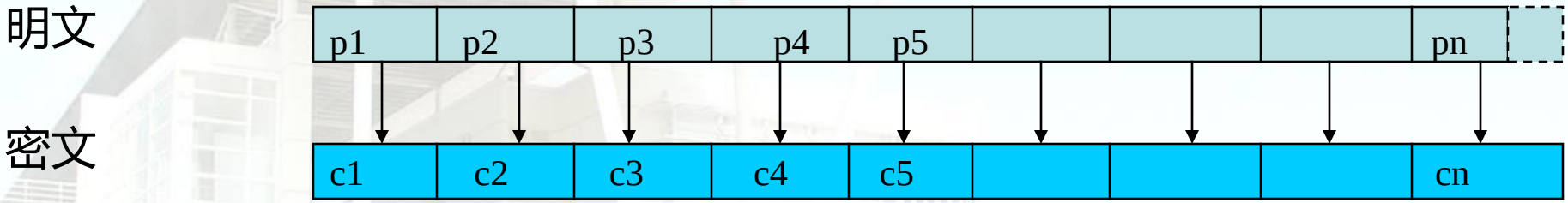
对称密码算法类型

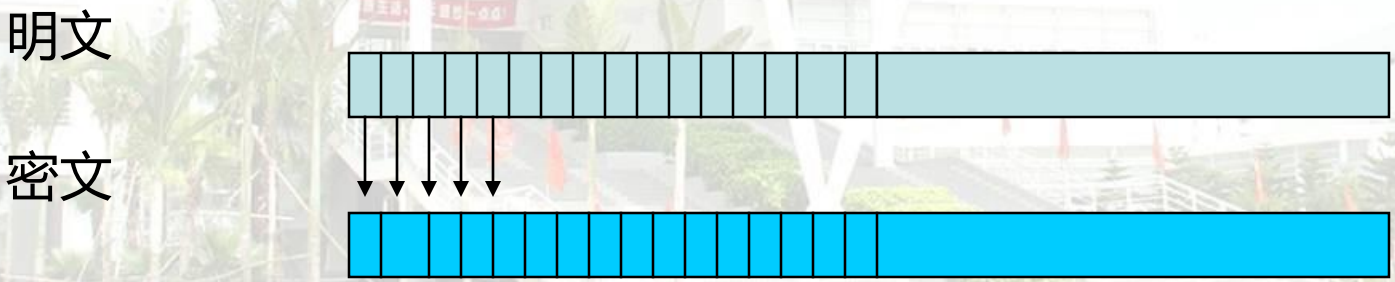
- **分组密码**

  在明文分组和密文分组上进行运算--通常分组长大于或等于64bits。相同的明文和相同的密钥得到相同的密文。



- **流密码**

  作用在明文和密文的数据序列的1 bit 或1 byte 上。



Symmetric Ciphers 对称密码

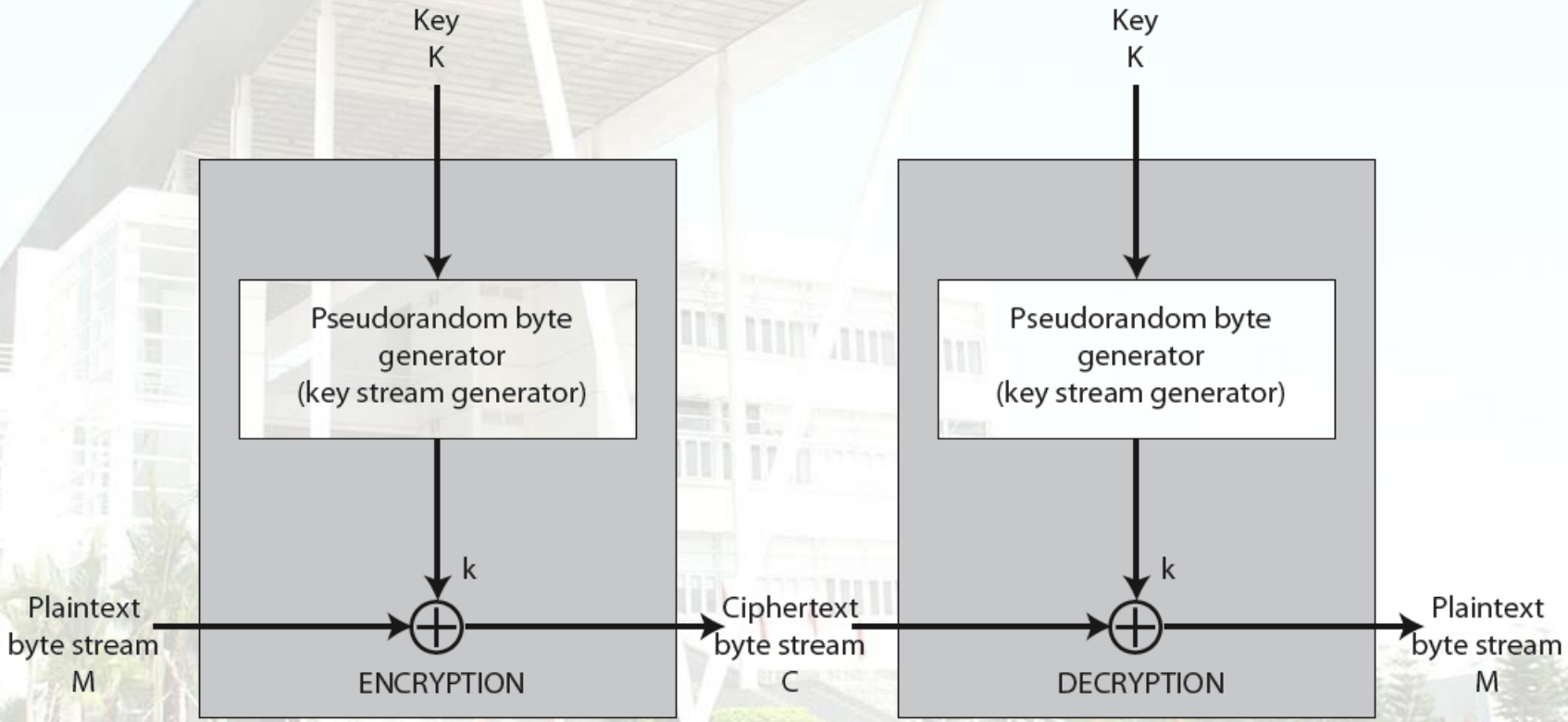
流密码算法

Stream Ciphers

Stream Ciphers

- process message bit by bit (as a stream)
- have a pseudo random **keystream （伪随机密钥流）**
- but must never reuse stream key

Stream Cipher Structure



Stream Cipher －Design Considerations

- long period 伪随机周期要长
- statistically random （统计上随机）
- depends on large enough key （密钥足够长）

RC4

- a proprietary cipher owned by RSA INC，another design by R. Rivest in1987
- 1994年匿名公开
- simple but effective
- variable key size
