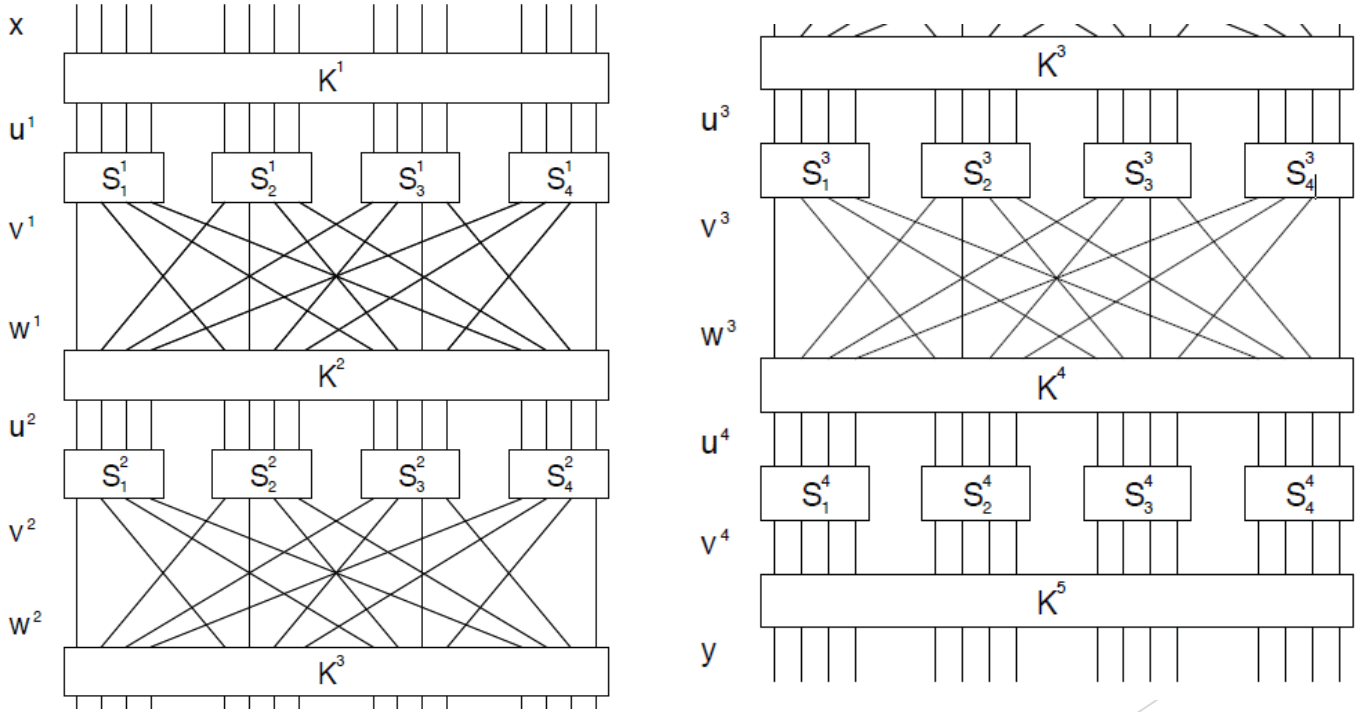
  其中π_S与π_P均为置换操作。 π_S称为S-盒（S为substitute的首字母缩写），是长度为l的2^l字符串上的置换操作，通常是用一个长度为l位的串，替换另一个长度为l位的串。对π_P而言，是通过改变长度为lm的串的每个位的位置的置换操作。
- 对于一个长度为lm的串，假设为$x = (x_1, x_2, \cdots, x_{lm-1}, x_{lm})$，为了适配算法，$x$亦可看成$m$个长度为$l$的串的连接，记为$x = x_{\langle 1\rangle} \| x_{\langle 2\rangle} \| \cdots \| x_{\langle m-1\rangle} \| x_{\langle m\rangle}$，其中$x_{\langle i\rangle}$代表第$i$个长度为$l$的串。

