# 现代密码学理论与实践
# 6 其它加密技术

苗付友，黄文超
主页：http://staff.ustc.edu.cn/~huangwc/crypto.html

mfy@ustc.edu.cn，huangwc@ustc.edu.cn

huangwc@ustc.edu.cn

# 本章要点

- 多重加密是将一个加密算法多次使用的技术，明文通过加密算法转化为密文，然后将该密文作为输入重新执行加密算法，该过程可以重复多次。
- 三重DES(3DES)在三个阶段使用DES算法，共用到两组或三组密钥。
- 选择工作模式是一项增强密码算法或者使算法适应具体应用的技术。
- 对称密码有5种工作模式，电码本模式、密文分组链接模式、密文反馈模式、输出反馈模式和计数器模式。
- 流密码是一种对称密码算法，其输出密文是由输入明文逐位或者逐字节产生的，RC4是应用最广泛的一种流密码。

huangwc@ustc.edu.cn

# 本章内容

1. 多重加密与三重DES算法
2. 分组密码的工作模式
3. 流密码和RC4

huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法

## 动机

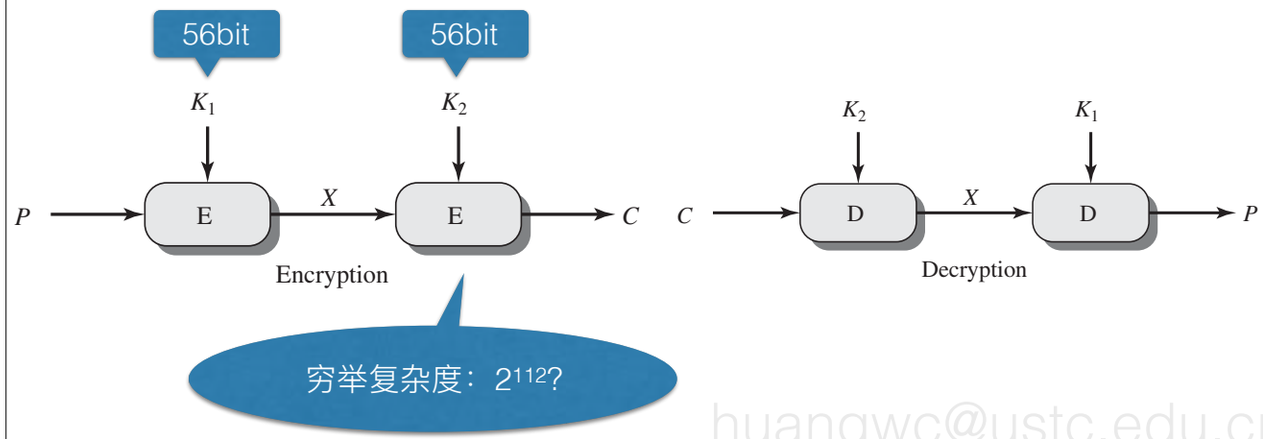
- DES的问题 ——穷举攻击
- 一种低成本的方案：
  - 使用已有的DES进行多次加密

huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法

## 双重DES

- 多次加密的**最简单形式**是进行两次加密，每次使用不同的密钥
- 加密：$C = \mathrm{E}(K_2, \mathrm{E}(K_1, P))$
- 解密：$P = \mathrm{D}(K_1, \mathrm{D}(K_2, C))$




huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法

## 双重DES

- 中间相遇攻击 “meet-in-the-middle attack”

$$\text{2DES}(K_1\|K_2, M) = \text{DES}(K_2, \text{DES}(K_1, M))$$
$$\text{2DES}^{-1}(K_1\|K_2, C) = \text{DES}^{-1}(K_1, \text{DES}^{-1}(K_2, C))$$
$$\text{DES}^{-1}(K_2, C_1) = \text{DES}(K_1, M_1)$$

- 给定明文对（M,C）
  - 用所有可能的密钥加密明文M并把结果**按顺序（或Hash）**存储起来
  - 然后用所有可能的密钥解密密文C，寻找匹配的中间值
  - 复杂度$2^{57}$

huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法

## 双重DES

- 中间相遇攻击 “meet-in-the-middle attack”

$MinM_{\mathsf{2DES}}(M_1, C_1)$
**for** $i = 1, \ldots, 2^{56}$ **do** $L[i] \leftarrow \mathrm{DES}(T_i, M_1)$
**for** $j = 1, \ldots, 2^{56}$ **do** $R[j] \leftarrow \mathrm{DES}^{-1}(T_j, C_1)$
$S \leftarrow \{\, (i, j) \,:\, L[i] = R[j] \,\}$
Pick some $(l, r) \in S$ and **return** $T_l \| T_r$

For any $(i, j) \in S$ we have

$$\mathrm{DES}(T_i, M_1) = L[i] = R[j] = \mathrm{DES}^{-1}(T_j, C_1)$$

- 使用n个(M,C)对时，密钥空间$2^{112}$，实际映射空间$2^{64*n}$
- 当有两个已知明文时，出错概率$2^{112\text{-}128}$

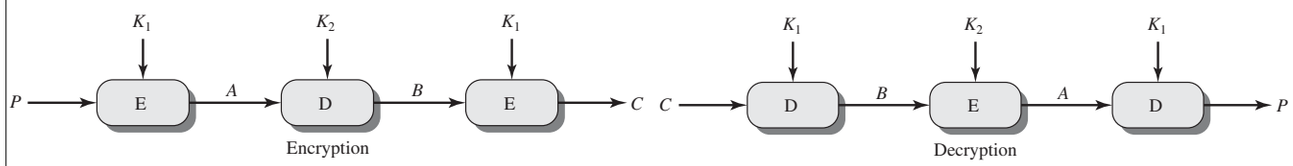
huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法
# 三重DES算法(2个密钥)

- 使用两个密钥

$$C = \mathrm{E}(K_1, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

$$P = \mathrm{D}(K_1, \mathrm{E}(K_2, \mathrm{D}(K_1, C)))$$



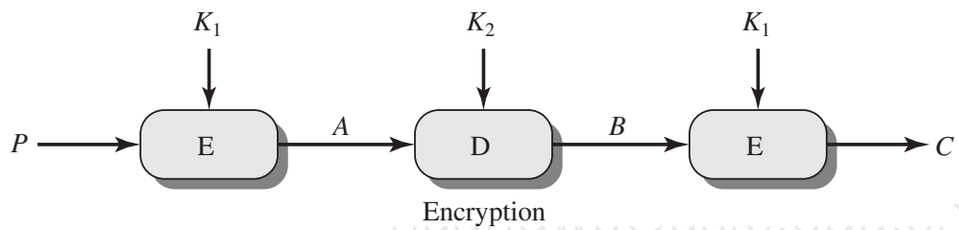

- 目前，没有针对3DES的可行攻击
- 已被用于密钥管理标准ANSI X9.17和ISO8732

huangwc@ustc.edu.cn

# 1. 多重加密与三重DES算法

## 三重DES算法(2个密钥)

- 攻击建议1
  - 选择明文$2^{56}$
    - For $K_1$=0~$2^{56}$
      - 设A=0, 由$K_1$, 求出P
      - 求出C, 求出B
    - For $K_2$=0~$2^{56}$
      - 设A=0, 由$K_2$, 求出B

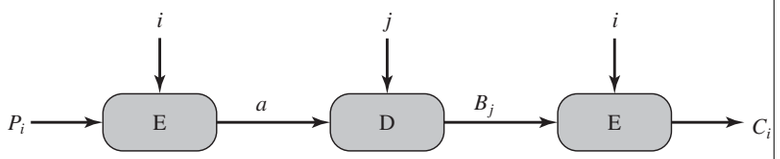


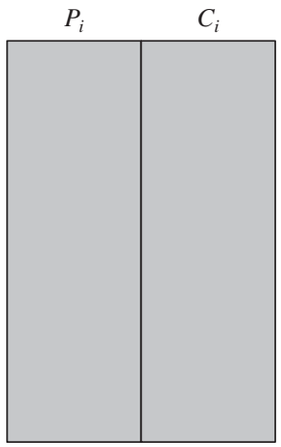
# 1. 多重加密与三重DES算法

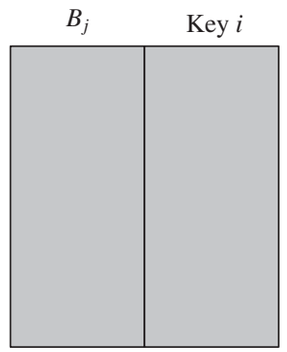
## 三重DES算法(2个密钥)