- faster
- widely used (web SSL/TLS, wireless WEP)

RC4 Key Schedule

- 初始化S： starts with an array S : 0..255 （256 bytes)
- Creates a Temporary Vector, T :

```
T[i] = K[i mod keylen])
```

- Produces the **initial permulation of S** (初始置换)

伪码：

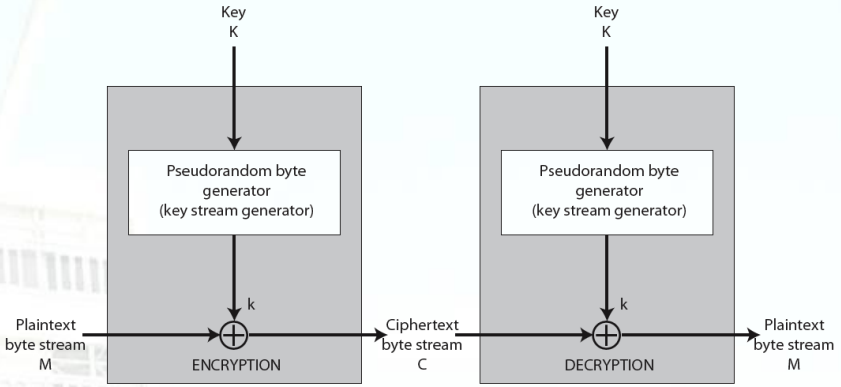
```
/ *初始化S */
for i = 0 to 255 do
S[i] = i
T[i] = K[i mod keylen])
/ *S的初始置换*/
j = 0
for i = 0 to 255 do
j = (j + S[i] + T[i]) (mod 256)
swap (S[i], S[j])
```

密钥的主要功能：
将S搅乱，不同的S在经过伪随机子密码生成算法的处理后可以得到不同的子密钥序列。

RC4 Encryption

- XOR S[t] with next byte of message to en/decrypt

```
/ *密钥流的生成*/
i = j = 0
for each message byte Mi
    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]
/*子密码sub_k用以和明文进行xor运算，得到密文*/
    Ci = Mi XOR k
```