分组密码
Block Cipher

- 对第i个长度为l的串，假设$1 \le i \le m$，则有
$$x_{\langle i \rangle} = \left(x_{(i-1)l+1}, x_{(i-1)l+2}, \cdots, x_{il-1}, x_{il}\right)$$
- 下面引入代换-置换网络的概念，令l、 m、 N为正整数，令$\pi_S: \{0,1\}^l \to \{0,1\}^l$，$\pi_P: \{1, \cdots, lm\} \to \{1, \cdots, lm\}$为置换操作。令$\mathcal{P} = \mathcal{C} = \{0,1\}^{lm}$， $\mathcal{K} \subseteq \left(\{0,1\}^{lm}\right)^{N+1}$包含所有能够从初始密码$K$推导的密码编排方案。对选定的编排方案$(K^1, \cdots, K^N)$，我们用上面讲解的加密过程加密明文$x$。
- 整体而言，对于由代换-置换网络构成的迭代密码系统，在N操作的每轮操作中（最后一轮除外），我们将先用π_S进行m次代换操作，然后再用π_P进行一次置换操作。注意，在代换操作之前，我们会用异或操作混入本轮密钥。

分组密码
Block Cipher

- 具体而言，对于第r轮操作，令u^r为S-盒的输入，u^r为S-盒的输出，同时作为π_P的输入，其输出为w^r。 u^{r+1}为将w^r混入第$r+1$轮密钥所得。在最后一步，只有S-盒的代换操作而没有置换操作，然后有额外的一步混入密钥操作（称为白化），这一步主要是为了防止攻击者在不知道密钥的情况下便尝试去解密，流程如下图所示：



分组密码
Block Cipher

- 例：设$l = m = N = 4$，π_S如下定义：

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

注意这里输入z与输出$\pi_S(z)$均用十六进制表示，即

$$0000_2 = 0_{16} \to E_{16} = 1110_2$$
$$0001_2 = 1_{16} \to 4_{16} = 0100_2$$
$$\cdots$$
$$1110_2 = E_{16} \to 0_{16} = 0000_2$$
$$1111_2 = F_{16} \to 7_{16} = 0111_2$$

分组密码
Block Cipher

- π_P定义如下：

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

- 关于密码编排方案，假设是从一个32 bits的密钥流$K = (k_1, \cdots, k_{32})$产生。对于第$r$轮，定义$K^r$是由$K$中从$k_{4r-3}$开始的16个连续的比特组成，一个具体的例子如下：

$$K = 0011 \quad 1010 \quad 1001 \quad 0100 \quad 1101 \quad 0110 \quad 0011 \quad 1111$$