- 攻击建议2-已知明文攻击
  - 但代价增多

复杂度： $\left(2^{56}\right)\frac{2^{64}}{n} = 2^{120-\log_2 n}$




- 1. 构建排过序的(P,C)表1
- 2.1. **任取值a**, 遍历i,求

$$P_i = \mathrm{D}(i, a)$$

- 2.2. 如果表1有Pi, 则求

$$B = \mathrm{D}(i, C)$$

- 2.3. 将B加入表2，并按B排序
- 3.1 遍历j, 求

$$B_j = \mathrm{D}(j, a)$$

- 3.2 查表2，如果找到匹配，则 S+={ i || j }

# 1. 多重加密与三重DES算法
# 三重DES算法(3个密钥)

- 2个密钥的三重DES算法总让人不放心
  - 另一种方案：三个密钥

$$C = \mathrm{E}(K_3, \mathrm{D}(K_2, \mathrm{E}(K_1, P)))$$

- PGP
- S/MIME

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式

| 模式 | | 描述 | 典型应用 |
|---|---|---|---|
| 电码本（ECB）<br>Electronic CodeBook | | $\text{cipher} = \text{out}$<br>$\text{in} = \text{plain}$ | 传**单个**数据，如密钥 |
| 密文分组链接 (CBC)<br>Cipher Block Chaining | | $\text{cipher} = \text{out}$<br>$\text{in} = \text{plain} \oplus \text{out}_{\text{prev}}$ | 面向分组的通用传输；<br>认证 |
| 密文反馈 (CFB)<br>Cipher FeedBack | out取<br>s位 | $\text{cipher} = \text{out} \oplus \text{plain}$<br>$\text{in} = \text{cipher}_{\text{prev}} \oplus \text{shift}$ | 面向分组的通用传输；<br>认证 |
| 输出反馈(OFB)<br>Output FeedBack | out取<br>整个分组 | $\text{cipher} = \text{out} \oplus \text{plain}$<br>$\text{in} = \text{out}_{\text{prev}}$ | **噪声**信道的传输 |
| 计数器 (CTR)<br>Counter | | $\text{cipher} = \text{out} \oplus \text{plain}$<br>$\text{in} = \text{counter}$ | 面向分组的通用传输；<br>**高速**要求 |

huangwc@ustc.edu.cn

\textrm{in} = \textrm{plain} \oplus \textrm{out}_{\textrm{prev}}
\textrm{cipher} = \textrm{out} \oplus \textrm{plain}
\textrm{in} = \textrm{cipher}_{\textrm{prev}}
\textrm{in} = \textrm{counter}
\textrm{in} = \textrm{cipher}_{\textrm{prev}} \oplus \textrm{shift}
\textrm{in} = \textrm{cipher}_{\textrm{prev}}

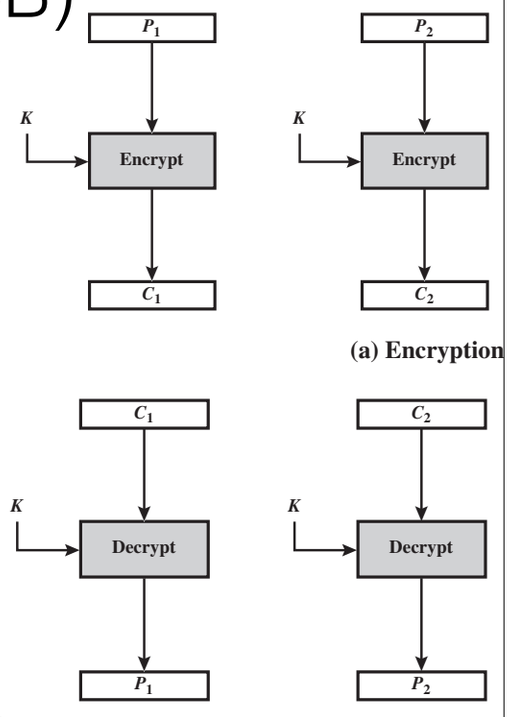
# 2. 分组密码的工作模式

## 电码本 (ECB)

- 加密流程

  $\text{cipher} = \text{out} \qquad \text{in} = \text{plain}$

- 特点
  - 适合数据较少的情况
    - 例：加密密钥
  - 长消息则可能不安全
    - 辨认重复的消息

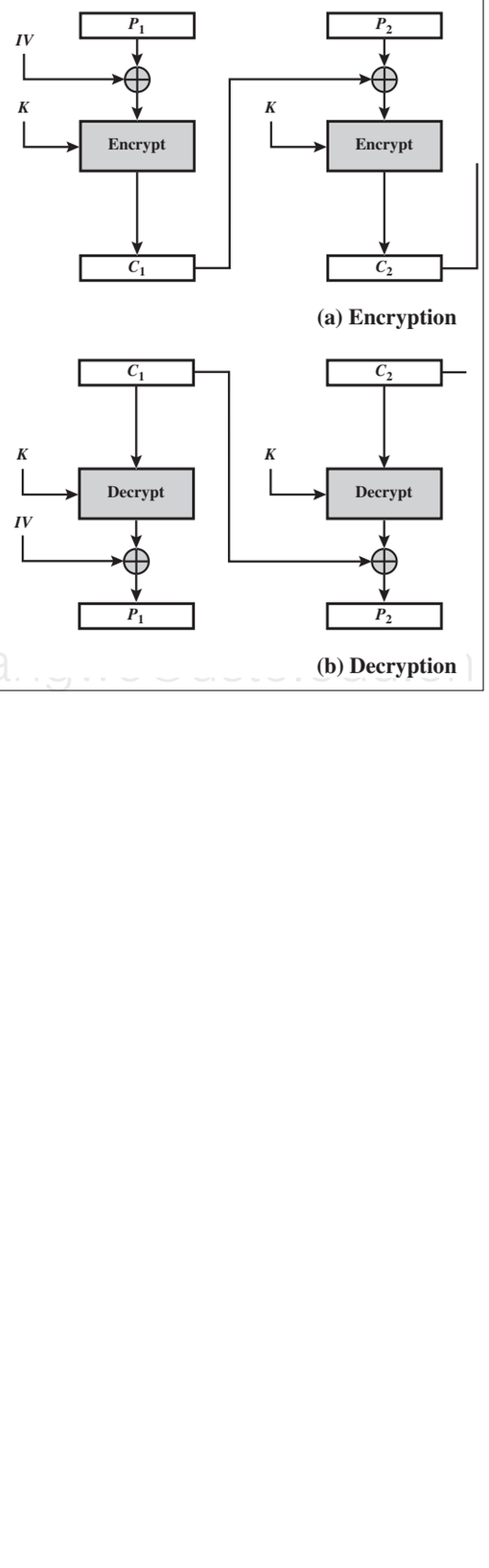



hua

# 2. 分组密码的工作模式
# 密文分组链接 (CBC)

- 加密流程

$$\text{cipher} = \text{out} \qquad \text{in} = \text{plain} \oplus \text{out}_{\text{prev}}$$

