

# Info Security

---

## 1. Introduction

---

### What is security?

什么是安全

What is security?

系统、应用程序或协议的安全性总是与以下因素相对应:

The security of a system, application, or protocol is always relative to:

- 一组预期特性: 想要实现什么?  
A set of desired properties: what do want to achieve?
- 具备特定能力的对手: 他们能做什么?  
An adversary with specific capabilities: what can they do?
- 我们需要什么样的安全 (服务) ?  
What security (service) do we need?
- 我们要抵御多强大的攻击者?  
How powerful an attacker to we want to defend against?

理论安全在任何情况下都是安全的。这很难做到--例如, 没有一种加密算法 (一次性密码匙除外) 在理论上是安全的, 因为你可以进行穷举密钥搜索。

**Theoretic security** is always secure under all circumstance. This is difficult – for example no encryption algorithm (except one-time pad) is theoretically secure as you can do exhaustive key search.

计算安全是指规避一项服务实际上并不可行。例如, 我可以暴力破解 AES 加密信息, 但就目前的技术而言, 我需要花费很多很多年的时间。

**Computational security** is when it is not practically feasible to circumvent a service. For example, I could brute force an AES encrypted message but given current technology it will take me many many years

有时, 人们也会根据成本做出决定--如果攻击者破解安全系统所需的成本高于他破解安全系统所能获得的收益, 那就可以认为是安全的。这对某些类型的攻击者有效, 但并不总是最佳方法, 因为攻击者的动机很难预测--他们可能不想赚钱!

Sometime people also make a decision based on cost – if an attacker needs to spend more money breaking the security than what he would make by breaking the security it is considered secure. This works for certain types of attackers but is not always the best approach given that attacker motives are hard to predict – they might not want to make money!

安全并不总是与技术有关.....我们需要从更广泛的角度来思考系统安全问题, 而不仅仅是密码学问题

Security is also not always about technology... We need to think in more general terms of system security rather than just cryptography

安全是对资产的保护

Security is about the protection of assets

## protection measures

保护措施分为三大类：

There are three broad classes of protection measures:

- **预防**: 防止您的资产受损  
**Prevention**: prevent your assets from being damaged.
- **检测**: 检测资产何时受损、由谁受损以及如何受损。  
**Detection**: detect when your assets have been damaged, by whom and how.
- **反应/恢复**: 恢复资产，或从资产受损中恢复。  
**Reaction/Recovery**: recover your assets, or recover from the damage to your assets.

## Basic Security goals(CIA)

最基本的安全目标： **(CIA)**

- **保密性**: 防止未经授权披露信息。 (其他人得知它)  
**Confidentiality**: prevention of unauthorised disclosure of information. (Someone else gets hold of them)
- **完整性**: 防止对信息进行未经授权的修改。 (其他人修改它)  
**Integrity**: prevention of unauthorised modification of information. (Someone changes it)
- **可用性**: 防止未经授权扣留信息或资源。 (其他人不许我用它)  
**Availability**: prevention of unauthorised withholding of information or resources. (Someone prevents me from getting it)

## Threads

威胁： **Threads**:

- **安全威胁**: 是违反安全策略的可能手段 (如丧失完整性或保密性)

A **security threat** is a possible means by which a security policy may be breached (e.g. loss of integrity or confidentiality)

- **反制措施**: 是防止威胁的控制措施  
**Countermeasures** are controls to protect against threats
- **漏洞**: 是系统 (和/或对策) 中的薄弱环节。  
**Vulnerabilities** are weaknesses in the system (and/or countermeasures).
- **攻击**是威胁的实现 (利用漏洞) 。  
An **attack** is a realisation of a threat (exploiting a vulnerability).

威胁可以被分为：

- **故意/意外**  
**deliberate/accidental**

另外，根据他们与CIA相对应还可以分为：

- **数据暴露**: 未经授权的人访问数据的威胁 (相对保密性)

**Exposure of data**: the threat that someone who is unauthorised can access the data

- **篡改数据**: 数据可能被篡改的威胁。 (相对完整性)

**Tampering with data**: the threat that the data could be altered from what it should be.

- **拒绝服务**: 数据或服务在需要时无法使用的威胁。 (相对可用性)

**Denial of service**: the threat that the data or service is unavailable when it is required.

注意: 以上三种都可以成为故意或意外

## Adversaries

**敌手 (Adversaries) :**

- **被动型**敌手只试图在未经授权的情况下获取信息

**Passive** adversaries only attempt to get unauthorised access to information

- **主动型**敌手是任何被动型以外的敌手

**Active** adversaries is any adversary other than passive

- 未经许可的改动

Unauthorised alteration

- 未经授权的删除

Unauthorised deletion

- 未经授权的传输

Unauthorised transmission

- 伪造信息来源

Falsification of origin of information

- 未经授权阻止获取信息

Unauthorised prevention of access to information

敌手不一定是第三方或是外部的

The adversary does not have to be a third party or external

敌手可能拥有层出不穷的技能

The adversary may have an endless array of skills

## Security Services and Mechanisms

**安全服务**是一种为应对威胁而采取的措施 (如提供保密性)。(抽象! )

A **security service** is a measure which can be put in place to address a threat (e.g. provision of confidentiality)

**安全机制**是提供服务的一种手段 (如加密、数字签名)。(具体! )

A **security mechanism** is a means to provide a service (e.g. encryption, digital signature).

**数据保密:**

## Data Confidentiality:

防止未经授权披露信息。

Protection against unauthorised disclosure of information.

对数据进行加密后传输，传输到后再进行解密。主要是预防措施，其余两者表现不好。

The data is encrypted for transmission and then decrypted after transmission. Mainly prevent, the rest of the two do not perform well.

## 数据完整性：

### **data integrity**

完整性是防止未经授权修改数据的保护措施

Integrity is protection against unauthorised modification of data

无法预防修改，我们需要检测，最好能恢复

There is no way to prevent modifications, we need to detect and preferably recover them

## 认证：

### **Authentication**

- **实体身份验证**可在某个时间点对声称的身份进行检查。

**Entity authentication** provides checking of a claimed identity at a point in time

- 通常在连接开始时使用。

Typically used at start of a connection

- 解决伪装和重放威胁。

Addresses masquerade and replay threats.

例如Alice向Bob提问得知了一个只有Bob才知道答案的问题，由此知道了对方就是Bob这个实体，可以防止伪装或重放

- **来源认证**可对数据来源进行验证。

**Origin authentication** provides verification of source of data.

- 不能防止重放或延迟

Does not protect against replay or delay

例如一封信上有来自Bob的签名，Alice可得知这封信由Bob写的，但无法确定把信递给她的就是Bob，也就是可能被伪装或重放攻击

## 访问控制：

Access Control:

提供保护，防止未经授权者使用资源，如：

Provides protection against unauthorised use of resource:

- 使用通信资源

use of a communications resource

- 读取、写入或删除信息资源

reading, writing or deletion of an information resource

- 执行处理资源

execution of a processing resource

## 不可否定性:

### **Non-repudiation**

分为两种:

- 防止数据发送方否认发送过数据 (不可否认来源)。

Protects against a sender of data denying that data was sent (non-repudiation of origin).

- 防止数据接收方否认收到数据 (不可否认交付)。

Protects against a receiver of data denying that data was received (non-repudiation of delivery).

示例: 类似于签署一封信并通过记录投递的方式寄出。

Example: analogous to signing a letter and sending via recorded delivery.

不可否认性在电子商务中的重要性在于, 当在线交易发生时, 买卖双方形成了一个法律合同, 约定用货物换取金钱。如果其中一方声称从未同意这个协议, 那么就需要有证据来证明交易的存在以及各方的认可。

**安全威胁**是指对系统安全构成威胁的事物。选择**安全服务**是为了应对已识别的威胁, 而**安全机制**则是提供服务的手段。

a **security threat** is something that poses a danger to a system's security. A **security service** is selected to meet an identified threat, and a **security mechanism** is the means by which a service is provided.

- Confidentiality (data disclosure)
- Integrity (data alteration)
- Availability (DoS)
- Entity Authentication (masquerade)
- Origin Authentication (forgery)
- Non-repudiation (repudiation it did not happen!)
- Access Control (illegitimate access)

## 机制:

### **Mechanisms:**

安全机制是提供服务的一种手段。

A security mechanism is a means to provide a service.

可以被分成:

- 用于提供特定安全服务的**特定安全机制**, 如数字签名 (完整性、来源或实体身份验证)

**Specific security mechanisms**, used to provide specific security services, e.g. digital signature

- **普及型安全机制**, 不针对特定服务, 如事件检测、标签等 (不提供直接安全服务但可以检测各自潜在威胁)

**Pervasive security mechanisms**, not specific to particular services, e.g. event detection, labelling

机制例子:

- Confidentiality ( encryption) –
- Integrity (MAC/digital signature) –
- Availability (redundancy) –
- Entity Authentication (authentication protocol) –
- Origin Authentication(MAC/digital signature) –
- Non-repudiation (digital signature) –
- Access Control (Access control model)

## Algorithms

算法用于建立机制

**Algorithms** are used to build mechanisms

- Encryption: DES/3DES/AES (modes) or RSA/ECC  
CAST(Canada), MISTY1/Camellia (Japan), SEED (Korea) –
- MAC: CBC mode, HMAC –
- Digital Signature: RSA, DSA, ECC –
- Hash: SHA-3 –
- Random number: True or Pseudorandom

## Standards

标准是由公司或国家机构定制的一套规范

A standard is a set of norms customised by a company or state agency

使用标准的优势: (advantages)

### 1. 成本效益和时间效率: Cost-effective and time-efficient:

- 成本效益: 使用标准可能比自己开发解决方案更经济。尽管标准本身并不能直接降低数据泄露的成本, 但它可以减少因开发自定义解决方案所需的时间和资源。

Cost-effectiveness: It may be more economical to use a standard than to develop your own solution. While a standard by itself does not directly reduce the cost of a data breach, it can reduce the time and resources required to develop a customised solution.

- 时间效率: 采用标准可以节省时间, 因为不必从头开始构建解决方案。

Time efficiency: Adopting standards saves time by not having to build solutions from scratch.

### 2. 商业可行性: Commercial feasibility:

- 标准的使用可以使产品或服务在市场上显得更加可信, 从而增加客户的信心。这可能会帮助企业打开新的市场, 因为客户会认为企业遵循的是最佳实践, 从而提高信任度。

The use of standards can increase customer confidence by making a product or service appear more credible in the marketplace. This may help the organisation to open up new markets as customers will perceive that the organisation is following best practices, thus increasing trust.

### 3. 可信度: Credibility:

- 与商业可行性相关联, 使用标准可以提升企业的信誉, 使企业看起来更加专业和可靠。

Linked to commercial viability, the use of standards can enhance the credibility of a business, making it appear more professional and reliable.

#### 4. 安全性：Security:

- 由于采用了由专家共同开发的最佳实践，因此使用标准可能会使解决方案更加安全。标准通常经过广泛测试和审查，因此它们可能包含了更全面的安全考量。

The use of standards may make the solution more secure because of the adoption of best practices developed by experts working together. Standards are often extensively tested and reviewed, so they may contain more comprehensive security considerations.

标准的三种用法：the three uses of a standard

- 作为解决方案的起点：**阅读标准，试图理解专家们的意图，然后基于标准的部分内容自行设计和实施解决方案

**As a starting point for a solution:** read the standard, try to understand what the experts are trying to do, and then design and implement a solution on your own based on parts of the standard

- 遵从：**严格按照标准描述的内容实施解决方案。最终，你按照自己的最佳能力实现了标准中所述的要求，从而符合标准的规定。

**Compliance:** Implementation of the solution in strict accordance with what the standard describes. In the end, you fulfil the requirements described in the standard to the best of your ability and thus comply with the standard.

- 认证：**在尽自己最大努力实施标准后，邀请一个公正的第三方机构来审核你的做法是否符合标准的要求。这个第三方通常是在该领域内被信任的机构，他们会出具一份证明，表明你已经正确地实施了标准

**Certification:** After doing your best to implement the standard, invite an impartial third-party organisation to audit your practices for compliance with the requirements of the standard. This third party is usually an organisation that is trusted in the field, and they will issue a certificate showing that you have implemented the standard correctly.

不利因素：

- 共识决策意味着妥协： compromise**

- 标准需要由标准组织成员达成一致意见。由于成员们可能有不同的观点，特别是在涉及范围很广的标准时，达成一致意见可能意味着标准需要支持多种选项和服务。这使得全面实施标准变得成本高昂且困难。

Standards need to be agreed upon by the members of the standards organisation. As members may have different perspectives, especially in the case of standards that cover a wide range of topics, agreement may mean that the standard needs to support a wide range of options and services. This makes full implementation of the standard costly and difficult.