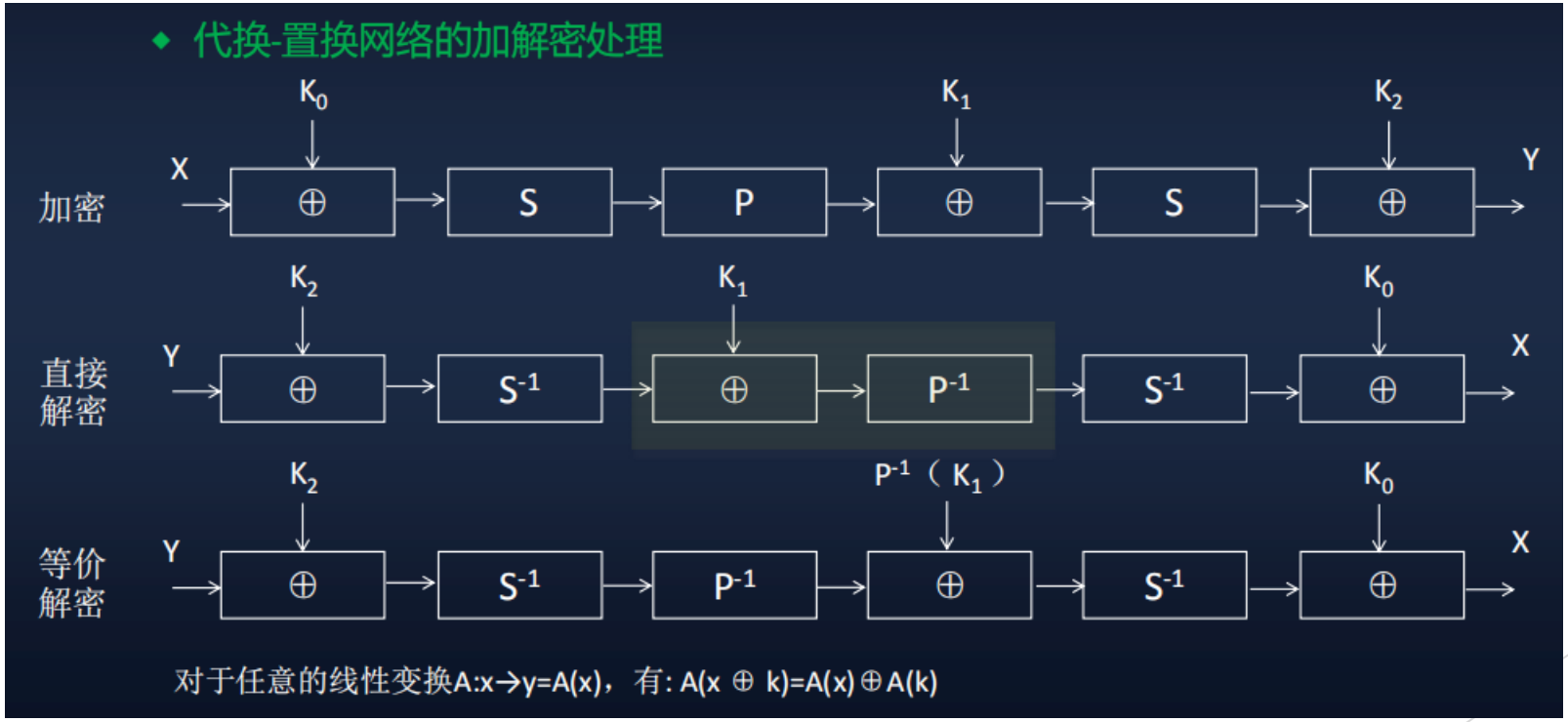
生成的轮密钥如下所示：

$$K^1 = 0011 \quad 1010 \quad 1001 \quad 0100 \quad , \quad K^2 = 1010 \quad 1001 \quad 0100 \quad 1101$$
$$K^3 = 1001 \quad 0100 \quad 1101 \quad 0110 \quad , \quad K^4 = 0100 \quad 1101 \quad 0110 \quad 0011$$
$$K^5 = 1101 \quad 0110 \quad 0011 \quad 1111$$

注意这样的密码编排方案只是为了展示，实际中一般不会采用这种不太安全的方式。

分组密码
Block Cipher

- 有了上面的铺垫，我们可以按照代换-置换网络的操作，进行加密与解密操作。



对称加密
Symmetric Encryption

- 在上面我们所介绍的几种加密算法中，加密过程所用的密钥e_K与解密过程所用的密钥d_K，或者是同一密钥，或者是由同一密钥派生的，这种情况下，我们可以认为加密钥与解密密钥相同，即$e_K = d_K$。具有这种特质的加密方式，称为对称加密。

- 对称加密方式有一个严重的制约，即物理地域分割的消息发送者与接收者，除非事前约定密钥，否则，则需要建立一个安全信道来传递密钥，不然接收者将无法对收到的加密消息进行解读。