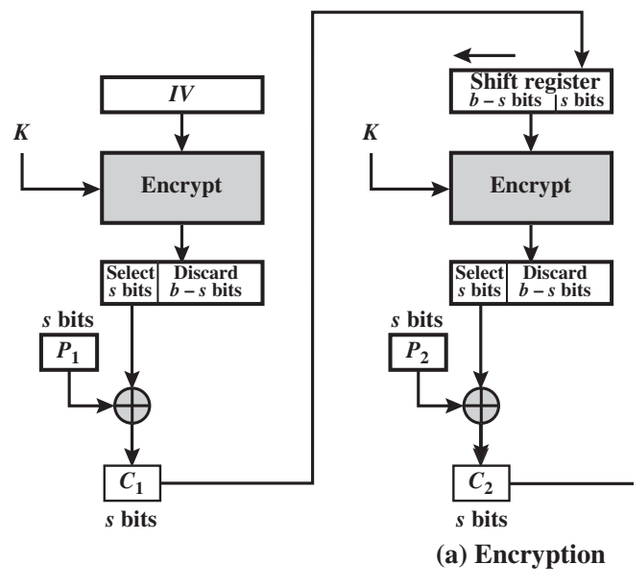



- 特点
  - 解决ECB的缺陷

# 2. 分组密码的工作模式
# 密文反馈 (CFB)

- 加密流程

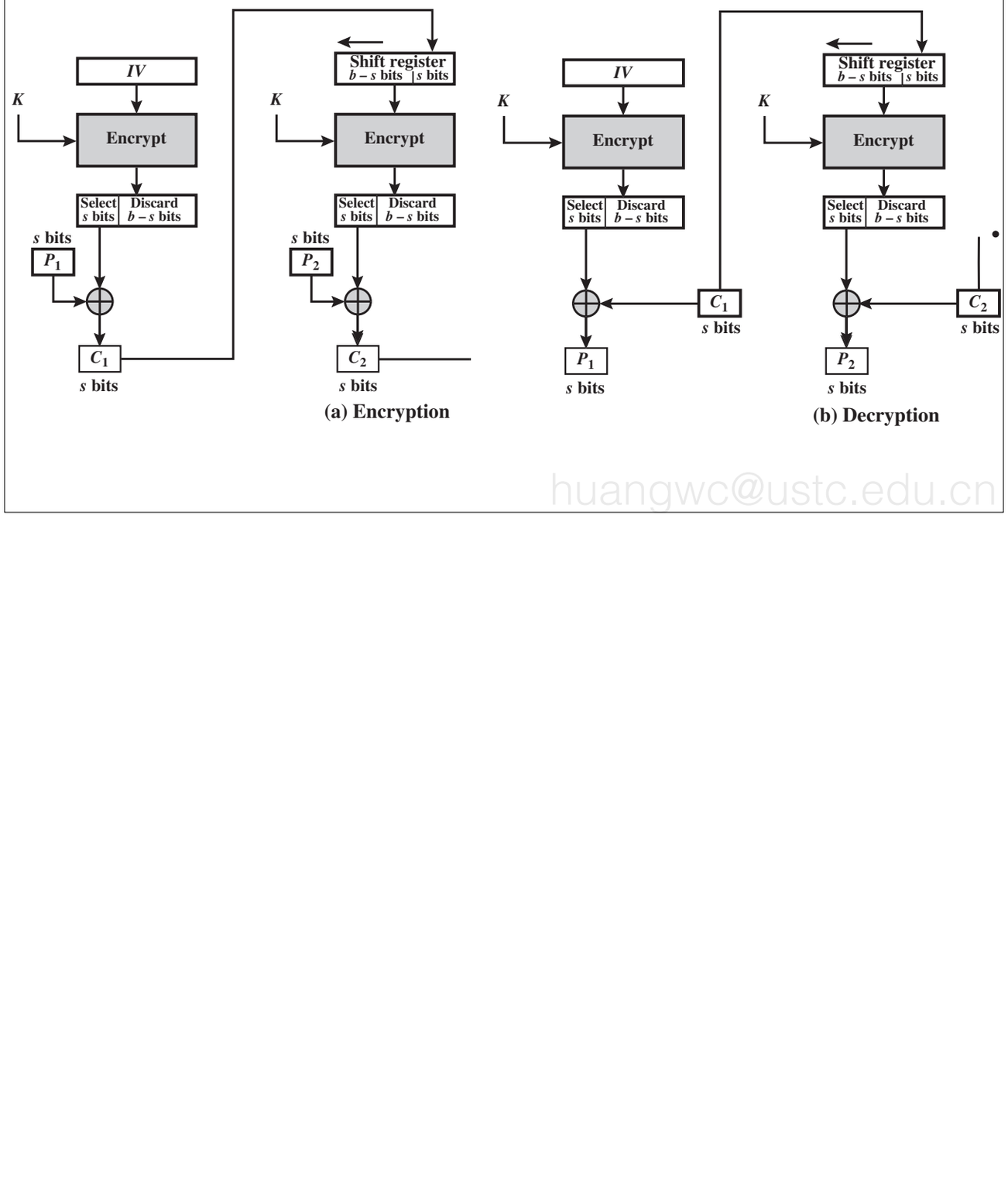
$$\text{cipher} = \overset{\text{取s位}}{\text{out}} \oplus \text{plain}$$
$$\text{in} = \text{cipher}_{\text{prev}} \oplus \text{shift}$$




- 特点
  - 可视为流密码
    - 节省传输能力

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 密文反馈 (CFB)

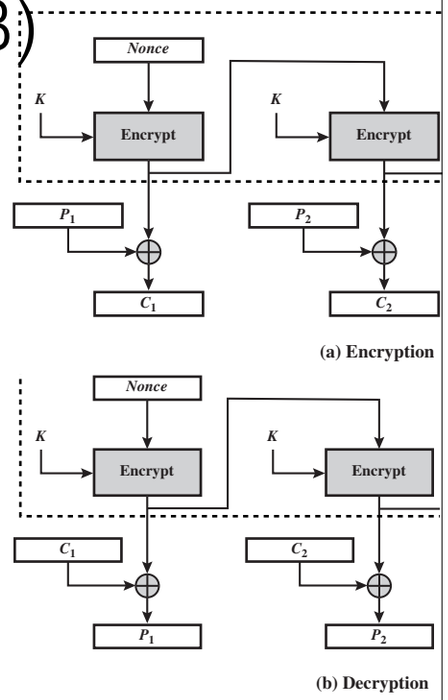



huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 输出反馈(OFB)

- 加密流程

$$\text{cipher} = \text{out} \oplus \text{plain}$$
$$\text{in} = \text{out}_{\text{prev}}$$




- 特点
  - 与CFB相似
  - 传输时某位错误不会影响其他位
  - 抗消息流篡改攻击不如CFB

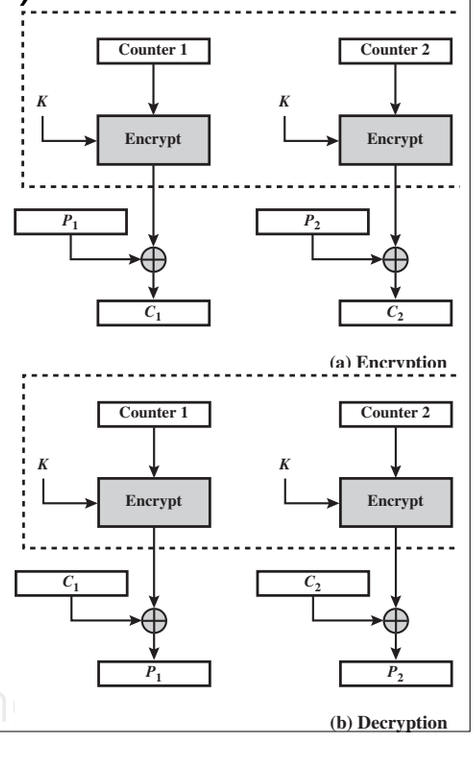
huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 计数器 (CTR)

- 加密流程

$$\text{cipher} = \text{out} \oplus \text{plain}$$
$$\text{in} = \text{counter}$$

- 特点
  - 并行性、预处理、随机读写
  - 可证明安全
  - 简单性



# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- 性能指标
  - 秘密性
  - 高效随机访问
  - 存储占用

huangwc@ustc.edu.cn

Disk encryption methods aim to provide three distinct properties:

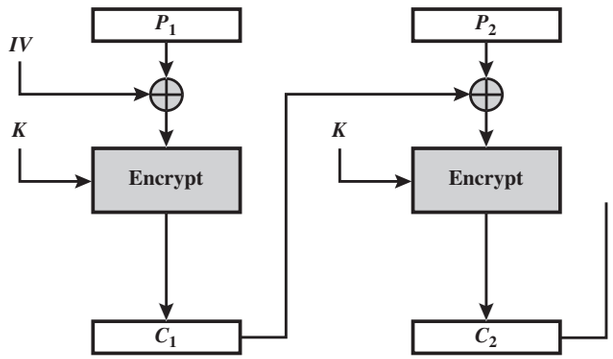
The data on the disk should remain confidential.
Data retrieval and storage should both be fast operations, no matter where on the disk the data is stored.
The encryption method should not waste disk space (i.e., the amount of storage used for encrypted data should not be significantly larger than the size of plaintext).

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

$\text{cipher} = \text{out} \qquad \text{in} = \text{plain} \oplus \text{out}_{\text{prev}}$

- CBC的缺陷
  - 初始值IV可预测
    - 敌方可构建水印 (watermark)

$$b_1 \oplus IV_1 = b_2 \oplus IV_2$$




**(a) Encryption**

huangwc@ustc.edu.cn

CBC suffers from some problems. For example, if the IVs are predictable, then an adversary may leave a "watermark" on the disk, i.e., store a specially created file or combination of files which is identifiable even after encryption. The exact method of constructing the watermark depends on the exact function providing the IVs, but the general recipe is to create two encrypted sectors which have identical first blocks b 1 \scriptstyle b_{1} and b 2 \scriptstyle b_{2}; these two are then related to each other by b 1 ⊕ I V 1 = b 2 ⊕ I V 2 \scriptstyle b_{1}\,\oplus \,IV_{1}\;=\;b_{2}\,\oplus \,IV_{2}. Thus the encryption of b 1 \scriptstyle b_{1} is identical to the encryption of b 2 \scriptstyle b_{2}, leaving a watermark on the disk. The exact pattern of "same-different-same-different" on disk can then be altered to make the watermark unique to a given file.

To protect against the watermarking attack, a cipher or a hash function is used to generate the IVs from the key and the current sector number, so that an adversary cannot predict the IVs. In particular, the ESSIV approach uses a block cipher in CTR mode to generate the IVs.

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- Encrypted salt-sector initialization vector (ESSIV)
  - 用于生成IV
    - SN=Sector Number
    - s=hash(K)

$$IV(SN) = E_s(SN)$$

- Malleability attack
  - 敌方已知明文
  - 有可能破坏完整性：每隔1个分组可以被赋予指定数据

huangwc@ustc.edu.cn

ESSIV[1] is a method for generating initialization vectors for block encryption to use in disk encryption. The usual methods for generating IVs are predictable sequences of numbers based on, for example, time stamp or sector number, and permits certain attacks such as a watermarking attack. ESSIV prevents such attacks by generating IVs from a combination of the sector number SN with the hash of the key. It is the combination with the key in form of a hash that makes the IV unpredictable.