- 文档可能存在不一致的解释： inconsistent interpretations of documents**

- 在任何大型文档中，错别字和模糊措辞都很容易出现，尤其是当用户或编写者尝试用外语解读文档时。

In any large document, typos and ambiguous wording are easy to come by, especially if the user or writer is trying to decipher the document in a foreign language.

- 商业压力和标准选项范围广导致部分实施： Commercial pressures and the wide range of standard options have led to partial implementation:**

- 由于商业压力和大型标准中通常提供的多种选项，公司常常不会实施整个标准，而是仅实施标准中包含的功能子集。这样做的结果是，尽管他们可以声称自己符合标准，但往往与其他符合标准的产品不兼容。

Due to commercial pressures and the wide range of options typically available in large standards, companies often do not implement the entire standard, but only a subset of the functionality included in the standard. The result is that, although they can claim to be compliant, they are often incompatible with other compliant products.

- 市场主导企业可能用专有版本替代独立标准： Market dominant firms may substitute proprietary versions for stand-alone standards:**

- 拥有较大市场份额的企业有可能通过发布广泛分发的产品来替换独立标准，这些产品适应或扩展了现有标准。企业可能会觉得他们可以改进标准，并将其纳入自己的产品中，同时加入专有的功能。这种方式下，企业可以声称自己符合标准，但却让用户依赖于实际上是专有的功能。如果这种产品被广泛使用，则会削弱原始标准的地位。

Enterprises with large market shares are likely to replace stand-alone standards by releasing widely distributed products that adapt or extend existing standards. Enterprises may feel that they can improve the standard and incorporate it into their own products while adding proprietary features. In this approach, firms may claim to be compliant with the standard, but leave users dependent on what is in fact proprietary functionality. If such products are widely used, this can weaken the status of the original standard.

有官方标准机构（制定世界或国家级标准），有制定标准的公司（通常有商业利益--有好有坏，有些你必须使用，如支付卡标准），还有互联网标准

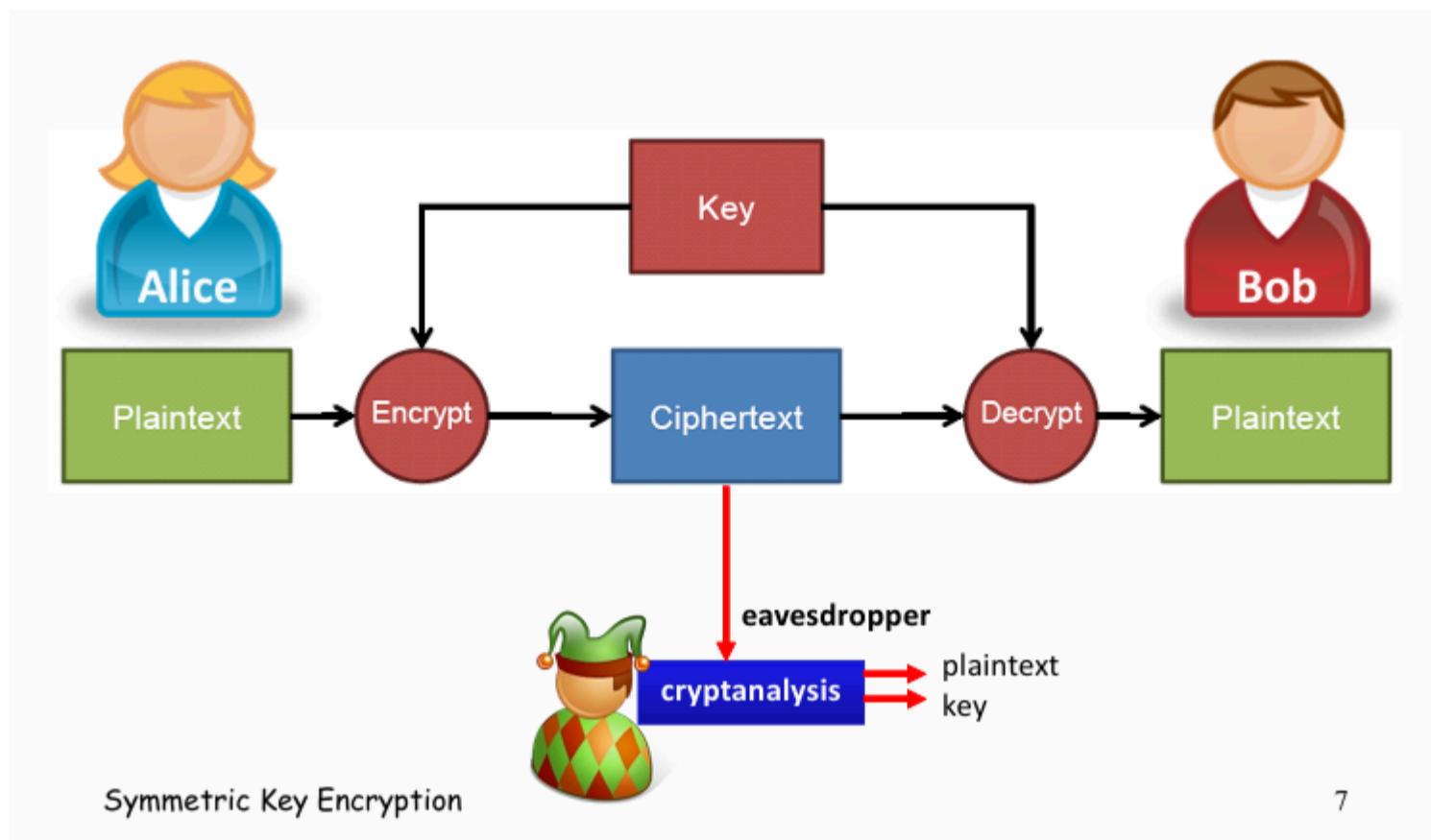
there are official standards bodies (defining standards at worldwide or national level), there are companies who make standards (often with commercial interest – some good some bad, and some you have to use like payment card standards) and there are Internet standards

## 2. Symmetric Key Encryption

### About

**对称密钥加密：Symmetric key encryption:**

- 使用对称密钥密码或密码系统对明文/密文进行加密/解密  
A symmetric-key cipher or cryptosystem is used for encrypting/decrypting a plaintext/ciphertext
- 加密和解密使用相同的密钥  
The same key is used for encrypting and decrypting



而非对称密钥加密（Asymmetric Key Encryption）中，

非对称密钥加密使用一对不同的密钥，即公钥（Public Key）和私钥（Private Key）。公钥可以公开分发，而私钥必须由持有者妥善保存。使用公钥加密的信息只能用对应的私钥解密，反之亦然。

Asymmetric key encryption uses a pair of different keys, the Public Key and the Private Key. The Public Key can be distributed publicly, while the Private Key must be kept safe by the holder. A message encrypted with the Public Key can only be decrypted with the corresponding Private Key and vice versa.

克尔克霍夫斯原则 (Kerckhoffs's Principle) :

在密码分析中，有两个基本原则： There are two basic principles in cryptanalysis:

- 系统完全为攻击者所知

The system is completely known to the attacker

- 只有密钥是秘密

Only the key is secret

也就是说安全无法建立在隐蔽性之上，算法应该透明化，密码分析聚焦于对密钥的保护上

This means that security cannot be built on obscurity, algorithms should be transparent, and cryptanalysis focuses on the protection of keys

## Caesar Cipher

一种替代密码

a type of substitution cipher

密码算法：用字母表中 n 位的字母替换明字母表中的每个字母

Cipher algorithm: each letter in the plain alphabet is replaced with the letter n places further on in the alphabet

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

上图中k=3 (小写字母右移三位得到大写字母)

上述替代密码中，密码就是移位的偏移量 (The offset of the shift.)。因此凯撒密码是一种特殊的替代密码。

通常替代密码不会有如此简单的规律

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
P	U	R	E	M	A	T	H	I	C	S	N	W	Y	O	F	L	G	D	Z	X	V	Q	K	J	B

将未使用过的字母按一定的系统顺序填入字母表中 (这里我们使用的是反字母表顺序)

这种密码字母表的可预测性比凯撒密码低，但只要发送方和接收方知道关键短语，仍然可以很容易地生成密码

This cipher alphabet is less predictable than Caesar's cipher, but can still be easily generated as long as the sender and receiver know the key phrases

然而这并不意味着替代密码的安全性低下。对于暴力破解来说，替代密码中第一个字母对应着其他25中可能性，第二个则为24种.....

因此可能的组合数量有26！

However this does not mean that alternative passwords are less secure. For brute force decryption, the first letter in the alternative password corresponds to one of the other 25 possibilities, and the second to 24 ..... Thus the number of possible combinations is 26!

$$26! = 403,291,461,126,605,635,584,000,000(27 \text{ digits}) \approx 2^{88}$$

可行的破解方法是频率分析 (Frequency Analysis)

替换不会改变明文的字母频率，ET最常出现

Substitution does not change the letter frequency of the plaintext

letter	frequency	letter	frequency
A	.082	N	.067
B	.015	O	.075
C	.028	P	.019
D	.043	Q	.001
E	.127	R	.060
F	.022	S	.063
G	.020	T	.091
H	.061	U	.028
I	.070	V	.010
J	.002	W	.023
K	.008	X	.001
L	.040	Y	.020
M	.024	Z	.001

因此从统计学角度看，置换密码不够强大。

substitution ciphers are not statistically strong enough

## One-time Pad Encryption

Plaintext XOR Key = Ciphertext

<b>Ciphertext:</b>	97	6F	DE	31	A8	A2	82	4B	5C	0D
<b>Key:</b>	FF	0A	B2	5D	C7	C3	EE	22	3F	68
<hr/>										
<b>Plaintext:</b>	68	65	6C	6C	6F	61	6C	69	63	65
	h	e			o	a		i	c	e

- Pad必须是随机的，只能使用一次

Pad must be random, used only once

- Pad大小与信息相同

Pad has the same size as message

在只用一次的情况下，One-time Pad提供了完美的安全性

但并不实际，因为：

- 每次发送信息时，我们都需要发送一个新的随机 "PAD"。

Each time we send a message we need to send a new random 'pad' with it

- 我们需要一个与信息长度相同的密钥。

We need a key that is the same length as the message.