- 对称加密的密码体系包括分组密码与流密码，上述基于代换-置换网络的分组密码，是现代密码学中重要的组成部分之一，其代表便是高级加密标准（AES），其加密与解密过程与代换-置换网络操作思路类似，只是更加复杂而已。我们不再赘述加密过程，只重点介绍一下密码编排方案。

高级加密标准
Advanced Encryption Standard

- 下面我们描述如何用128 bits的种子密钥，为10轮版本的AES构造密码编排方案。对于10轮版本的AES，需要11轮密钥，每个轮密钥由16个字节组成 。密码编排算法的处理中，一般以字为单位，由于一个字等于4个字节，因此每轮需要4个字的轮密码。这些轮密码的串接又称为扩展密码，因此，10轮版本的AES密钥最后总共需要44个字，即1408 bits。
- 对于初始密码，其为128 bits即8个字节，key[0]，key[1]，…，key[15]。对于前四个字w[0] – w[3]，有w[i] = (key[4i], key[4i+1], key[4i+2], key[4i+3])。
- 对于剩下的字，有如下操作：

$$\textbf{do}\begin{cases} temp \leftarrow w[i-1] \\ \textbf{if } i \equiv 0 \pmod 4 \\ \quad \textbf{then } temp \leftarrow \textsc{SubWord}(\textsc{RotWord}(temp)) \oplus RCon[i/4] \\ w[i] \leftarrow w[i-4] \oplus temp \end{cases}$$

其中RCon[1] – RCon[10]为常量，SubWord为字代换操作，RotWord为字轮转操作。

高级加密标准
Advanced Encryption Standard

- 根据AES加密过程中对密钥的使用方式，AES有如下工作模式：
  - 电码本模式（ECB模式）
  - 密码反馈模式（CFB模式）
  - 密码分组链接模式（CBC模式）
  - 输出反馈模式（OFB模式）
  - 计数模式
  - 计数密码分组连接模式（CCM模式）

高级加密标准
Advanced Encryption Standard

- 我们通过OpenSSL来使用AES，Windows平台可以从如下网址下载：https://kb.firedaemon.com/support/solutions/articles/4000121705
- SSL是Secure Sockets Layer（安全套接层协议）的缩写，其目标是保证两个应用间通信的保密性和可靠性，OpenSSL是对SSL的一个开源实现。
- OpenSSL整个软件包大概可以分成三个主要的功能部分：SSL协议库、应用程序以及密码算法库。作为一个基于密码学的安全开发包，OpenSSL提供的功能相当强大和全面，囊括了主要的密码算法、包括7种是分组加密算法，与一种流加密算法RC4，且支持分组密码的多种工作模式。
- 使用OpenSSL提供的对称加密功能，既可以采用基于命令行的方式，也可以采用编程的方式。下面分别来讲述。

高级加密标准
Advanced Encryption Standard

- 我们可以通过直接使用命令行的方式来使用OpenSSL。
- 列出OpenSSL支持的加密方式：

```
C:\Program Files\openssl-3\x64\bin>openssl enc -list
Supported ciphers:
-aes-128-cbc               -aes-128-cfb               -aes-128-cfb1
-aes-128-cfb8              -aes-128-ctr               -aes-128-ecb
-aes-128-ofb               -aes-192-cbc               -aes-192-cfb
-aes-192-cfb1              -aes-192-cfb8              -aes-192-ctr
-aes-192-ecb               -aes-192-ofb               -aes-256-cbc
-aes-256-cfb               -aes-256-cfb1              -aes-256-cfb8
-aes-256-ctr               -aes-256-ecb               -aes-256-ofb
-aes128                    -aes128-wrap               -aes192
-aes192-wrap               -aes256                    -aes256-wrap
```