Malleability attack[edit]
While CBC (with or without ESSIV) ensures confidentiality, it does not ensure integrity of the encrypted data. If the plaintext is known to the adversary, it is possible to change every second plaintext block to a value chosen by the attacker, while the blocks in between are changed to random values. This can be used for practical attacks on disk encryption in CBC or CBC-ESSIV mode.[4]

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES
  - XEX-based tweaked-codebook mode with ciphertext stealing
  - 著名应用
    - Mac OS X Lion's FileVault 2
    - Windows 10's BitLocker等

huangwc@ustc.edu.cn

XTS is supported by BestCrypt, Botan, dm-crypt, FreeOTFE, TrueCrypt, VeraCrypt,[12] DiskCryptor, FreeBSD's geli, OpenBSD softraid disk encryption software, OpenSSL, Mac OS X Lion's FileVault 2, and Windows 10's BitLocker.[13]

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES
  - 攻击者模型：攻击者可以随意获取密文，如
    - 数据库的密文记录的访问
    - 非授权用户对加密记录的访问
    - 被偷的磁盘的数据访问

huangwc@ustc.edu.cn

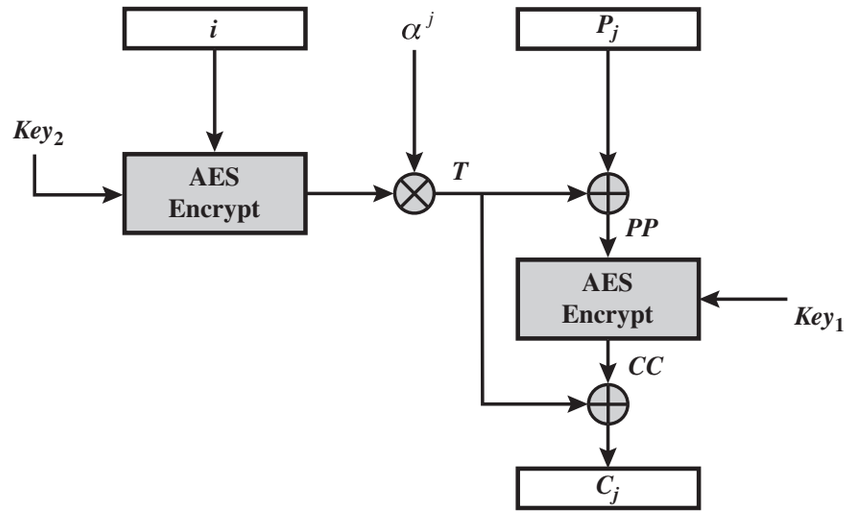
# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 要求：
  - 密文应该与明文大小相同，从而不改变数据的布局
  - 数据分组可单独访问，且相互**独立**
  - 加密分组大小为16字节
  - 没有使用其他元数据（除了数据分组的位置）
  - 不同位置 相同明文 ==》密文不同
  - 符合标准的加解密设备可以相互**通用**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密