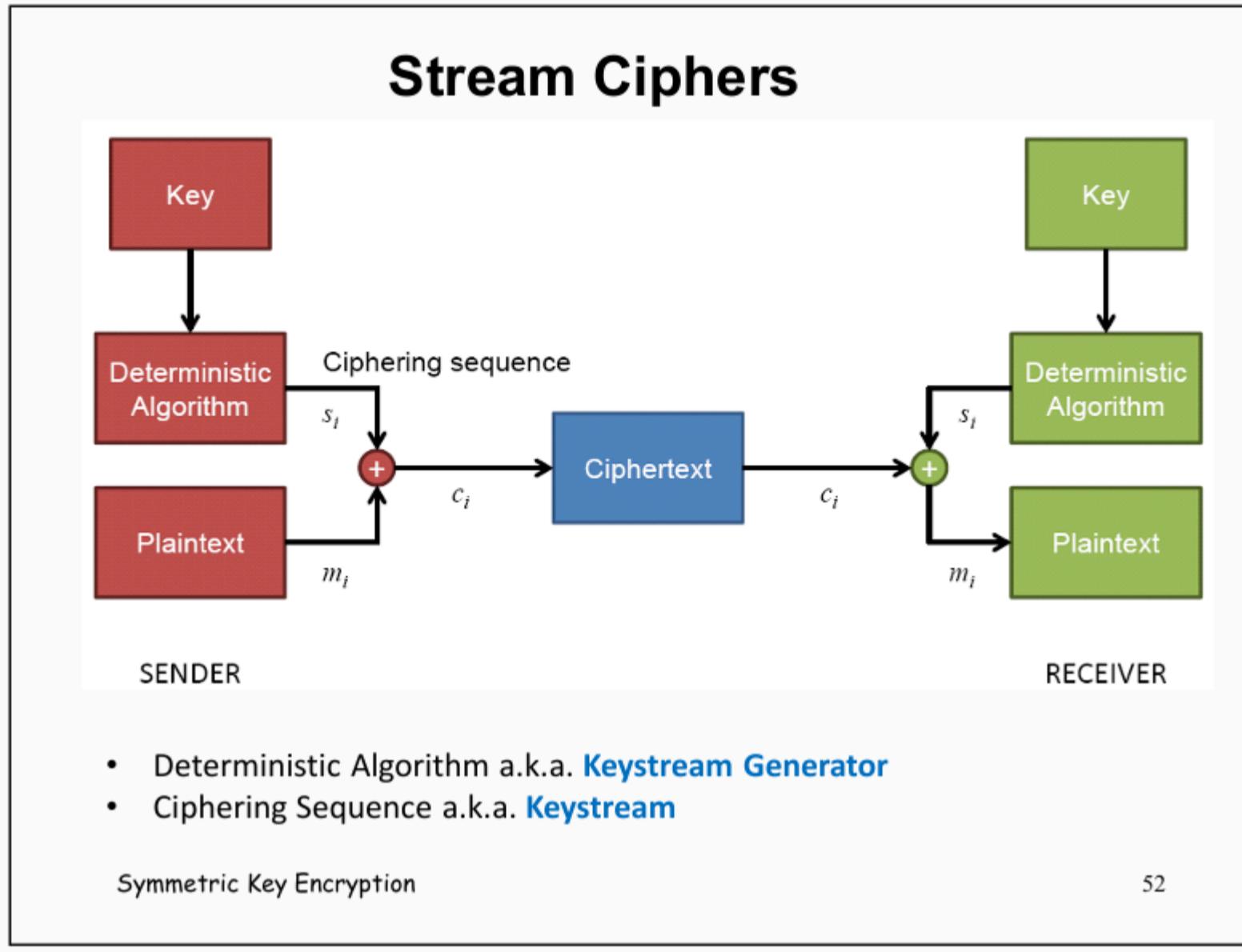
- 我们每次还需要一把新钥匙

We also need a new key each and every time

完美保密意味着攻击者对明文没有任何了解。这不是说攻击者不知道明文的具体内容，而是指他对于明文没有任何线索或信息。例如，如果我们使用一个好的密码算法，并且两次加密相同的明文得到了相同的密文，那么即使明文未知，但由于我们知道两次加密的结果相同，这就破坏了完美保密，因为我们知道了某些信息——即消息是相同的。

perfect secrecy means that the attacker has no idea about any of the plaintext – this does not mean he does not know the plaintext, this means he knows *nothing*. For example, if we use a good cipher and we encrypt the same plaintext twice and it gives the same ciphertext then even the plaintext is not known we lose perfect secrecy as we know something – the message is the same.

## Stream Ciphers



XOR 是特殊的二进制函数，如果  $C = A \text{ XOR } B$ , 那么  $C \text{ XOR } A = B$ , 或者  $C \text{ XOR } B = A$

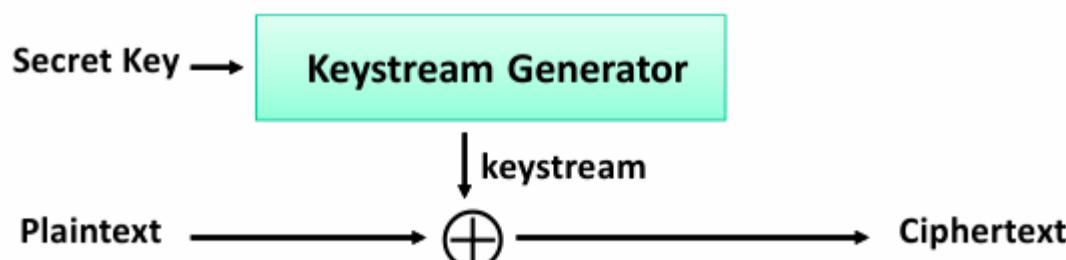
XOR is special binary function, if  $C = A \text{ XOR } B$  then  $C \text{ XOR } A = B$ , or  $C \text{ XOR } B = A$

密钥被输入一个确定性算法（未来状态没有随机性的函数--可复制、可预测）。这仅适用于拥有密钥的人--如果你知道密钥，算法就是确定的。

The key is fed into a deterministic algorithm (a function that has no randomness in future states – reproducible and predictable). This only holds for someone with the key – if you know the key the algorithm is deterministic.

如果有人只是观察输出，而他没有密钥，那么输出看起来就像是随机的。

If someone just observes the output, and he does not have the key, it appears like random output.



流密码是一种加密方法，它通过对明文逐位（或逐字节）进行加密来生成密文

A stream cipher is an encryption method that generates a ciphertext by encrypting the plaintext bit-by-bit (or byte-by-byte)

- **密钥 (Secret Key)** : 用于生成密钥流的密钥

- **明文 (Plaintext)** : 需要加密的数据。
- **密钥流生成器 (Keystream Generator)** : 产生用于加密的密钥流。
- **密钥流 (Keystream)** : 由密钥流生成器产生的随机序列，用于加密明文。有着和明文相同的长度
- **密文 (Ciphertext)** : 通过明文与密钥流进行XOR运算后得到的结果。由于直接使用密钥流和密文进行XOR操作即可得到明文，因此不需要加密的逆操作

我们认为一个流密码是安全的，如果攻击者获得了一段很长的密钥流（而不是密文），他既不能确定密钥，也不能计算出未来的密钥流。

We consider a stream cipher to be secure if – the attackers gets a long sequence of the keystream (not the ciphertext) then he cannot 1) determine the key 2) calculate future keystream.

**Secret key length:** 128 bits, 256 bits, etc.

**最大明文长度：**通常可以任意长。

**Maximum plaintext length:** usually can be arbitrarily long.

### 优点

- **速度快**: 流密码被认为是快速的加密方法。  
Stream ciphers are considered to be fast
- **可扩展性强**: 可以根据明文的长度生成相应长度的密钥流。  
can make as much cipherstream as you have plaintext

### 缺点:

- **已知明文攻击**: 如果攻击者知道了明文，他可以通过明文与密文进行XOR运算来得到密钥流。然后，攻击者可以用得到的密钥流与自己构造的消息进行XOR运算，生成新的密文。  
if we know the plaintext, we can XOR the plaintext with the ciphertext – then we have the keystream. Then we XOR it with our own message.
- **完整性问题**: 由于使用XOR运算，如果只考虑加密而不提供数据完整性保护，攻击者可以在不知道明文的情况下修改密文，从而改变解密后的明文内容。例如，如果发送了一个值 `0010` 并用密钥流 `1010` 加密（密文 `c = 0010 XOR 1010 = 1000`），攻击者可以修改密文为 `c' = 0000`，那么解密后的新明文将是 `P' = 0000 XOR 1010 = 1010`，这改变了原始值。

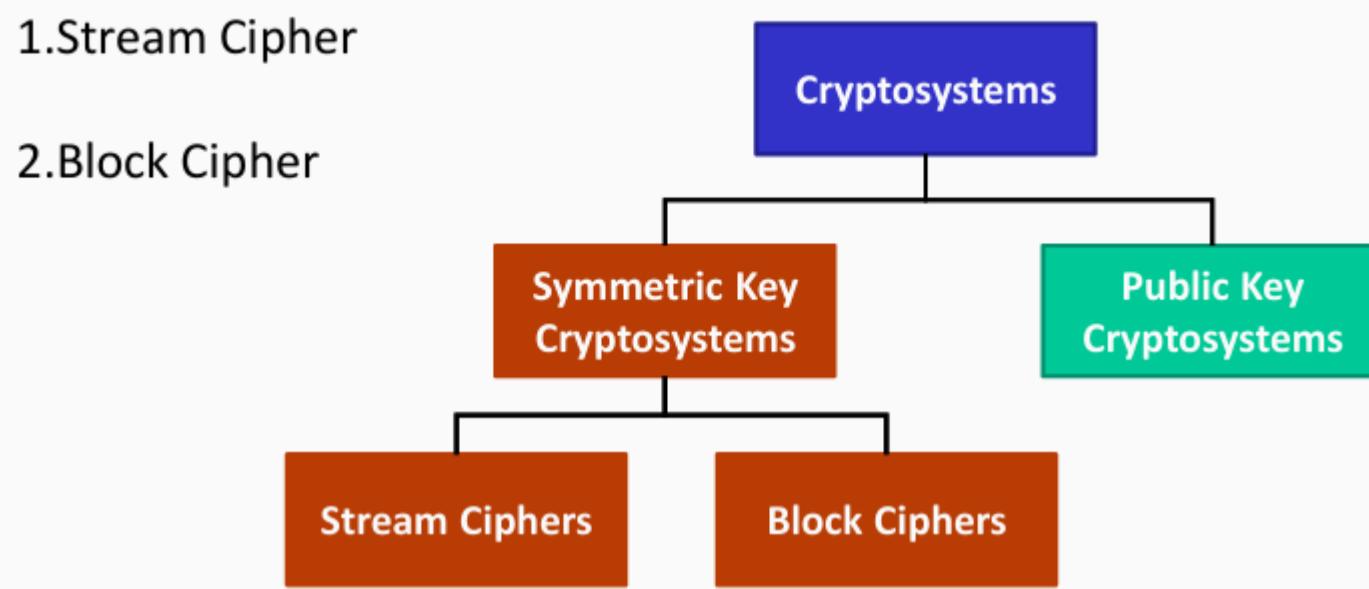
如果我们要在实践中使用流密码，就需要额外的安全服务，尤其是完整性/数据来源认证方面的服务。

If we are to use stream ciphers in practice we need additional security services – especially with regards to integrity/data origin authentication

RC4 就是流密码的一个例子

RC4 is example of a stream cipher

# Block Ciphers



**分块密码** (Block Cipher) : 分块密码接收明文块和密钥，生成密文块，密钥在不同的明文块重复使用。

A block cipher takes a block of plaintext and a secret key, produces a block of ciphertext. The key is reused for different plaintext blocks

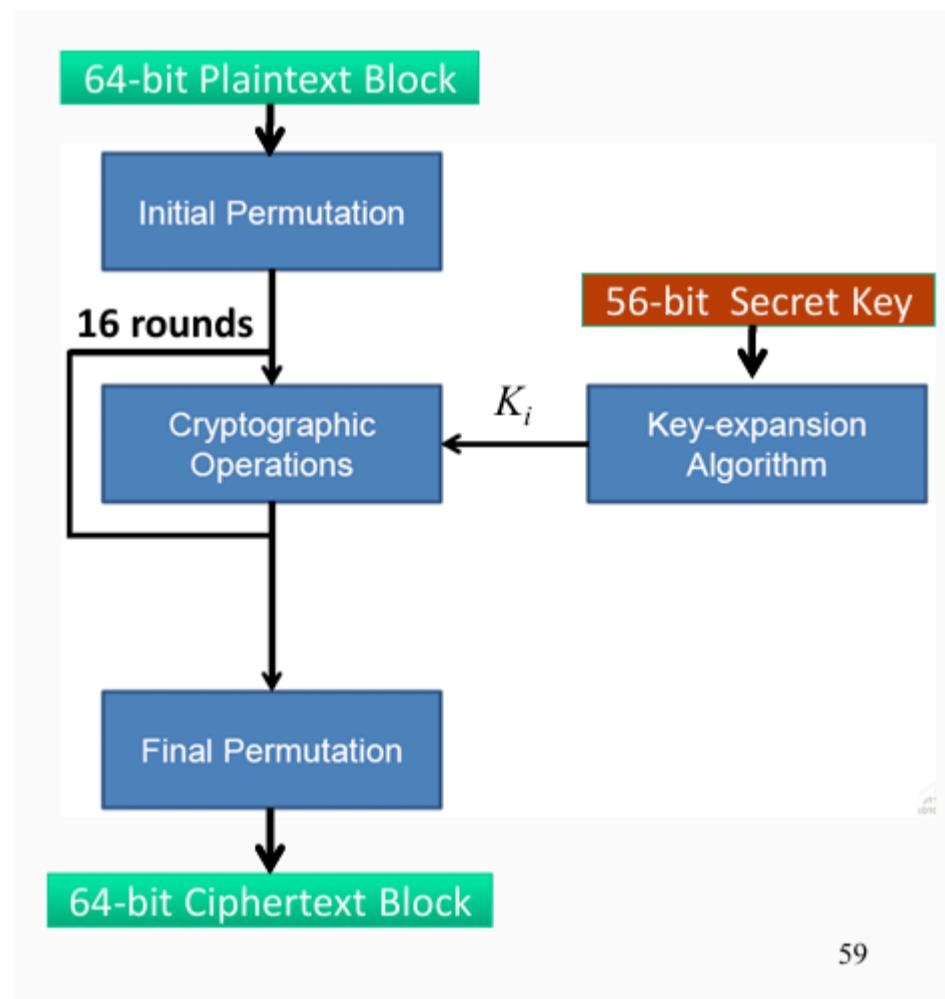
常用密钥规模: 64 bits, 128 bits, 192 bits, 256 bits