高级加密标准
Advanced Encryption Standard

- 比如，我们选用aes-256-cbc对一个文件进行加密，即：

  openssl enc -aes-256-cbc -pass pass:123456 -in plain.txt -out encrypted.txt -p

  - -aes-256-cbc — the cipher name(symmetric cipher : AES ;block to stream conversion : CBC(cipher block chaining))
  - -pass pass:<password> — to specify the password (here password is 123456)
  - -p — Print out the salt, key and IV used.
  - -in file— input file /input file absolute path(here plain.txt)
  - -out file— output file /output file absolute path(here encrypted.txt)

高级加密标准
Advanced Encryption Standard

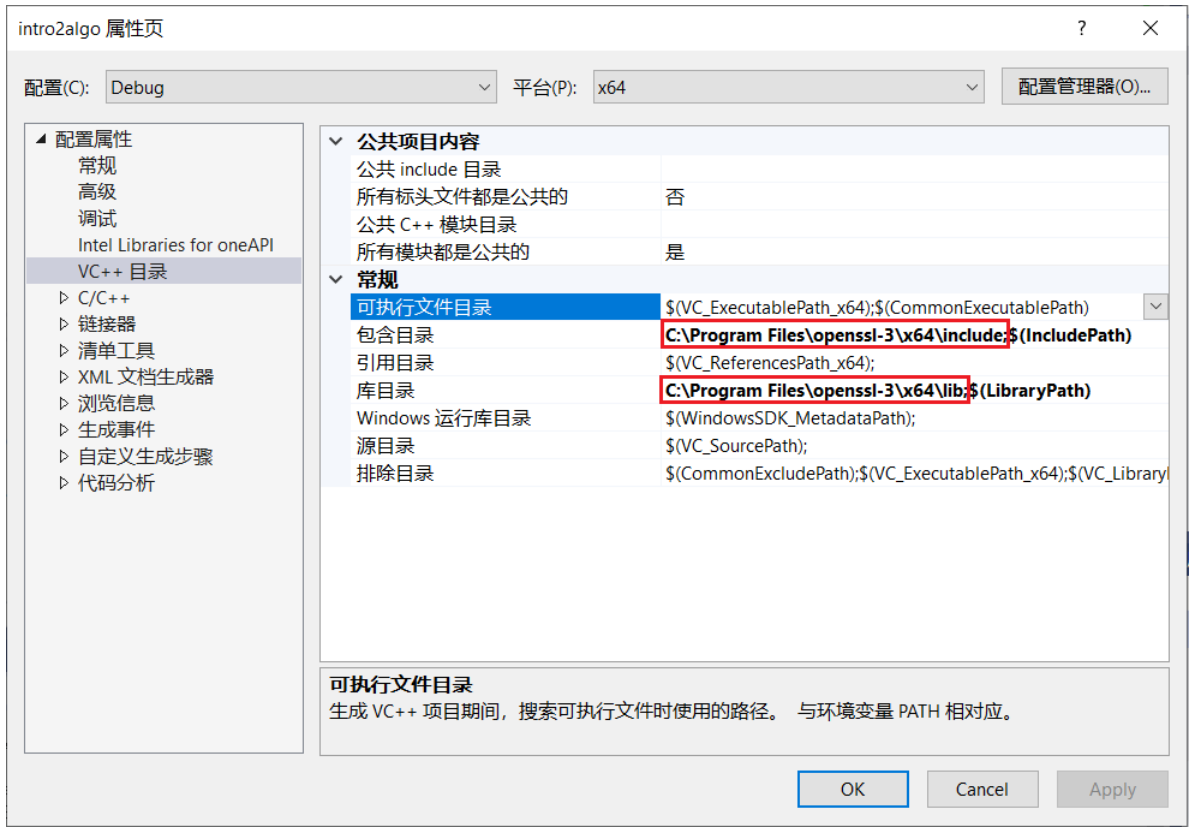
- 比如，我们选用aes-256-cbc对一个文件进行加密，即：

  openssl enc -aes-256-cbc -pass pass:123456 -d -in encrypted.txt -out decrypted.txt -p

  - -aes-256-cbc — the cipher name(symmetric cipher : AES ;block to stream conversion : CBC(cipher block chaining))
  - -pass pass:<password> — to specify the password (here password is 123456)
  - -p — Print out the salt, key and IV used.
  - -in file— input file /input file absolute path(here plain.txt)
  - -out file— output file /output file absolute path(here encrypted.txt)
  - -d — Decrypt the input data.