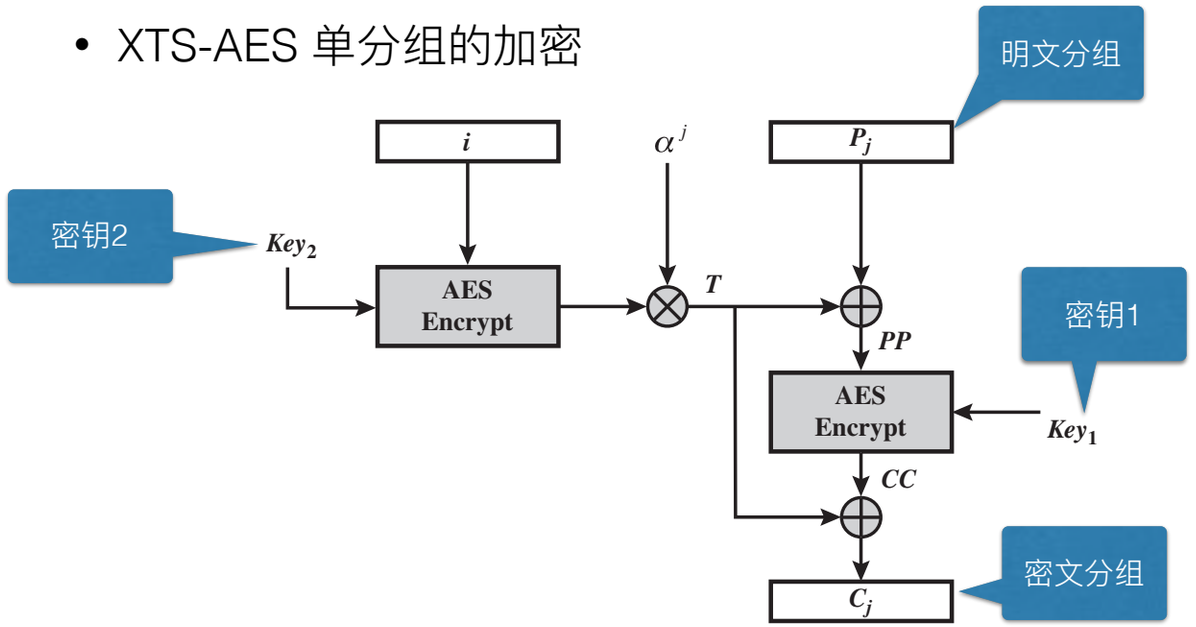



**(a) Encryption**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密



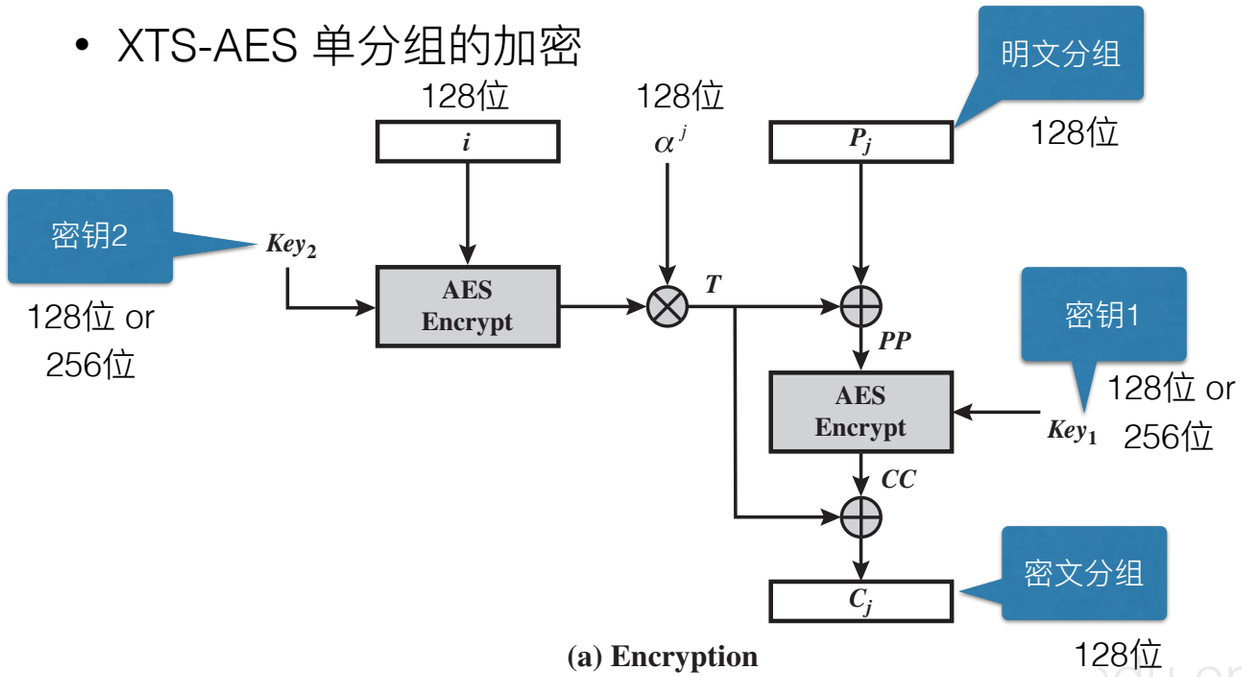

**(a) Encryption**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式

# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密

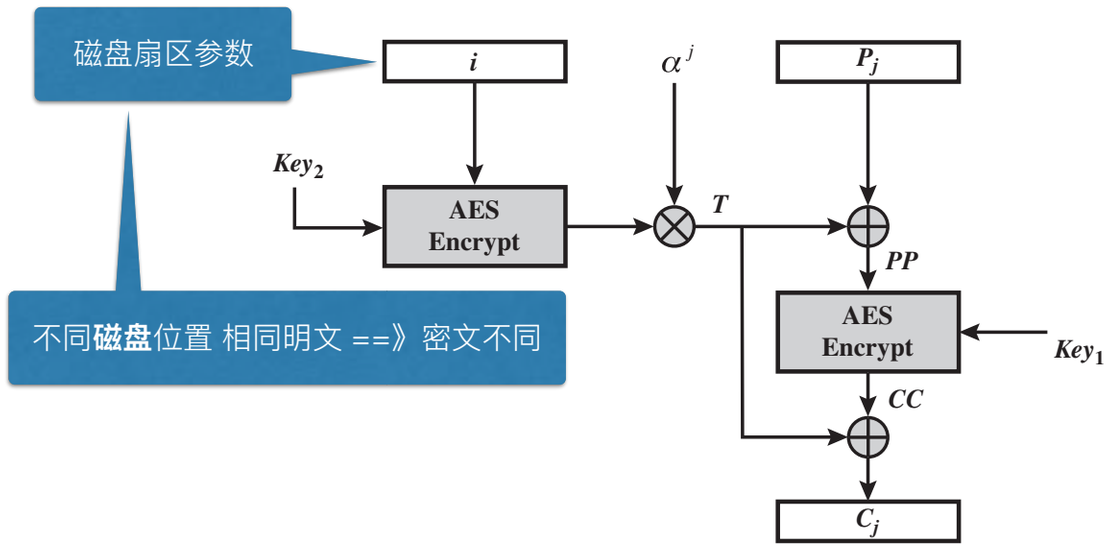



(a) Encryption

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密



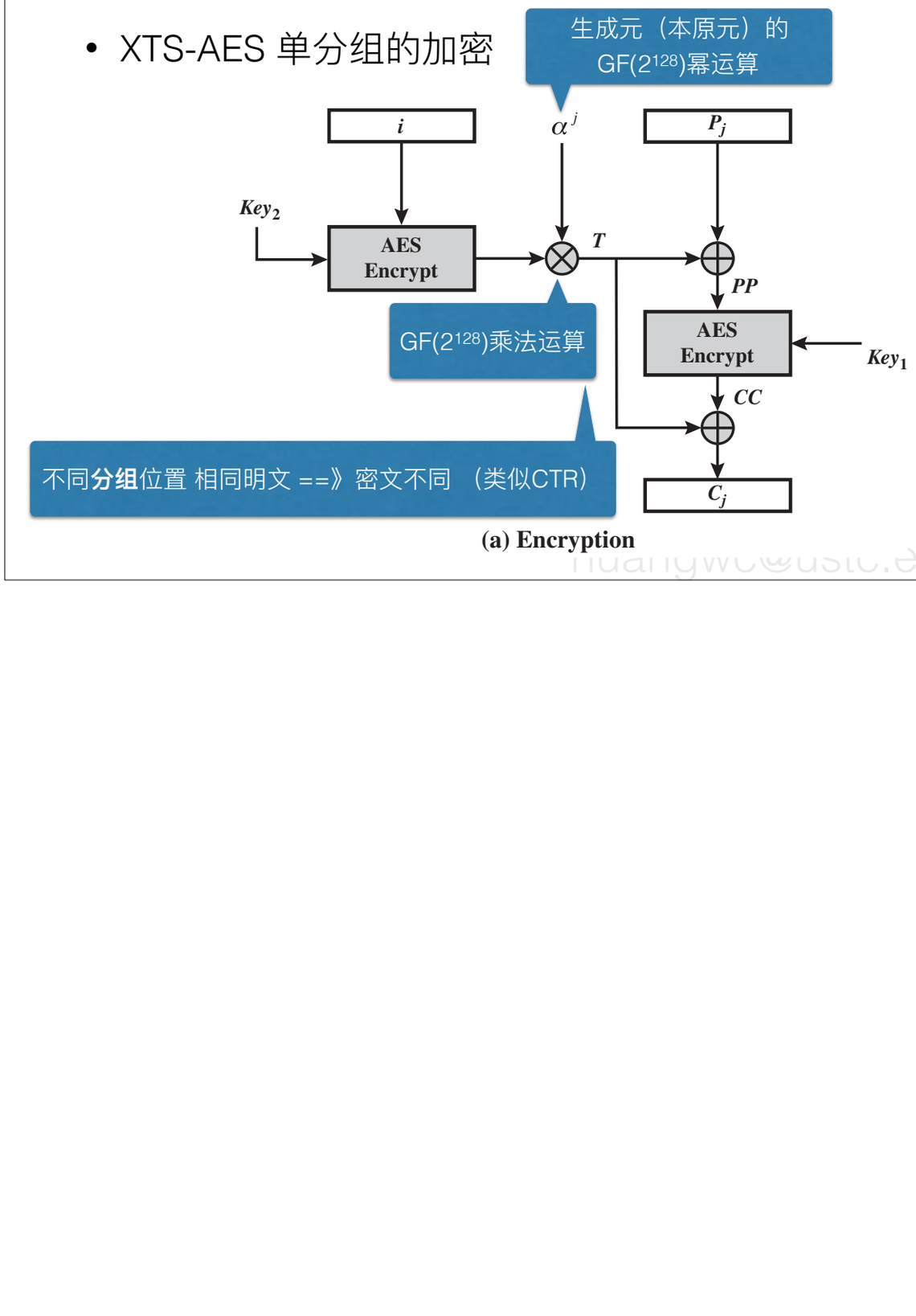

**(a) Encryption**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式

# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密

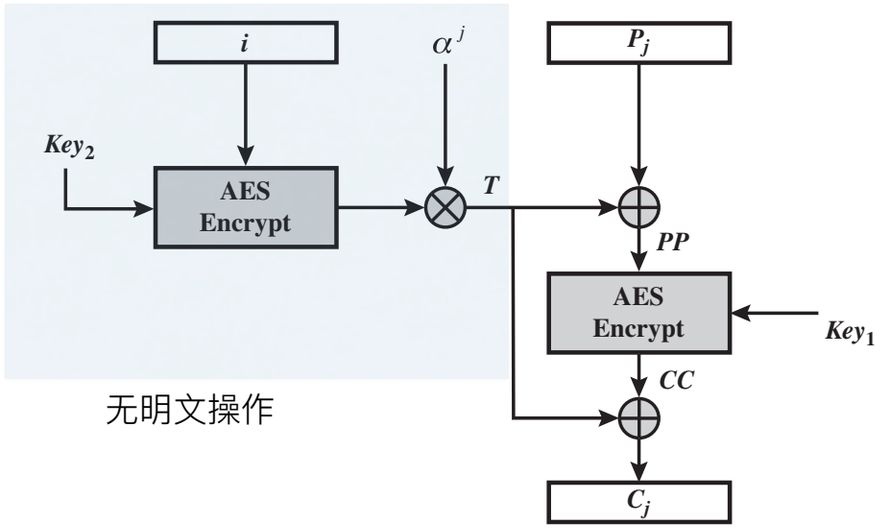



**(a) Encryption**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的加密

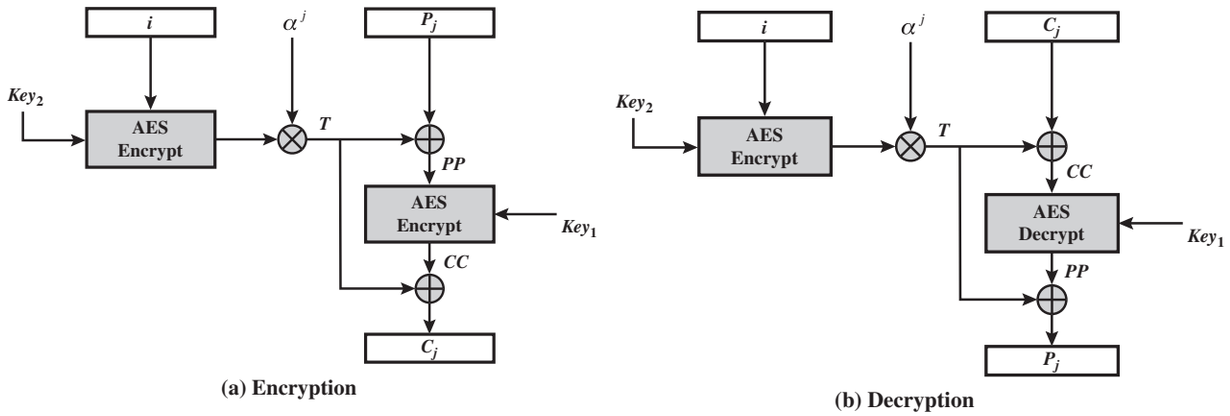



**(a) Encryption**

huangwc@ustc.edu.cn

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 单分组的运算




huangwc@ustc.edu.cn

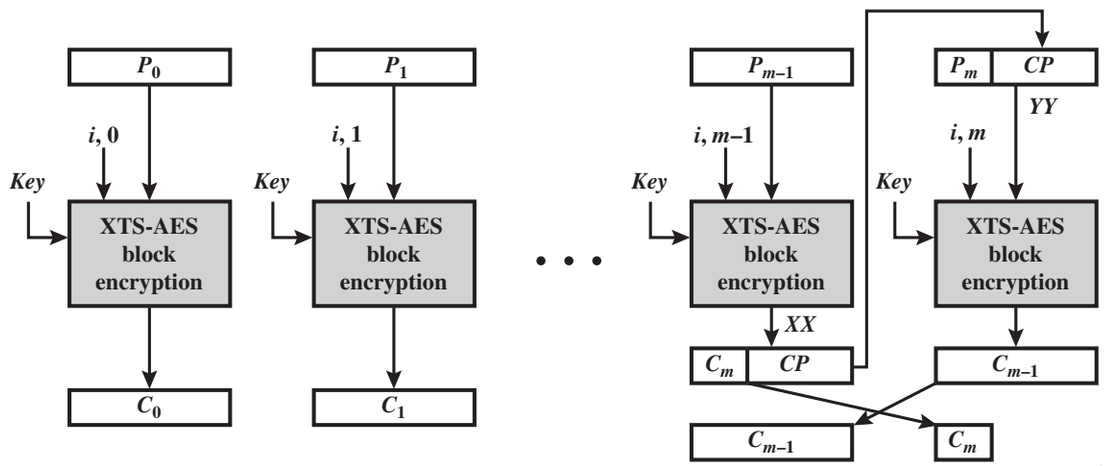
# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 在扇区上的加密

i:扇区参数（Tweaked），j：分组序号

Block encryption: XTS-AES-blockEnc($K$, $P_j$, $i$, $j$)