Typical block sizes: 64 bits, 128 bits, 192 bits, 256 bits

Key sizes: 56 bits (DES), 128/192/256 bits (AES)

分块密码接收固定的明文输入，若长度不足则进行填充，随后产生固定长度的密文输出。需要加密算法的逆操作以进行解密

## DES structure



59

## DES (Data Encryption Standard)

- 56位密钥
- 64位明文
- 64位输出
- 初始排列---->16轮加密函数（每轮都有独特密钥）----->最终排列
- 每一轮加密函数都包含扩展，替换和排列。 expansion, substitution and permutation
- S盒中将每6位替换成4位 Substitution every 6 bits to 4 bits

- P-box (Permutation Box) 是一个固定的位重排表 one fixed shuffle of each bit

优秀块密码算法的特性: Properties of good block cipher algorithms

- 混乱: Confusion

- 密钥的微小变化应能改变 50% 的密文文本

A small change in the key should be able to change 50% of the ciphertext

- 使用暴力破解攻击的攻击者不应该收到任何迹象表明他正在接近正确的密钥

An attacker using a bruteforce attack shouldn't receive any signs that he is getting closer to the correct key

- 扩散: Diffusion

- 明文的微小变化应导致 50% 的密文发生变化

A small change in the plaintext should cause 50% of the ciphertext to change

- 隐藏明文和密文之间的任何统计关系

Hide any statistical relation between the plaintext and the ciphertext

- 完备: Completion

- 密文的每一位都取决于密钥的每一位

Each bit of the ciphertext depends on each bit of the key

- 攻击者无法使用分而治之的方法找到密钥的有效部分

The attacker won't be able to find valid parts of the key using divide and conquer methods

针对DES的最佳攻击是穷尽密钥攻击 (exhaustive key search)

当前在对称加密中推荐的最小密钥长度是128 bits

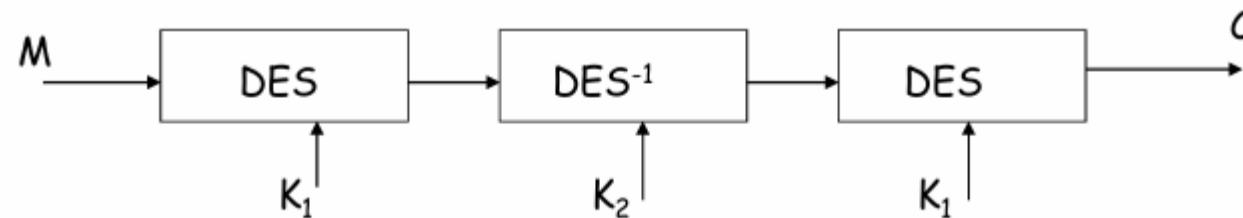
128 bit symmetric keys recommended

对于n位密钥，可能的密钥数量是 $2^n$ ，平均要尝试一半的密钥数量才能找到密钥，也就是 $2^{n-1}$

而DES的 $2^{56}$ 目前已经可被暴力破解 (可考虑提高密钥长度以提高安全性)

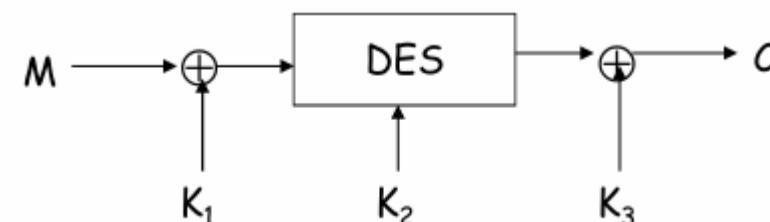
目前推荐使用的块密码是3DES和AES

3DES:



两个56位密码，加密解密再加密

DESX:



3个56位密码，XOR加密再XOR

注：看似密钥长度是112位和168位，但有效密钥空间（effective key space）实际是80位和112位，长于DES但短于AES

## AES (Advanced Encryption Standard)

- 128位块长
- 128-256位密钥长
- 根据不同密钥长运行10-14轮
- 在许多平台上比3DES更快

主要阶段包括：

- 初始轮 (**Initial Round**)：明文与第一轮密钥 XOR.  
XOR plaintext with key of the first round
- 主加密轮 (**Main Rounds**)
  - 字节代换 (**SubBytes**)：使用一个固定的非线性S盒 (Substitution Box) 对每一个字节进行替换  
Substitution of each byte using a fixed non-linear S-box (Substitution Box)
  - 行移位 (**ShiftRows**)：将矩阵的每一行向左循环移位  
Cyclically shift each row of the matrix to the left
  - 列混合 (**MixColumns**)：通过矩阵乘法对每一列的数据进行混合  
Mixing the data in each column by matrix multiplication
  - 密钥加法 (**AddRoundKey**)：将当前轮的子密钥与状态矩阵进行XOR操作  
XOR the subkey of the current round with the state matrix
- 最终轮 (**Final Round**)：与主轮类似，但没有 MixColumns 步骤。  
similar with the Main Rounds, but without MixColumns step.

每一个轮密钥都由AES密钥产生

we generate a round key from the single AES key

平均搜索耗时由密钥长表示

- searching for DES?  $2^{55}$
- searching for AES (128-bit key version)?  $2^{127}$
- One time pad:  $2^{keysize}$
- RC4 40-2048 bits

多块加密：

由于块密码只有一个固定长度的密钥，所以在加密多块数据或长于块长度的数据时无法像流密码那样持续产生密钥流，如果为每个块都使用一个密钥时不可避免会遇到密钥管理问题，因此引出了加密模式的概念。

Since block ciphers have only one fixed-length key, it is not possible to continuously generate a stream of keys as in stream ciphers when encrypting multiple blocks of data or data longer than the length of a block, and if a key is used for each block it is inevitable that a key management problem will be encountered, thus leading to the notion of encryption patterns

当数据长不是块长的倍数时，需要一个填充方法

need a padding method if out message is not a multiple of the block size

## Modes of Operation - how to use Block Ciphers

**ECB (Electronic Codebook mode):**

- 显而易见

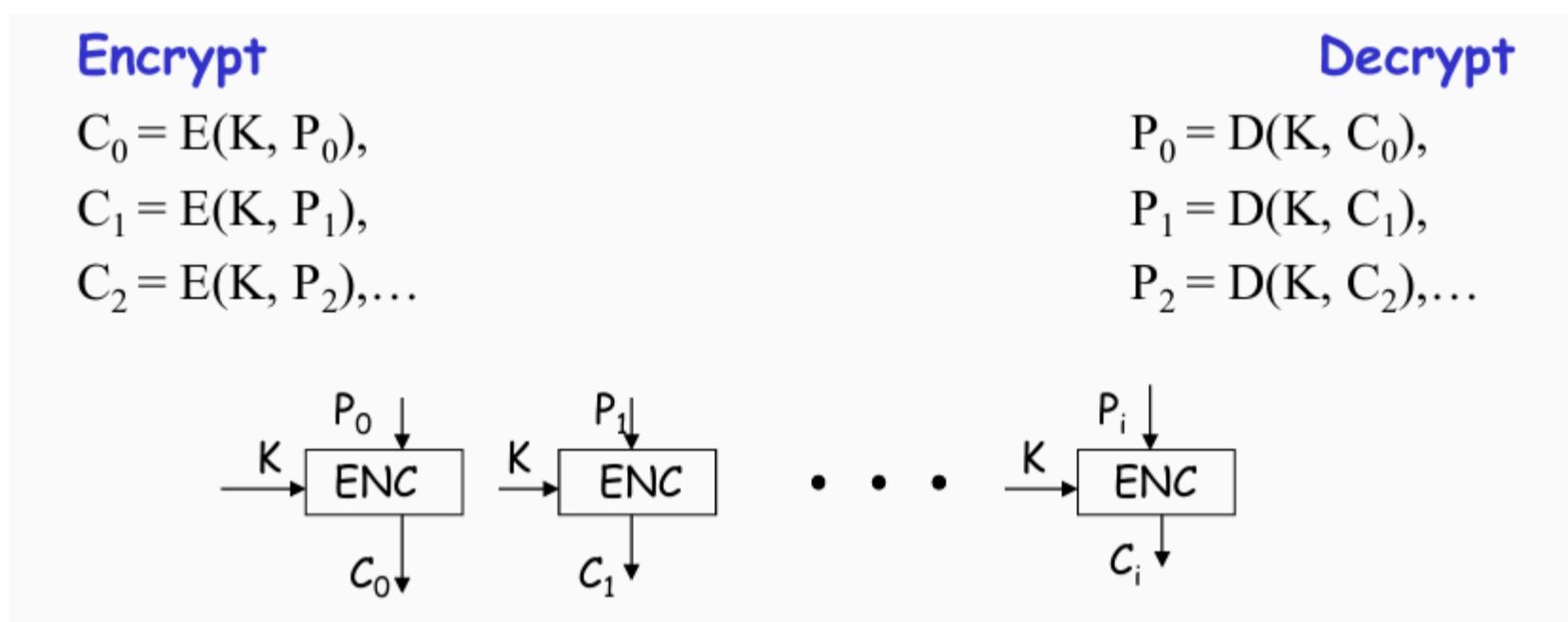
Obvious thing to do

- 独立加密每个区块

Encrypt each block independently

- 存在严重弱点

There is a serious weakness



独立加密每个消息（同个密钥），也导致消息容易被删除或更改（integrity）

如果两个加密消息相同，得到密文相同（就像一个巨大的替换密码）

**Cipher Block Chaining (CBC) mode:**

- 将块链在一起

Chain the blocks together

- 比ECB更安全

More secure than ECB

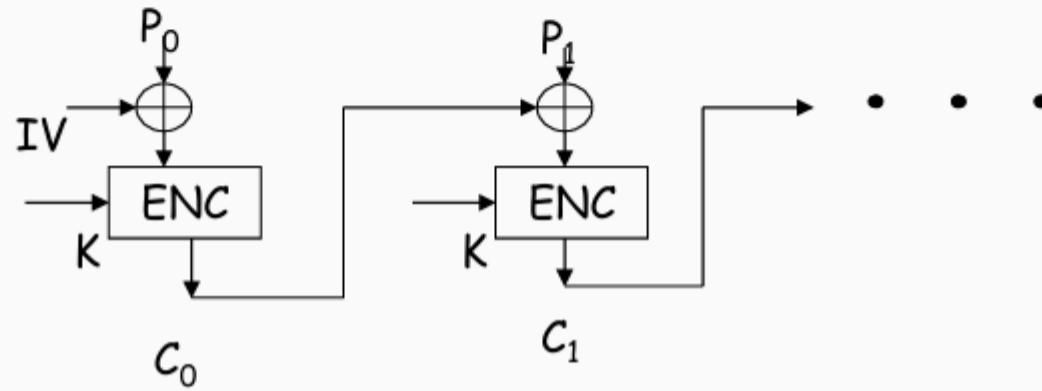
- IV is random, but is not a secret

### Encryption

$$\begin{aligned} C_0 &= E(K, IV \oplus P_0), \\ C_1 &= E(K, C_0 \oplus P_1), \\ C_2 &= E(K, C_1 \oplus P_2), \dots \end{aligned}$$

### Decryption

$$\begin{aligned} P_0 &= IV \oplus D(K, C_0), \\ P_1 &= C_0 \oplus D(K, C_1), \\ P_2 &= C_1 \oplus D(K, C_2), \dots \end{aligned}$$



使用一个随机填充值IV。由于IV的存在，每次的密文不同（但如果IV相同则仍然相同）

Use a random padding value IV. The ciphertext is different each time due to IV (but remains the same if IV is the same)

如果信息中的 P0 发生变化，所有 C 字块会发生什么变化？全部改变。（chained）

If P0 changes in the messages what happens to all C blocks? All change.

但若C0变化，只有P0和P1变化

But if C0 changes, only P0 and P1 change

好模式 (Good Mode) :

- 密文的信息依赖性

Message dependence of ciphertext

- 有限的误差传播

Limited error propagation

- 无需区块同步即可工作(如果有一个信息块丢失，对后续信息的影响应该是有限的)

Works without block synchronisation(if a block of the message goes missing, the impact on subsequent messages should be limited)

- 优化解密/加密的使用

Optimise use of decrypt/encrypt

- 减少填充物

Reduce padding

### 传输错误类型 (types of transmission errors)

- 传输错误是指通信信道中出现的错误（1 变成 0 或 0 变成 1）。

Transmission errors are errors (a 1 becomes a 0 or a 0 becomes a 1) that occur in the communication channel.