高级加密标准
Advanced Encryption Standard

- 当用编程实现时，注意包含适当的头文件与库：




```
#include <openssl/evp.h>
#include <openssl/aes.h>

#pragma comment(lib, "libcrypto.lib")
```

哈希函数
Hash Function

- 在网络上特别是经由不安全信道进行通信的过程中，需要一些机制对传递的消息的完整性提供验证，即接收的消息没有被中间人篡改。通过哈希函数可以达到这样的目的。
- 假定所要传递的消息x的长度记为$|x|$，令h为一个哈希函数，且$y = h(x)$。这里y被称为消息摘要或消息指纹等。一般说来，y的长度$|y| \ll |x|$，且几乎所有情况下，当x变为x'时，y将随之变为y'且$y \neq y'$。
- 一般说来，哈希函数可以带密钥，也可以不带密钥。带密钥的哈希函数通常来产生消息认证码MAC，如在数字签名中的应用。在密钥被通信双方共享的情况下，消息认证码可以在不安全通道传递。在不带密钥的情况下，消息本身与消息摘要必须被安全地存放。例如从网站上下载软件时，同时会提供消息验证码。此时一定要从官网下载。否则，假设软件被篡改，但给出的验证码也是经由被篡改的软件计算的。虽然经用户验证，软件的“完整性”没有问题，但由于软件本身已经被篡改，这种完整性也没有什么意义。

哈希函数
Hash Function

- 并非所有的函数都能做成一个Hash函数，我们首先从安全性的角度，来研究不带密钥的Hash函数需要满足的一些条件。首先，令$\mathcal{X}$表示所有可能的消息的集合，令$\mathcal{Y}$表示所有可能的消息摘要的集合。
- 原像稳固性：对给定的Hash函数h及消息摘要y，若能找到x，满足$y = h(x)$，则称x为关于y的原像，称(x, y)是关于h的有效对。不能有效找到原像问题的Hash函数称为原像稳固的。
- 第二原像稳固性：对于Hash函数h，消息x及消息摘要$y = h(x)$，若能找到x'，使$x \neq x'$但$h(x) = h(x')$，则称x'为关于y的第二原像。不能有效找到第二原像的Hash函数称为第二原像稳固的。
- 碰撞稳固性：对于Hash函数h，若能找出这样的x与x'，使得$x \neq x'$且$h(x) = h(x')$。此时，(x', y)为有效对。不能有效找出这样的x与x'的Hash函数称为碰撞稳固的。

哈希函数
Hash Function

- 由于$|y| \ll |x|$，一般$\mathcal{X} \to \mathcal{Y}$的映射是满射，即原像，第二原像，碰撞均是存在的，因此，好的哈希函数取决于解决这几个问题的各自的困难性，特别是相对困难性。
- 对任给一个函数，证明上面所提的三个问题，是十分困难的。仿分组密码的方式，实际中所用的Hash函数，都是迭代Hash函数。即我们先寻找一个基本操作构成的函数，然后反复运用这个基本函数，构造整体的Hash函数。
- 这个基本操作构成的函数一般称为压缩函数。这个compress函数为从$\{0,1\}^{m+t} \to \{0,1\}^m$的映射，这里$t \geq 1$；构造的Hash函数形式地表示为$h$: $\bigcup_{i=m+t+1}^{\infty}\{0,1\}^i \to \{0,1\}^m$，即我们可以处理无限长的输入，当然实际中的输入都是有限的。
- 下面来说明施用上面算法的一般性过程。