Block decryption: XTS-AES-blockDec($K$, $C_j$, $i$, $j$)

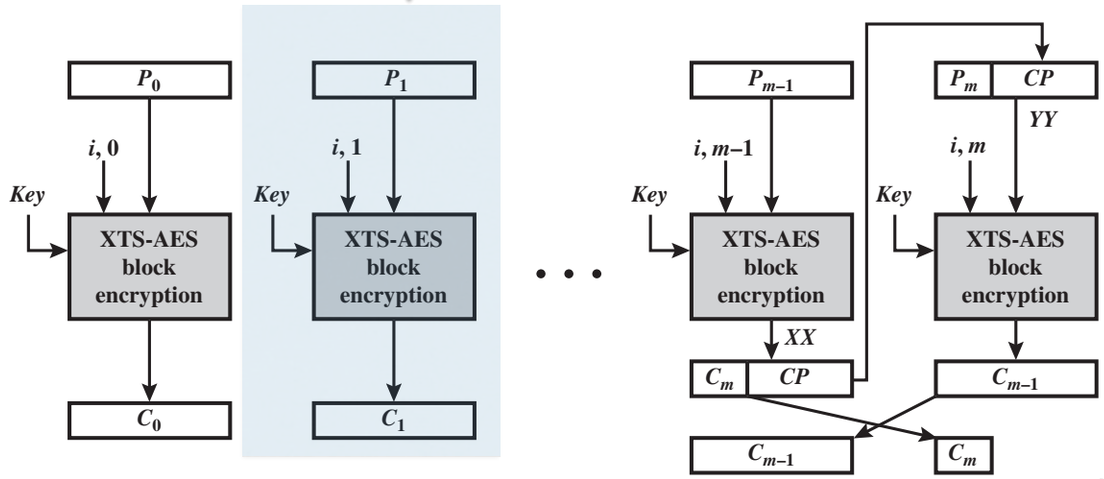



(a) Encryption

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

- XTS-AES 在扇区上的加密

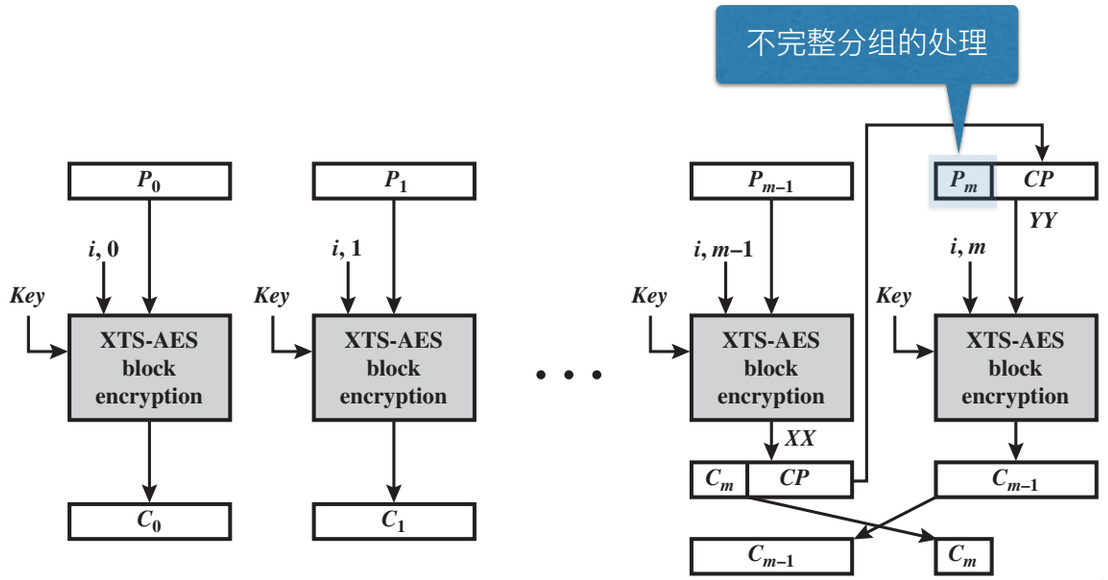
每一块加密过程相互独立，适于并行




(a) Encryption

# 2. 分组密码的工作模式
# 磁盘加密 (Disk Encryption)

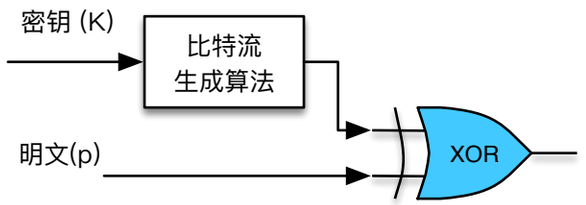
- XTS-AES 在扇区上的加密




(a) Encryption

# 3. 流密码和RC4

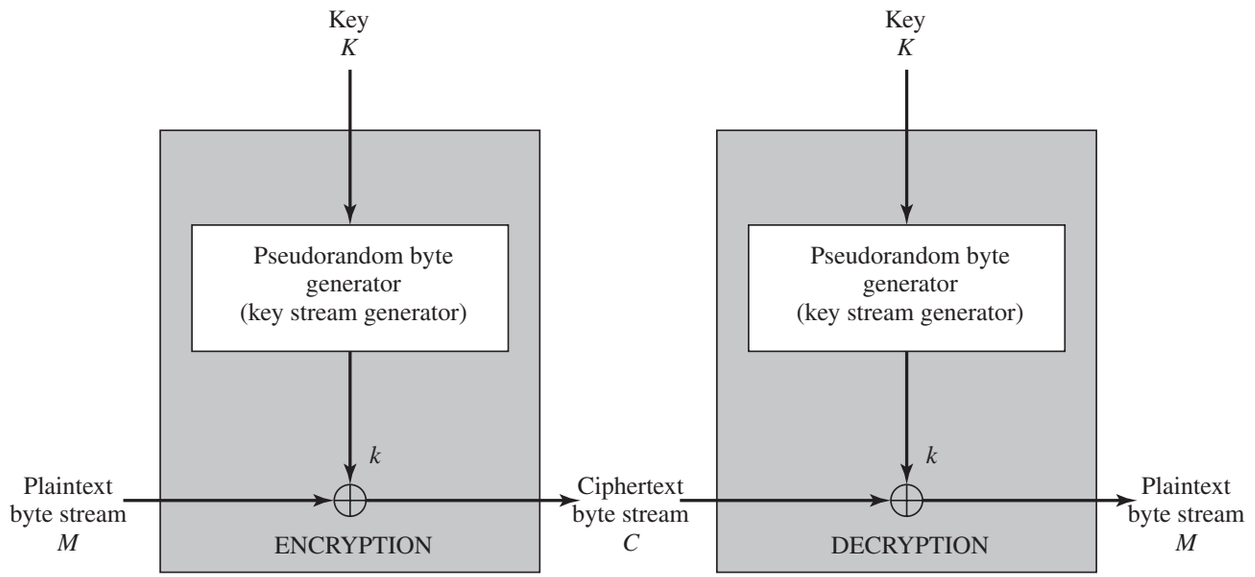
- 特点
  - 按位处理明文消息(as a stream)
  - 典型做法是用一个伪随机流密钥与明文按位异或
  - 流密钥的随机性完全扰乱了明文消息的统计特性
  - 不能重复使用流密钥，不然系统可能被破解