- 传输损耗是指在通信信道中丢失（从未到达）的比特。

Transmission losses are bits that get lost (they never arrive) in the communication channel.

## 错误传播 (Error Propagation)

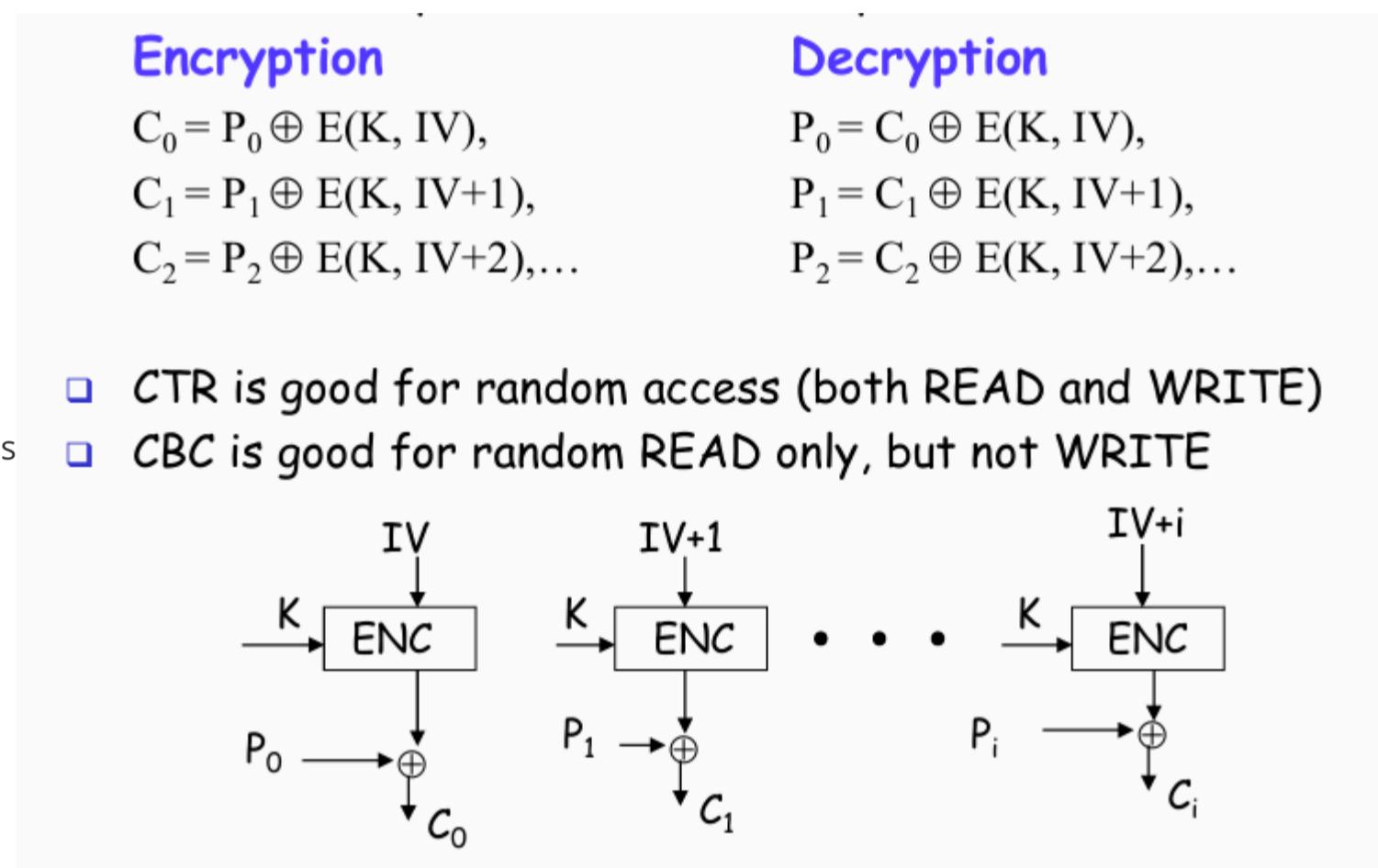
如果密文输入有一个错误位，而明文输出有一个以上的错误位，则解密过程涉及错误传播。

A decryption process involves error propagation if a ciphertext input that has one incorrect bit produces a plaintext output that has more than one incorrect bit.

- 在流密码中每个错误影响到一个对应位，而在块密码中取决于加密模式
- 错误难以预防，我们只能限制其对消息的影响，且当错误发生数量很小时，尝试修复
- 错误传播是一件坏事

## Counter Mode (CTR):

- 流密码一样的表现  
Acts like a stream cipher
- 常用于随机存取



CTR 模式要求双方之间的计数器同步（需要知道接收到哪一块）。它克服了 ECB 的弱点，因为即使是相同的明文块，由于 IV 值不同，密文也会不同。

CTR mode requires synchronisation between readers in terms of the counter. It overcomes ECB weakness as even with same plaintext block it will have different ciphertext as value of IV is different

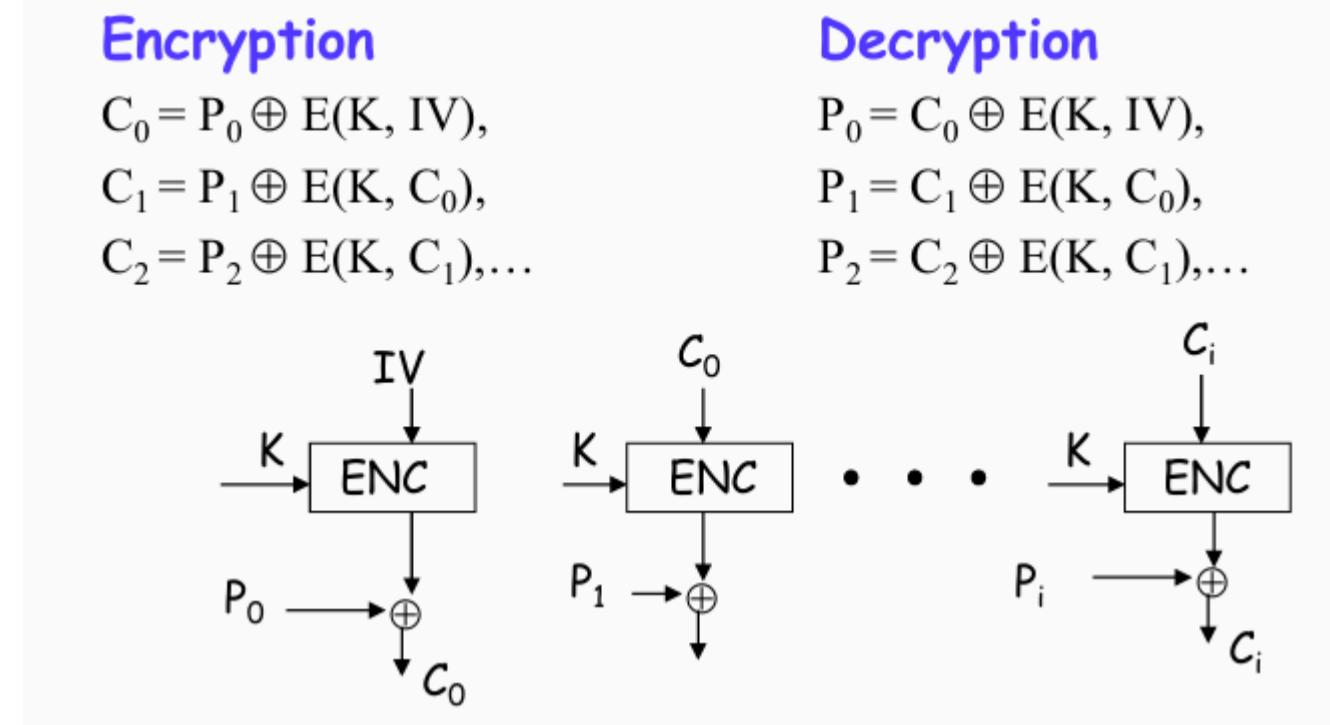
CBC 不需要同步，您只需将接收到的数据块输入解密功能，无需记录接收到了多少数据块。

如果有1位出错，CTR中会有对应1位明文出错

如果有1位丢失，所有明文都会出错

## Cipher Feedback Mode (CFB):

像流密码一样使用



使用 $C_i$ 来计算 $C_{i+1}$

E就像一个密钥流产生器

### 3. Number Theory

#### divisor (除数)

如果存在某个整数 m, 使得  $a = m * b$ , 那么说 b 除以 a (b divides a)

$b|a$

eg. 1,2,3,4,6,8,12,24 divide 24

b是a的除数(divisor)

#### Congruence(同余)

如果  $n | a - b$ , 则 a 与 b 同余。

$$a \equiv b \pmod{n}$$

$$23 \equiv 8 \pmod{5}$$

#### Modular Arithmetic(模算术)

r 是 a 除以自然数 n 的余数

r is the remainder when a is divided by a natural number n

$$18 \bmod 5 = 3$$

可以在任意点上做模算术, 即  $a + b \bmod n = [a \bmod n + b \bmod n] \bmod n$

$$97 + 23 \bmod 7 = [93 \bmod 7 + 23 \bmod 7] \bmod 7$$

#### Prime (质数)

如果一个整数 p 只有 1 和他本身作为除数, 那么他就是质数

2, 3, 5, 7, 11 是质数

质因式分解(Prime Factorization): 如果一个数是合数, 那么其可以用质数表示为  $a = p_1^{c_1} p_2^{c_2} \dots$ , 其中 p 是质数, c 是自然数

当数字较大时, 一般很难进行 (质) 因式分解

## Greatest Common Divisor(GCD)

a 和 b 的 GCD (a,b) 是同时除以 a 和 b 的最大数

GCD (a,b) of a and b is the largest number that divides both a and b

$$\text{GCD}(60,24)=12$$

如果  $\text{GCD}(a,b)=1$ , 那么说 a, b 互质

计算 GCD 常采用欧几里得算法(Euclidean Algorithm)

---

Compute  $\text{gcd}(911, 999)$  :

$$\begin{aligned} A &= q \times B + R \\ 999 &= 1 \times 911 + 88 \\ 911 &= 10 \times 88 + 31 \\ 88 &= 2 \times 31 + 26 \\ 31 &= 1 \times 26 + 5 \\ 26 &= 5 \times 5 + 1 \\ 5 &= 5 \times 1 + 0 \end{aligned}$$

模逆元：如果  $AB \bmod n = 1$ , 那么称  $A = B^{-1}$ , A 是 B 的模逆元

要求模逆元，使用扩展欧几里得算法 (Extended Euclidean Algorithm)

### Example

Let  $a = 911$  and  $b = 999$ . From the Euclidean algorithm,

$$\begin{aligned} 999 &= 1 \times 911 + 88 \\ 911 &= 10 \times 88 + 31 \\ 88 &= 2 \times 31 + 26 \\ 31 &= 1 \times 26 + 5 \\ 26 &= 5 \times 5 + 1 \quad \Rightarrow \text{gcd}(a, b) = 1 \end{aligned}$$

Tracing backward, we get

$$\begin{aligned} 1 &= 26 - 5 \times 5 \\ &= 26 - 5 \times (31 - 1 \times 26) = -5 \times 31 + 6 \times 26 \\ &= -5 \times 31 + 6 \times (88 - 2 \times 31) = 6 \times 88 - 17 \times 31 \\ &= 6 \times 88 - 17 \times (911 - 10 \times 88) = -17 \times 911 + 176 \times 88 \\ &= -17 \times 911 + 176 \times (999 - 1 \times 911) = 176 \times 999 - 193 \times 911 \end{aligned}$$

we now have

$$\text{gcd}(911, 999) = 1 = -193 \times 911 + 176 \times 999.$$

If we do a modular reduction of 999 to this equation, we have

$$\begin{aligned} 1 \pmod{999} &= -193 \times 911 + 176 \times 999 \pmod{999} \\ \Rightarrow 1 &= -193 \times 911 \pmod{999} \\ \Rightarrow 1 &= (-193 \pmod{999}) \times 911 \pmod{999} \\ \Rightarrow 1 &= 806 \times 911 \pmod{999} \end{aligned}$$

$$1 \equiv 806 \times 911 \pmod{999}.$$

Hence 806 is the **modular inverse** of 911 modulo 999.

## The Euler phi Function

For  $n \geq 1$ ,  $\phi(n)$  denotes the number of integers in the interval  $[1, n]$  which are relatively prime to  $n$ . The function  $\phi$  is called the **Euler phi function** (or the **Euler totient function**).

**Fact 1.** The Euler phi function is **multiplicative**. I.e. if  $\gcd(m, n) = 1$ , then  $\phi(mn) = \phi(m) \times \phi(n)$ .