哈希函数
Hash Function

- 初始化阶段：对于给定的消息或输入x，用公开算法构造串z。如$z = x||\text{pad}(x)$，其中若$|x| \bmod t \neq 0$，则$\text{pad}(x)$的选择是使得$|z| \bmod t = 0$的添零操作。令$z = z_1||z_2||\cdots||z_r$，其中$|z_i| = t$。
- 处理过程：令IV为长度为m的初始串，则处理过程如下所示：

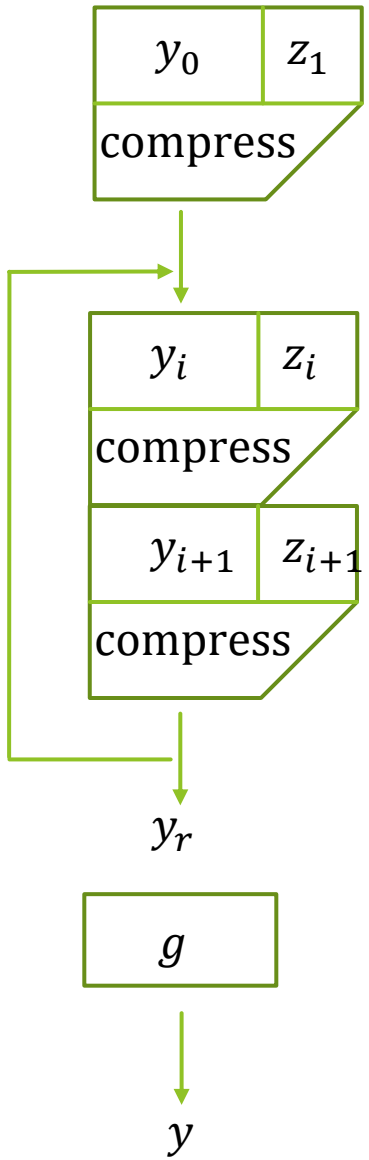
$$\begin{aligned} y_0 &\leftarrow \text{IV} \\ y_1 &\leftarrow \text{compress}(y_0||z_1) \\ &\vdots \quad \vdots \\ y_r &\leftarrow \text{compress}(y_{r-1}||z_{r-1}) \end{aligned}$$

- 输出：令$g: \{0,1\}^m \to \{0,1\}^l$为一公开函数，令$y = g(y_r)$。

- 我们不再证明，如此构造的Hash函数具有比较好的上面提到的性质。

哈希函数
Hash Function

- 下面所述过程如下图所示：



哈希函数
Hash Function

- 运用上述思想设计的典型的Hash算法有SHA-1，全名为Secure Hash Algorithm 1。其由美国国家安全局设计，并由美国国家标准技术研究所（NIST）发布为联邦数据处理标准（FIPS）。SHA-1可以生成一个长度为160位（20字节）散列值的消息摘要，散列值通常的呈现形式为40个十六进制数。
- 在后续过程中，由于发现SHA-1的一些问题，后又引入了SHA-2，包括SHA-224，SHA-256，SHA-384与SHA-512等。
- 最新的SHA标准是SHA-3，其设计方式与前面迭代Hash函数结构不太一样，其基于称为海绵结构的方式进行设计（Sponge Construction）。

- 这些SHA标准在OpenSSL中均提供了支持。

哈希函数
Hash Function

- 运用OpenSSL对指定的文件计算摘要的一个例子如下所示：

```
C:\Users\MSUser\working-files>openssl dgst -sha256 utils.py
SHA256(utils.py)= 825e3faddaeac7bc64a9b38a2ba1f5fbbbadf8fdc8e9d43b0c4c421b14487b77
```

- 相同的功能亦可通过编程的方法实现，注意包含相应的头文件与库。

```
#include <openssl/evp.h>
#pragma comment(lib, "libcrypto.lib")
```