流密码

huangwc@ustc.edu.cn

# 3. 流密码和RC4




huangwc@ustc.edu.cn

# 3. 流密码和RC4

## 流密码设计方面的一些考虑

- 无重复的长周期
- 统计上是随机的
- 依赖于足够长的密钥
- 线性复杂性要大
- 对于相关性要有免疫性
- 扰乱
- 扩散
- 采用高度非线性的布尔函数

huangwc@ustc.edu.cn

# 3. 流密码和RC4

## RC4算法

- 由RSA在1987提出，Ron Rivest设计
- 简单且高效
- 周期大于$10^{100}$
- 密钥长度可变，面向字节的流密码
- 使用广泛(web SSL/TLS, wireless WEP， WPA)
- 对所有8位的值以密钥实现随机置换
- 用置换加密输入消息，每次一个字节

huangwc@ustc.edu.cn

# 3. 流密码和RC4

## RC4算法

- 总体思路
  - 密钥：可变长度密钥(1~256字节)
  - 流程
    1. 初始化状态向量S
       - S有256字节，记为S[0], S[1], … S[255]
    2. 生成密钥流
       - S数组内部字节置换
       - 由S生成密钥流

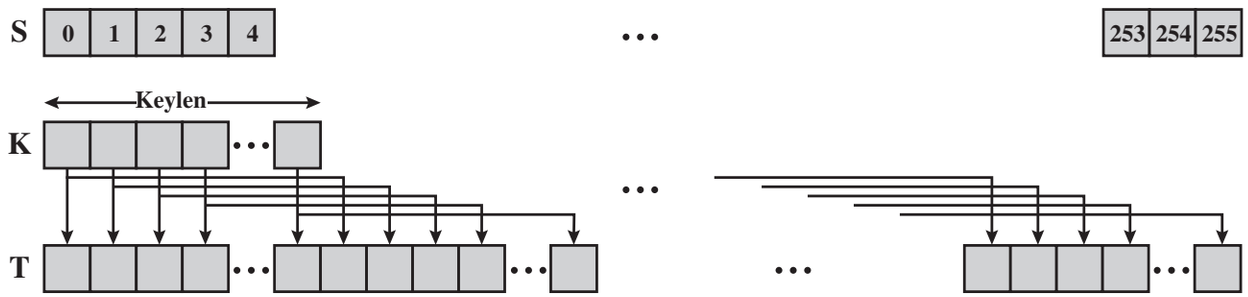
huangwc@ustc.edu.cn

# 3. 流密码和RC4

## RC4算法

1. 初始化状态向量S

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```




huangwc@ustc.edu.cn

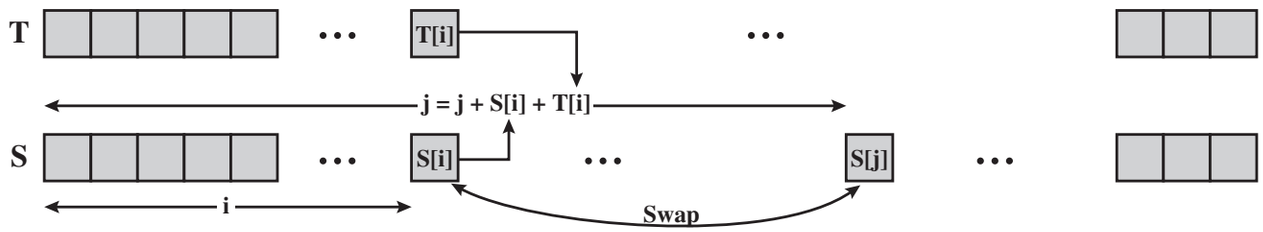
# 3. 流密码和RC4

## RC4算法

1. 初始化状态向量S

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
```



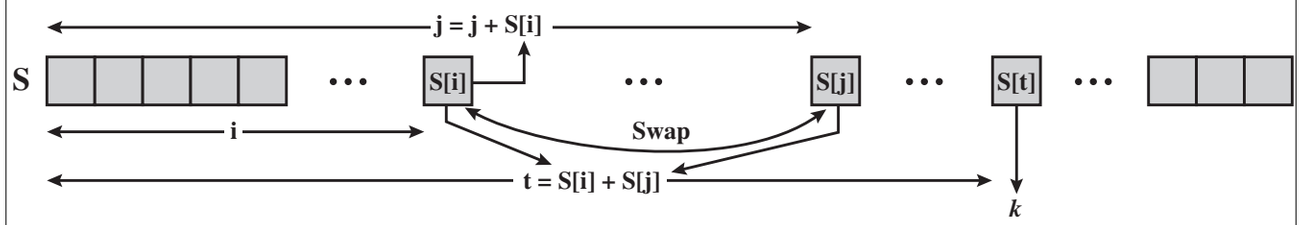

huangwc@ustc.edu.cn

# 3. 流密码和RC4

## RC4算法

2. 密钥流的生成

```
/* Stream Generation */
i, j = 0;
while (true)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```




huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- 可以抵御已知的各种攻击？— Usenix Security ’15
- 加密结果是非常“非线性”的
- RC4是流密码，决不重复使用密钥
- 用于为802.11无线局域网提供安全性的WEP协议
  - WEP易受一种特殊攻击，问题不在RC4本身，而是RC4中输入的密钥的产生途径有漏洞

huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- 应用
  - 1997 WEP
  - 2003/2004 WPA
  - 1995 SSL
  - 1999 TLS
- 禁止使用
  - 2015 TLS

huangwc@ustc.edu.cn

RC4 became part of some commonly used encryption protocols and standards, such as WEP in 1997 and WPA in 2003/2004 for wireless cards; and SSL in 1995 and its successor TLS in 1999, until it was prohibited for all versions of TLS by RFC 7465 in 2015, due to the RC4 attacks weakening or breaking RC4 used in SSL/TLS. The main factors in RC4's success over such a wide range of applications have been its speed and simplicity: efficient implementations in both software and hardware were very easy to develop.

# 3. 流密码和RC4
# RC4算法的安全性

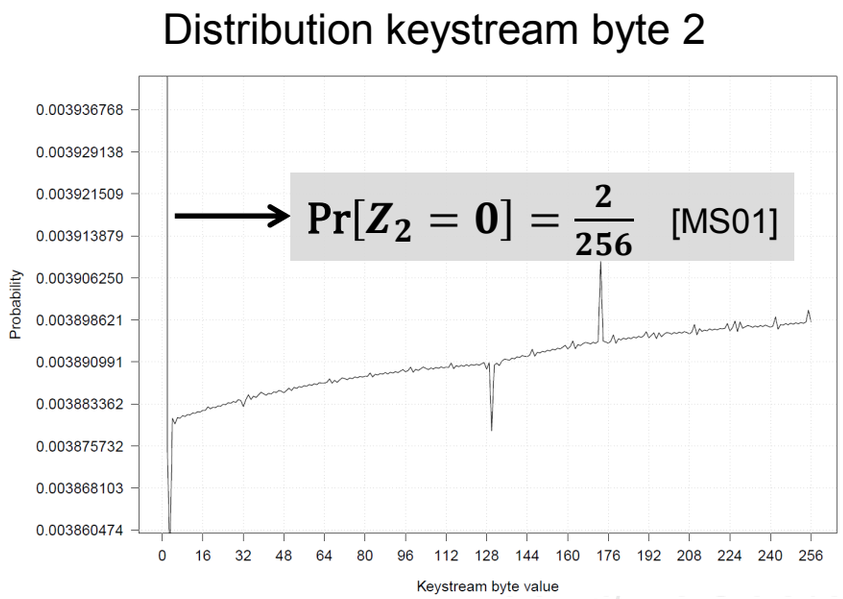
- 理论攻击
  - 问题：非平均分布
  - Fluhrer-McGrew biases
  - Mantin's ABSAB biases

huangwc@ustc.edu.cn

The first one is that two consecutive bytes are biased towards certain values. These are commonly called the Fluhrer-McGrew biases.
The second type of biases is that a pair of consecutive bytes is likely to repeat itself. These are called the Mantin's ABSAB biases.