**Fact 2.** For a prime  $p$  and an integer  $e \geq 1$ ,  $\phi(p^e) = p^{e-1}(p-1)$ .

Let  $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$  where  $p_1, \dots, p_k$  are prime and each  $e_i$  is a nonzero positive integer.

Then

$$\phi(n) = p_1^{e_1-1} (p_1-1) \cdot p_2^{e_2-1} (p_2-1) \dots p_k^{e_k-1} (p_k-1)$$

$$\cdot \phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$$

## Fermat's Little Theorem (费马小定理)

Let  $p$  be a prime. Any integer  $a$  not divisible by  $p$  satisfies  $a^{p-1} \equiv 1 \pmod{p}$ .

欧拉推广:

**Euler's Generalization** Let  $n$  be a composite. Then  $a^{\phi(n)} \equiv 1 \pmod{n}$  for any integer  $a$  which is relatively prime to  $n$ .

$$\text{E.g. } a=3; n=10; \phi(10)=4 \Rightarrow 3^4 \equiv 81 \equiv 1 \pmod{10}$$

$$\text{E.g. } a=2; n=11; \phi(11)=10 \Rightarrow 2^{10} \equiv 1024 \equiv 1 \pmod{11}$$

## Modular Exponentiation (模指数)

Square-and-Multiply Algorithm

$$\text{e.g. } 11^{15} \bmod 13 = 11^{8+4+2+1} \bmod 13 = 11^8 \times 11^4 \times 11^2 \times 11 \bmod 13 \quad - (1)$$

- $11^2 = 121 \equiv 4 \pmod{13}$  — (2)
- $11^4 = (11^2)^2 \equiv (4)^2 \equiv 3 \pmod{13}$  — (3)
- $11^8 = (11^4)^2 \equiv (3)^2 \equiv 9 \pmod{13}$  — (4)

Put (2), (3) and (4) into (1) and get

$$11^{15} \equiv 9 \times 3 \times 4 \times 11 \equiv 5 \pmod{13}$$

- performed at most  $2\lfloor \log_2 15 \rfloor$  modular multiplications
- Complexity =  $O(\lg(e))$

## 4. PKE

### Public Key Encryption

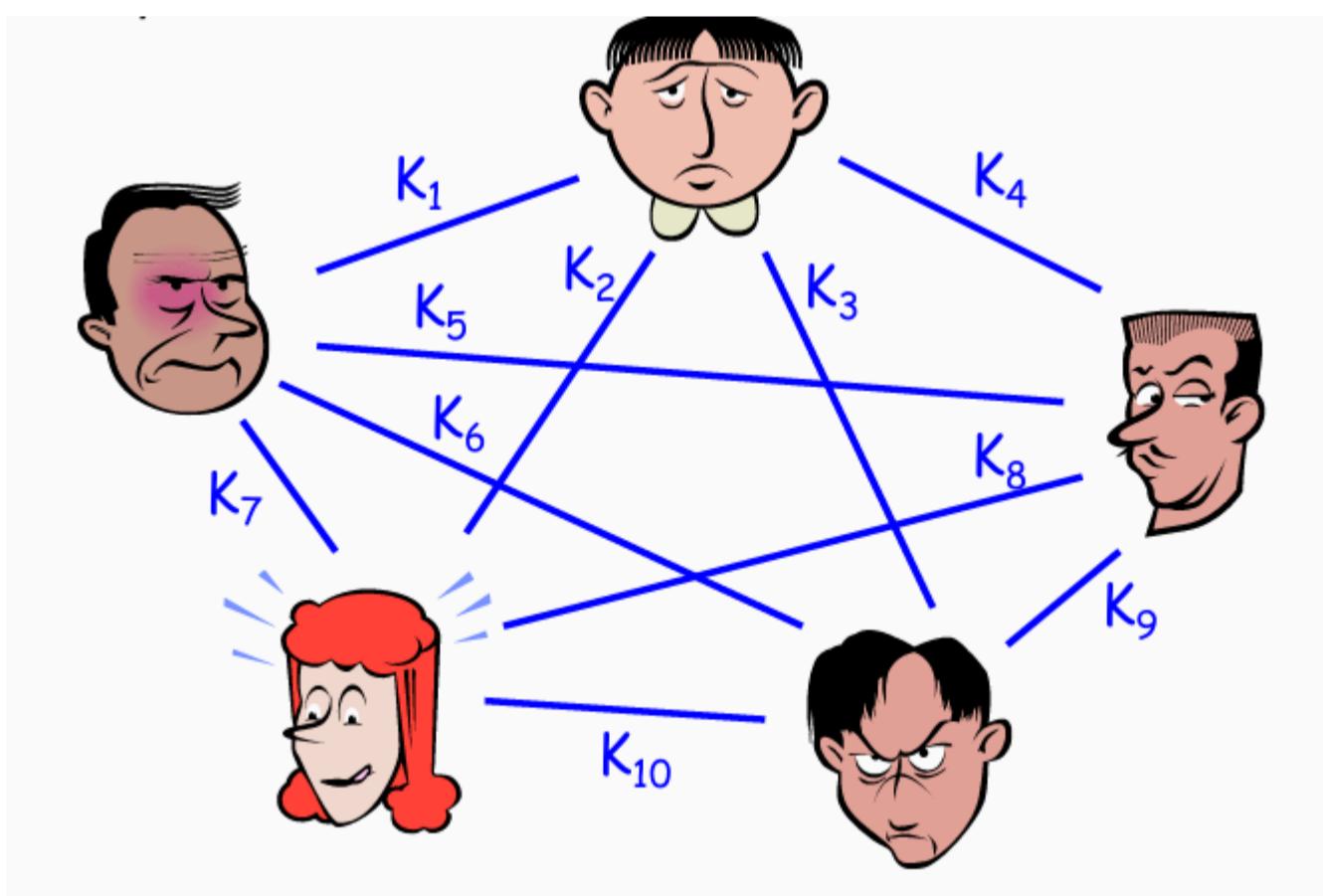
在涉及到多方通信的时候，对称加密就很不实用，因为要为每个通信对象都存储一个密钥，在有n个人的通信系统中，则共需要  $\frac{n(n-1)}{2}$

个密钥（每个人需要有n-1个）

Symmetric encryption is impractical when multiple parties are involved, because a key has to be stored for each communicating object, and in a communication system with n people, then a total of  $\frac{n(n-1)}{2}$  keys are needed (each person needs to have n-1)

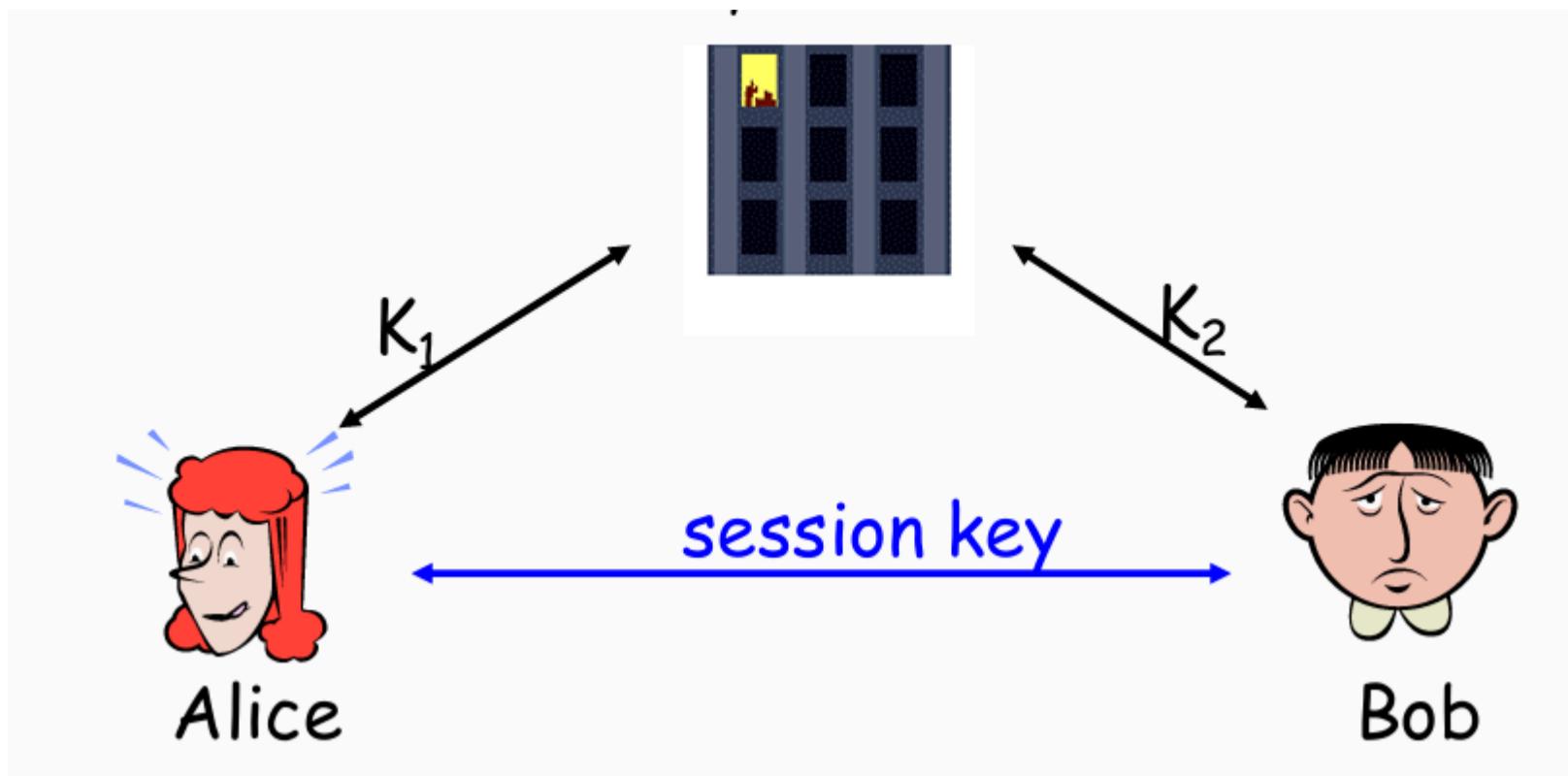
如何告知密钥也是一个难题

How to tell secret keys is also a problem



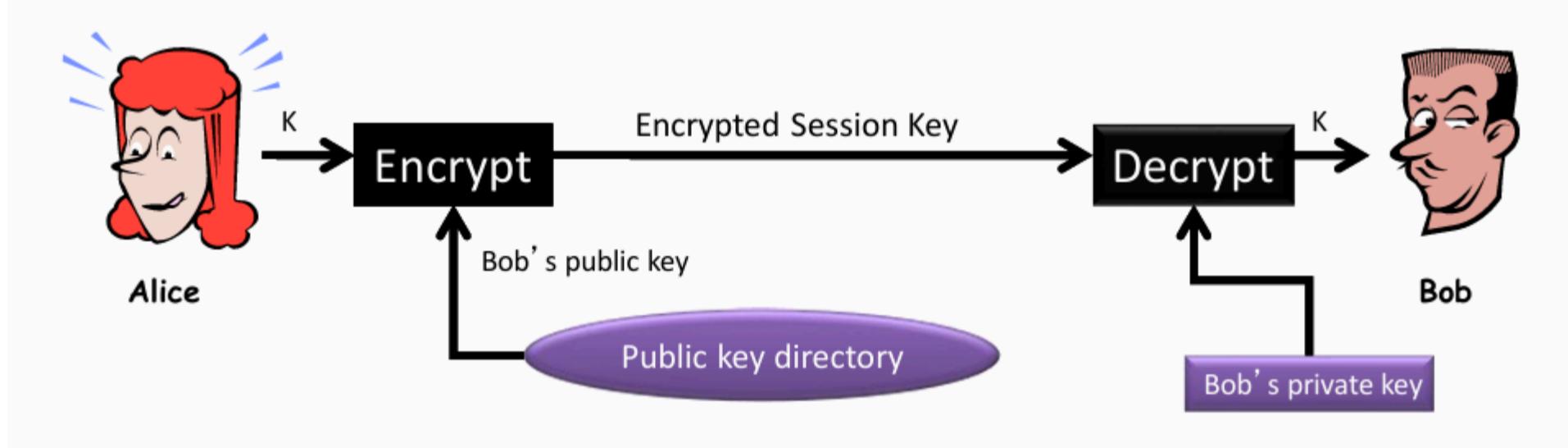
在这种情况下，需要考虑使用共享密钥。

In this case, the use of a shared keys needs to be considered.