RC4 Security

Key >=128 bit

- since RC4 is a stream cipher, must **never reuse a key**
  加密是采用**xor**，一旦子密钥序列出现重复，密文就有可能被破解
- have a concern with WEP in 802.11
- 谷歌公布放弃 **RC4, September 18, 2015,** 1, 2

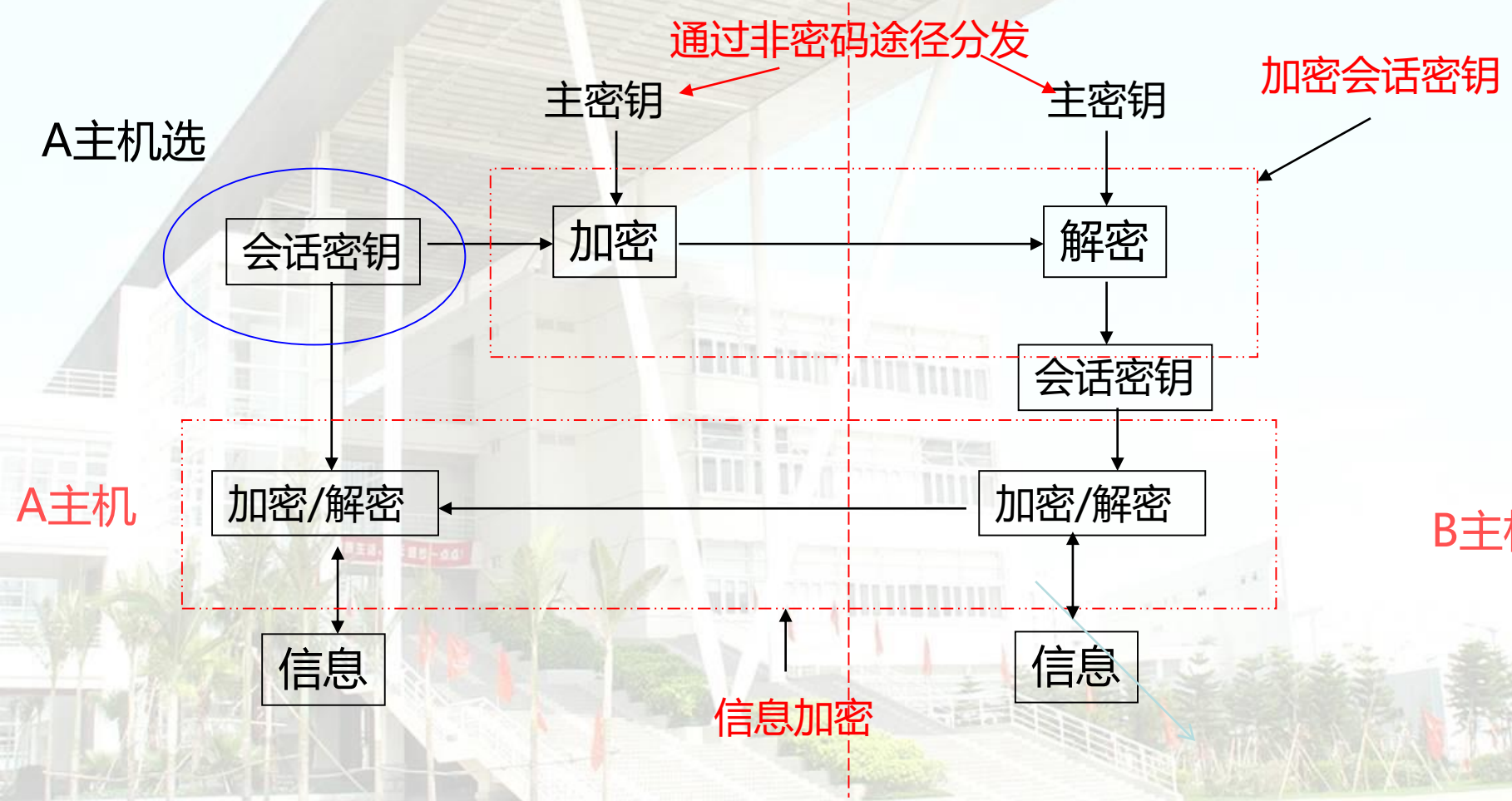
对称密钥分配问题

多种方法：

- 密钥由A选定，通过物理的方式传递给B
- 由第三方选定密钥，然后物理的传递给A和B
- 如果AB曾经使用过一个密钥，一方可以使用旧密钥加密新密钥后传递给另一方
- 如果AB都有一个到第三方C的加密连接，C就可以用加密形式把密钥传递给AB

设立密钥分配中心（KDC）越来越重视及应用

分层次：A选定会话密钥，用主密钥加密



Key Distribution Scenario