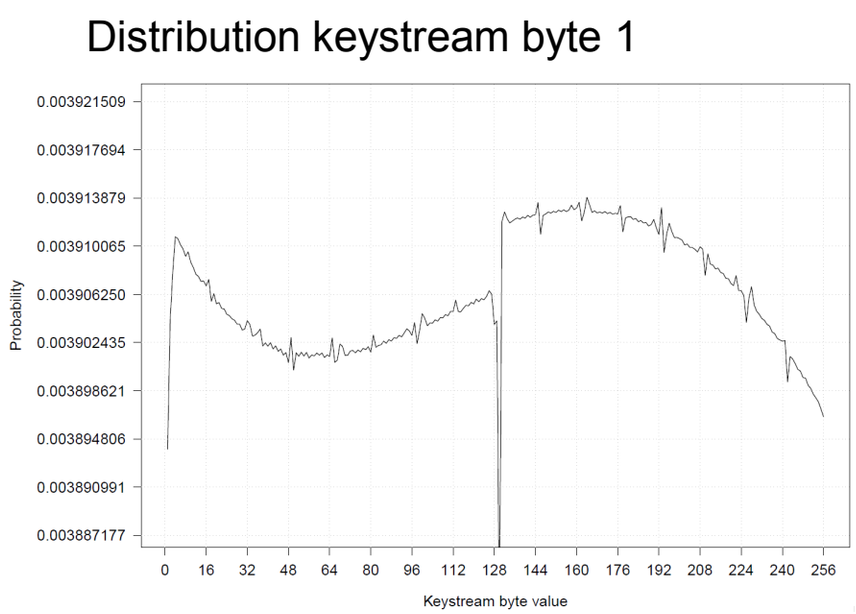
# 3. 流密码和RC4
# RC4算法的安全性

- 问题：Short-Term Biases



# 3. 流密码和RC4
# RC4算法的安全性

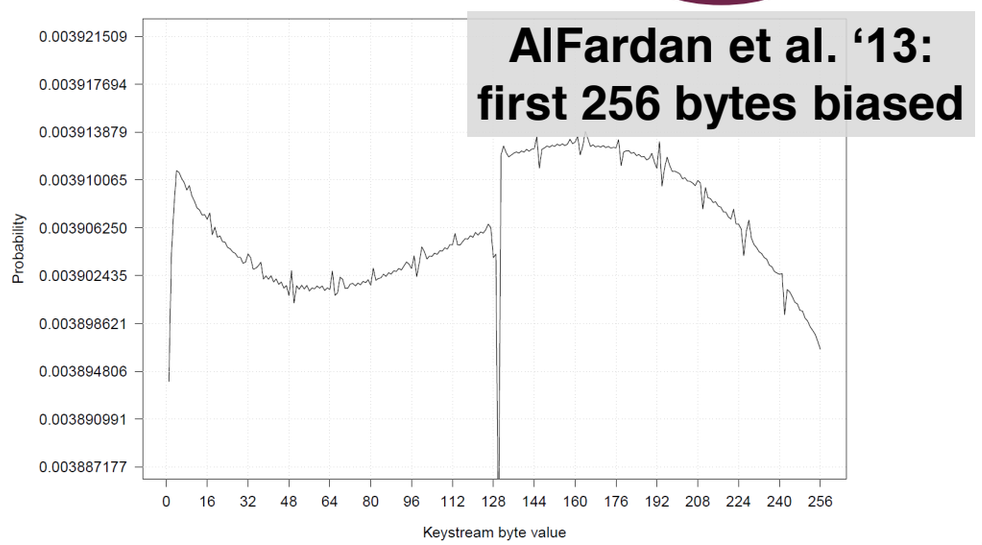
- 问题：Short-Term Biases



# 3. 流密码和RC4
# RC4算法的安全性

- 问题：Short-Term Biases

Distribution keystream byte 1 (to 256)



# 3. 流密码和RC4
# RC4算法的安全性

- 问题：**Long**-Term Biases

Fluhrer-McGrew (2000):

- Some consecutive values are biased

Examples: $(0,0)$ and $(0,1)$

Mantin's ABSAB Bias (2005):

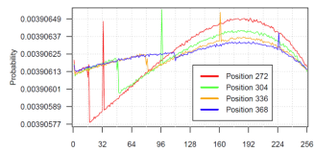
- A byte pair $(A,B)$ likely reappears




huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- RC4 NOMORE attack (Usenix Security '15)




New Biases

$$\lambda_{\widehat{\mu}} = (1-\alpha(g))^{|\mathcal{C}|-|\widehat{u}|} \cdot \alpha(g)^{|\widehat{\mu}|}$$

**Plaintext Recovery**



Break WPA-TKIP



Attack HTTPS

huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- RC4 NOMORE attack (Usenix Security ’15)
  - 已知明文 （Assuming there’s surrounding known plaintext）
  - WPA-TKIP破解: an hour
  - HTTPS-cookie: 75 hours, $9*2^{27}$ request, 4450 r/s

huangwc@ustc.edu.cn

In 2015, security researchers from KU Leuven presented new attacks against RC4 in both TLS and WPA-TKIP.[55] Dubbed the Numerous Occurrence MOnitoring & Recovery Exploit (NOMORE) attack, it is the first attack of its kind that was demonstrated in practice. Their attack against TLS can decrypt a secure HTTP cookie within 75 hours. The attack against WPA-TKIP can be completed within an hour, and allows an attacker to decrypt and inject arbitrary packets.

# 3. 流密码和RC4

## RC4算法的安全性

- RC4 NOMORE attack (Usenix Security '15)
  - HTTPS attack (思路：修改HTTP request)

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
Trident/7.0; rv:11.0) like Gecko
Host: a.site.com
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: auth=??????????????; P=aaaaaaaaaaaaaaaa
```

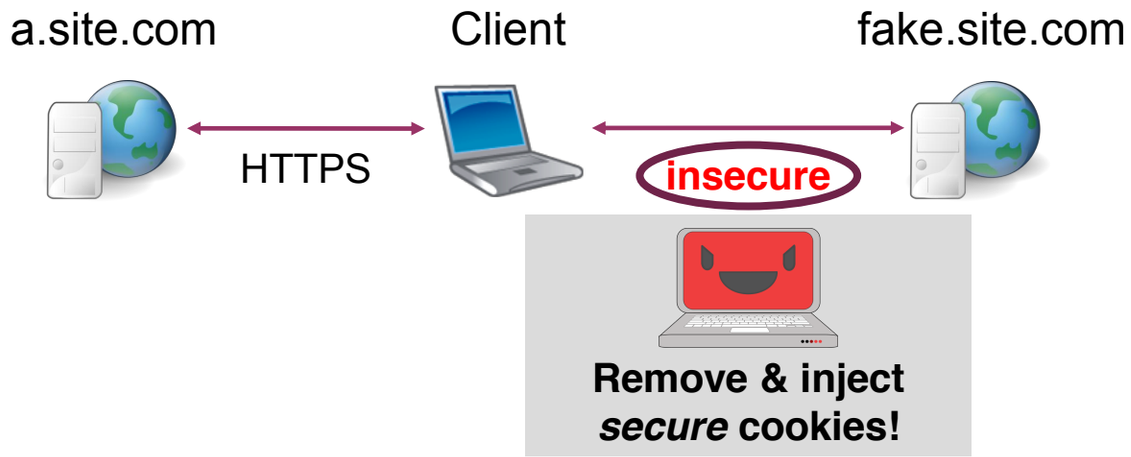
Headers are predictable

**Surrounded by known plaintext at both sides**

huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

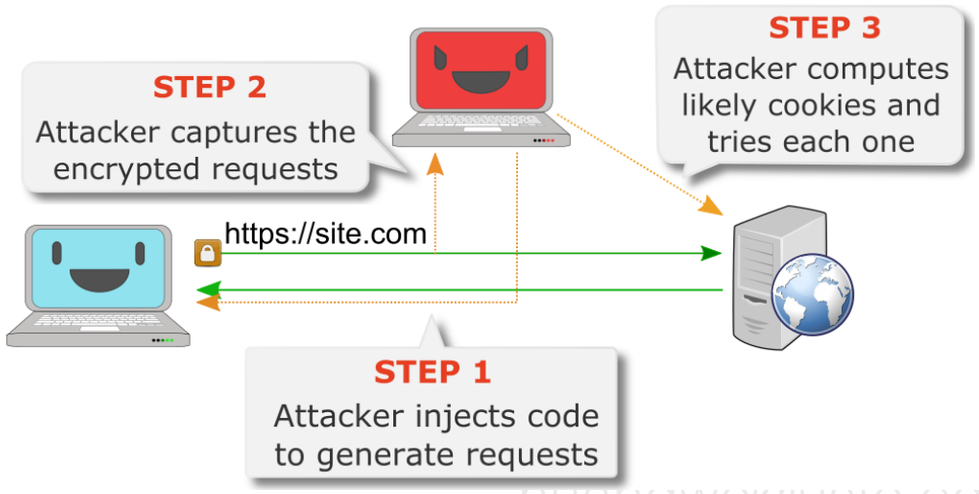
- RC4 NOMORE attack (Usenix Security '15)
  - HTTPS attack




huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- RC4 NOMORE attack (Usenix Security '15)
  - HTTPS attack




huangwc@ustc.edu.cn

# 3. 流密码和RC4
# RC4算法的安全性

- RC4 NOMORE attack (Usenix Security ’15)
  - http://www.rc4nomore.com
  - video

huangwc@ustc.edu.cn

# 作业

6.4 在 DES 的 ECB 模式中，若在密文的传输过程中，某一块发生了错误，则只有相应的明文分组会有影响。然而，在 CBC 模式中，这种错误具有扩散性。比如，图 6.4 中传输 $C_1$ 时发生的错误将会影响明文分组 $P_1$ 和 $P_2$。

(a) $P_2$ 以后的所有块是否会受到影响？

(b) 假设 $P_1$ 本来就有一位发生了错误。则这个错误要扩散至多少个密文分组？对接收者解密后的结果有什么影响？

6.8 在 8 位的 CFB 模式中，若传输中一个密文字符发生了一位错，这个错误将传播多远？

6.5 对于RC4的密钥值是什么的时候，使得S在初始化过程中没有变化？即自对 S进行初始置换后，S的元素的值按升序分别等于0到255？

6.10 在讨论 CTR 模式时，提到如果用给定的计数器值加密的明文分组内容已知，那么从关联的密文分组很容易求出加密函数的输出。请给出计算过程。

huangwc@ustc.edu.cn