这种情况下，只需要n个长期密钥(long-term key)存储在服务器中，每当要进行对话时，就分发session key

In this case, only n long-term keys need to be stored in the server, and the session key is distributed whenever a conversation is to take place



在公钥加密体系中，Bob 具有一个公钥和一个私钥。

- 公开公钥，他人用公钥加密消息  
publish the public key, other people use public key to encrypt message
- 保留私钥，用于解密  
keep secret key, use it to decrypt

安全需求 Security requirement:

1. 难以从密码中找到私钥或明文  
difficult to find private key or plaintext from ciphertext
2. 难以从公钥中找到私钥  
difficult to find private key from public key

## Problems

公钥加密体系基于： public key crypto based on

- 数学单向函数  
mathematical one way functions
  - 给定输入易于计算输出

Easy to compute output given the inputs

- 给定输出难于计算输入

Difficult to compute input given the output

- 因式分解问题

Factorisation problem

- 对于两个质数相乘

Multiplying two prime numbers

- 给定质数  $x$  和  $y$ , 很容易计算  $x \cdot y = z$

Given prime  $x$  and  $y$  it is easy to compute  $x \cdot y = z$

- 给定  $z$  后, 要计算  $x$  和  $y$  并不容易

Given  $z$  it is not easy to compute  $x$  and  $y$

- 离散对数问题

Discrete logarithm problem

- 数字的幂级数

Exponentiation of a number

- 给定  $a$ 、 $b$  和质数  $n$ , 是否很容易计算出  $z = a^b \bmod n$

Given  $a$ ,  $b$  and prime  $n$  is it easy to calculate  $z = a^b \bmod n$

给定  $z, a, n$  却很难计算出  $b$

Given  $z$ ,  $a$  and  $n$  it is not easy to compute  $b$

## Rivest, Shamir, and Adleman (RSA)

RSA算法是最广为人知的公钥加密算法之一

RSA is one of most well known algorithms

算法如下:

1. 随机选择两个长度大致相等的大质数  $p$  和  $q$
2. 设置  $n = pq$  ( $n$  称为公有模数)
3. 随机选择  $e$  使得  $\gcd(e, \varphi(n)) = 1$ , 也就是说两者互质
  - $e$  称为公共指数
  - $\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$
4. 计算  $d$  使得  $ed \equiv 1 \pmod{\varphi(n)}$ 
  - 也就是说  $d$  是模  $\varphi(n)$  下  $e$  的模逆元
  - $d$  被称为私有指数

随后即有

- 公钥:  $PK = (n, e)$
- 私钥:  $SK = (n, d)$

- 加密:  $C = M^e \bmod n$
- 解密:  $M = C^d \bmod n$

RSA可加密的最大消息要小于n

RSA的安全性建立在以下难题上：

### 1. 因子分解问题 (FACTORING) :

- 给定一个正整数 n, 找到它的素因数分解；即写成  $n = p_1^{e^1} p_2^{e^2} \dots p_k^{e^k}$  其中  $p_i$  是素数且每个  $e_i \geq 1$
- 示例:  $72 = 2^3 \cdot 3^2$

### 2. RSA问题 (RSAP) :

- 给定:
  - 一个正整数 n, 它是两个不同等长素数 p 和 q 的乘积,
  - 一个正整数 e, 使得  $\gcd(e, (p-1)(q-1)) = 1$
  - 从  $Z_n^*$  中随机选择的一个整数 c
- 找到一个整数 m 使得  $m^e \equiv c \pmod{n}$ 。注意: p 和 q 不给定。 (也就是对于RSA加密中, 给出密文, n, e, 找出明文)

以上两个问题相关但没有被证明

- RSAP的不可解性构成了RSA公钥密码系统安全性的基础。
- RSAP与因子分解问题密切相关但不等价。
- 如果能解决FACTORING, 则可以解决RSAP。

以上难题导致了暴力破解RSA是不可行(not feasible) 的

<b>Protection Lifetime of Data</b>	<b>Present – 2010</b>	<b>Present – 2030</b>	<b>Present – 2031 and Beyond</b>
<b>Minimum symmetric security level</b>	80 bits	112 bits	128 bits
<b>Minimum RSA key size</b>	1024 bits	2048 bits	3072 bits

建议的最小RSA密钥长度如上, 且需要不断更新

## ElGamal Encryption

选择一个大素数p, 有  $Z_{p-1} = \{0, 1, 2, \dots, p-2\}$ 。再选择一个本原根  $g \in Z_p$ , 使得  $Z_p$  内所有元素都可以用  $g$  表示

- 私钥:  $x \in_R Z_{p-1}$
- 公钥:  $y = g^x \pmod{p}$

加密:

1.  $r \in_R Z_{p-1}$
2.  $A = g^r \pmod{p}$
3.  $B = M * y^r \pmod{p}$ , 其中  $M \in Z_p^*$  是消息

密文=(A, B)

解密:

1.  $K = A^x \pmod{p}$
2.  $M = BK^{-1} \pmod{p}$

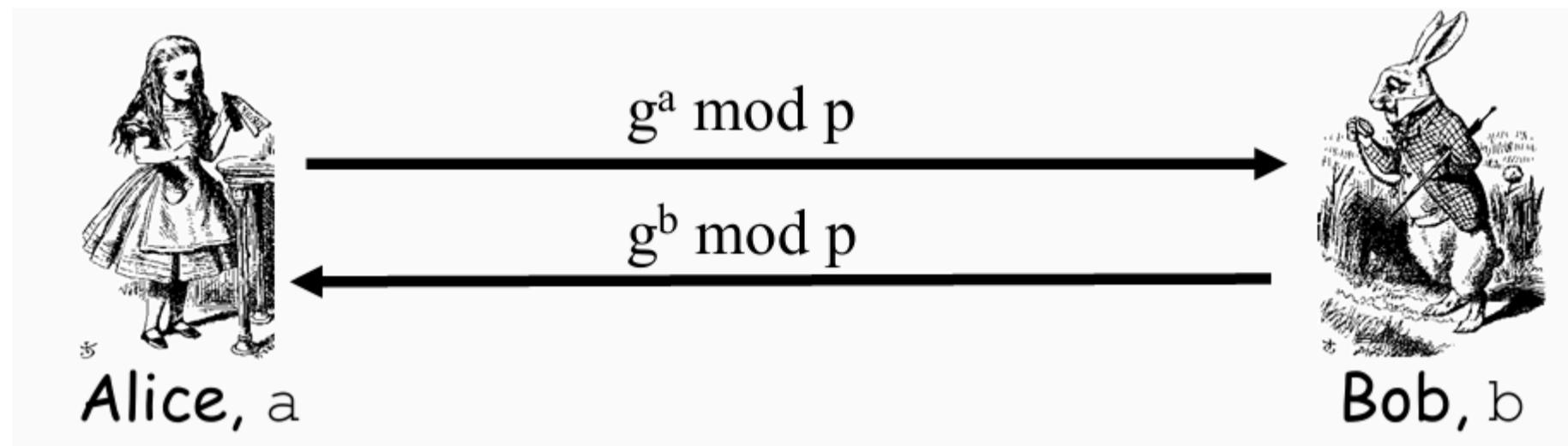
ElGamal的安全性建立在离散对数问题上, 假如离散对数问题被破解, 则可由  $y = g^x \pmod{p}$  得到私钥x

注：在RSA加密中，对同段明文分别加密两次会得到两段相同的密文，因为 $C = M^e \bmod n$ , e和n保持不变  
而在ElGamal中，由于引入了随机数r，对同一明文的两次加密会分别得到不同的密文

## Diffie-Hellman

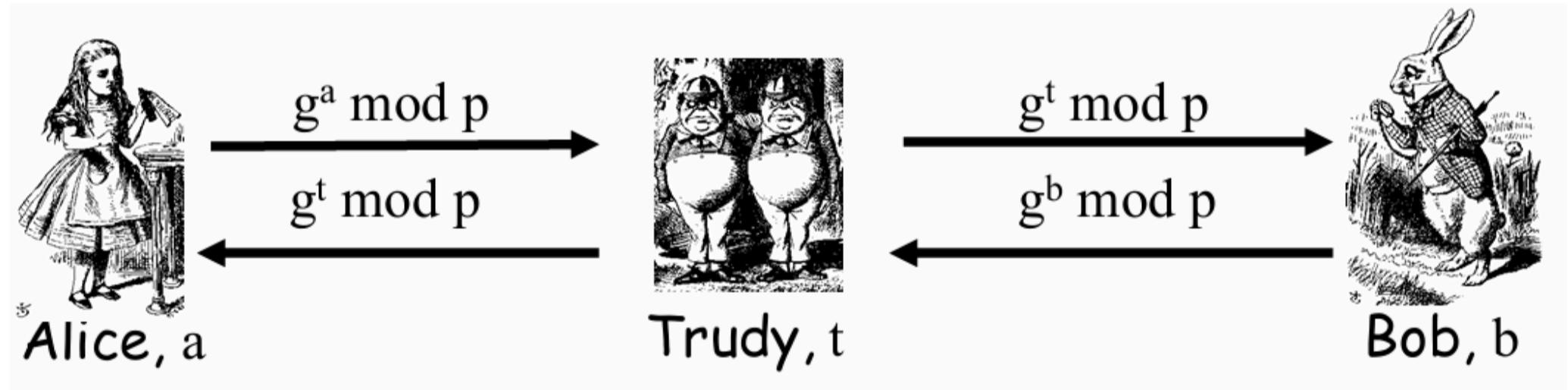
是一种密钥交换协议，而不是加密算法

同样建立在离散对数问题上



随后两人分别计算 $g^{ab}$ 作为共享密钥

但要小心中间人(Man-in-the-Middle)攻击



两人分别拿到了 $g^{at}$ 和 $g^{bt}$ 作为共享密钥，密钥交换失败！

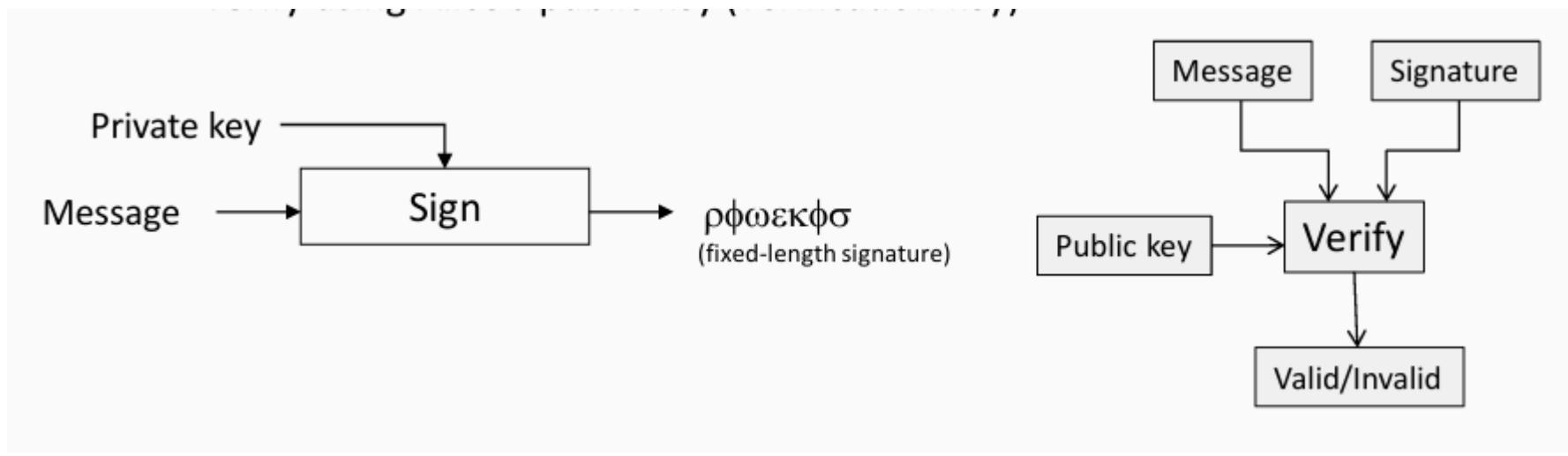
## 5.Integrity

### Electronic Signature

事实上加密并不保护完整性

加密无法检测到消息被修改、删除或添加

电子签名的作用是确认消息来源以及解决不承认消息的纷争



使用非对称加密

只有一方可以签名

- 使用秘钥签名
- 使用公钥验证

## RSA Signature Scheme

RSA可以作为一种签名算法

准备Setup阶段:

- $n = pq$ , 其中 p、q 均为大质数 (例如各为 512 位长)
- $ed = 1 \bmod (p-1)(q-1)$
- 签名秘钥 Signing (Private) Key d
- 验证秘钥 Verification (Public) Key : (e,n)

签名产生 Signature Generation阶段:

$$S = M^d \bmod n$$

签名验证Signature Verification阶段:

如果  $S^e \bmod n = M$ , 则消息有效

然而使用RSA则要求了加密的最大消息M必须小于n

而由于PKE体系的低效率, 我们不能将消息分割再分别加密

必须要很大的n才能保证足够的安全性, 但很大的n也会带来低效率问题

## Hash Function

由于用RSA直接对消息签名的弊端, 我们考虑对消息M生成一个摘要, 随后对摘要进行签名

也就是使用哈希函数hash function生成摘要

- Alice发送  $S = \text{Sign}(SK_{Alice}, h(M))$  给Bob
- Bob验证  $\text{Verify}(PK_{Alice}, h(M), S)$  是否有效

但哈希函数本身不提供任何安全服务, 因为任何人都可以使用它

哈希函数应该有以下特性:

两个功能性属性:

- **压缩功能**: 任意长度的输入映射到固定的小长度输出。

Compression – arbitrary length input to output of small, fixed length

- **易于计算**: 期望运行速度快

Easy to compute – expected to run fast

三个安全属性:

- **单向性** One-way : 给定哈希值  $y$ , 难以找到一个  $x$ 使得  $h(x)=y$ 。 (也称为前像抗力)
- **第二前像抗力** Second pre-image resistance: 给定  $y$  和  $h(y)$ , 无法找到另一个  $x$ 使得  $h(x)=h(y)$
- **碰撞抗力** Collision resistance: 难以找到两个不同的输入  $x$ 和  $y$ 使得  $h(x)=h(y)$

但由于输入空间远大于输出空间, 碰撞理论上一定存在, 但难以找到

第二前像抗力和抗碰撞力的区别:

对于第二前像抗力, 给定了消息和哈希值, 要寻找另一个具有相同哈希值的消息

对于碰撞抗力, 没有给定消息, 可以任意选择消息但要求他们具有相同的哈希值

安全属性中, 单向性是最容易实现的, 抗碰撞性是最难实现的。攻击者常常从碰撞角度攻击, 而不是单向性角度攻击  
即使攻击者可以破坏抗碰撞性, 单向性仍然难以破坏

一个好的哈希, 哪怕消息产生了极其细微的变化, 结果中也会大幅体现

对于每一个具有碰撞抵抗性的哈希函数  $h$ , 其输出长度是固定的

攻击者可以通过暴力破解的方式来尝试破坏哈希函数的碰撞抵抗性。具体方法是反复选择随机值  $x$ , 计算  $h(x)$ , 并检查是否与其他之前计算过的哈希值相等

寻找碰撞和穷举密钥的开销是不同的。

如果目标是找到任何两个不同的输入  $x$ 和  $y$ , 使得  $h(x)=h(y)$ , 则适用生日问题。安全的  $N$ 位哈希函数需要  $2^{N/2}$  的工作量来“打破”(其实也就是  $\sqrt{2^N}$ )

而对于穷举密钥, 则最多需要  $2^N$  来打破

我们通常不采用块密码来作为哈希函数, 因为:

- 块密码的块长短于哈希函数的输出长, 安全性不足
- 使用块密码作为哈希函数, 可能会导致更容易出现碰撞

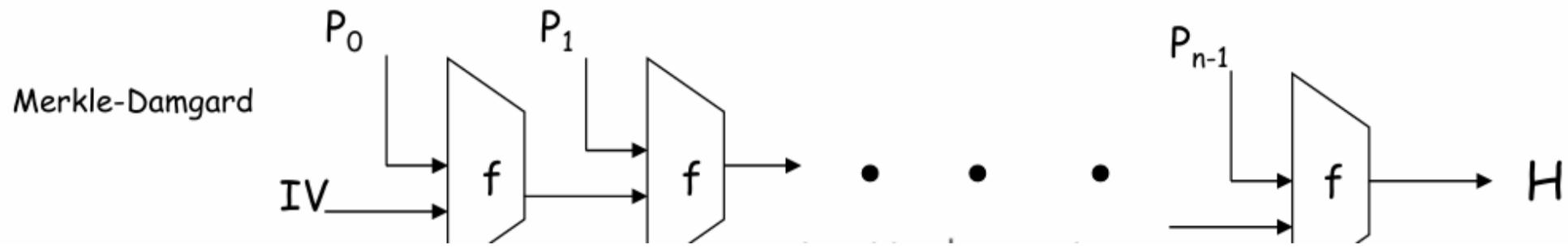
哈希函数的设计理念:

两种结构都需要首先将消息分割成固定长度的块

**Merkle-Damgard** 结构:

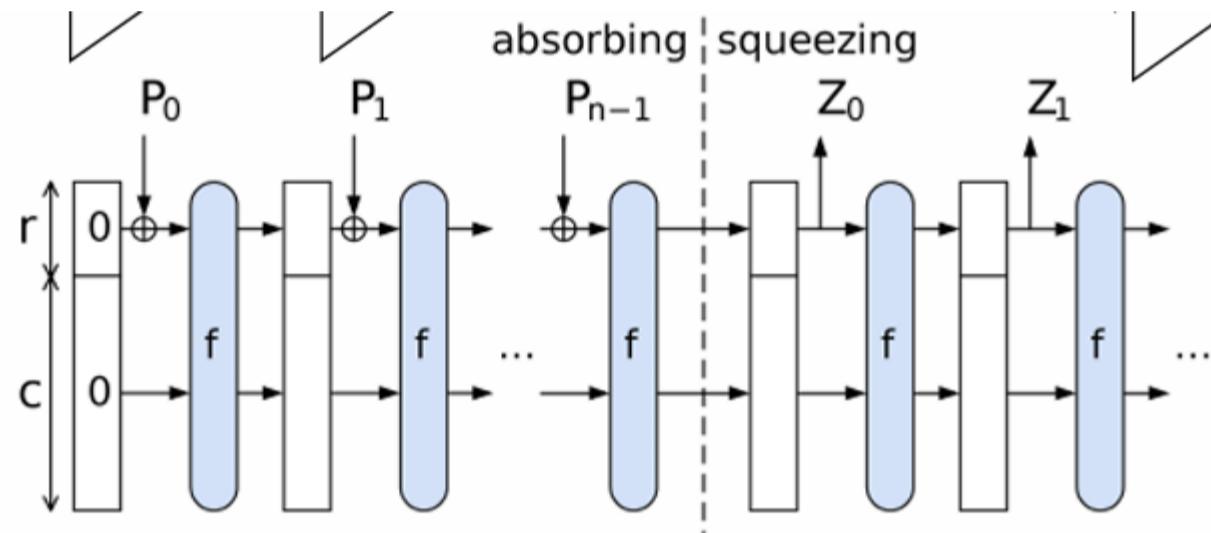
使用一个压缩函数f，对每一组输入块进行迭代处理。初始值IV (Initial Vector) 初始化第一个压缩函数，得到的输出用于下一组块的处理

Each set of input blocks is processed iteratively using a compression function f . Initial Vector (IV) initialises the first compression function and the output obtained is used for the processing of the next set of blocks, the



**Sponge**结构：

- 吸收(absorb): 输入块会被逐个吸收进状态变量中 The input blocks are absorbed into the state variables one by one
- 挤压(squeeze): 从状态变量中提取出哈希值H Extract the hash value H from the state variable



如何将哈希函数应用到签名中呢？

发送方计算消息的哈希值，并对哈希值进行签名，将签名和消息一起发送

接收方对消息再次进行哈希值计算，用于签名的验证

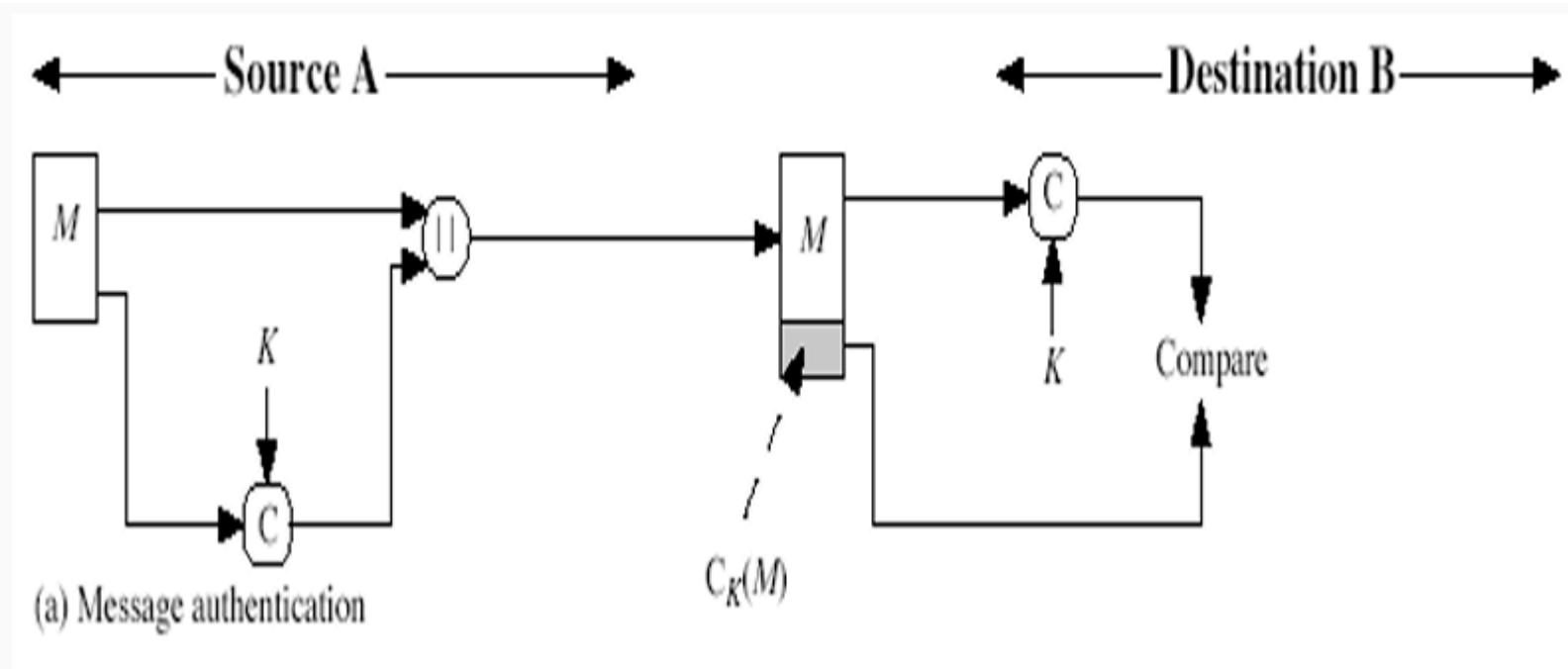
常见的哈希函数为MD5(128位输出)

## Message Authentication

确保所发即所收，检测未经授权的数据修改

加密只提供保密性，不提供完整性。

MAC 是一种**对称加密**机制，双方共享密钥用于生成验证码和验证



1. 发送方使用密钥和消息生成一个消息认证码 MAC Tag，随后和消息一起发送给接收方
2. 接收方使用Tag和消息验证是否完整

与哈希函数比较：

- 两者都能将任意长的信息映射到固定长度的输出中  
Both maps arbitrarily long message to fixed length output
- 任何人都可以不需要密码计算哈希，而MAC的计算和验证需要密钥

与数字签名比较：

- 对计算机更快--对称加密/哈希算法比签名更快  
Faster to computer- symmetric encryption/hash faster than signing
- MAC 不提供不可抵赖性non-repudiation
  - 因为双方都可以生成MAC，而不像数字签名那样只有一方可以签名
  - 使用数字签名来提供不可抵赖性non-repudiation

最常用的MAC是CBC-MAC，生成的最后一个密钥块作为MAC

由于CBC是链式密码，加入明文的任何一个消息被篡改，那么最后的MAC码也会被修改，完整性的检测成功

但以此方法构造的MAC容易受到已知明文攻击

**E.g. Given two pairs of (message, MAC tag)**

- $(P', T')$  and  $(P'', T'')$  where

$$P' = P_1, P_2$$

$$P'' = P_3$$

- Attack: anyone can forge a message and have correct MAC tag  $(P''', T''')$  without knowing the MAC key by setting  $P''' = P_1, P_2, P_3 \oplus T'$  and  $T''' = T''$ .

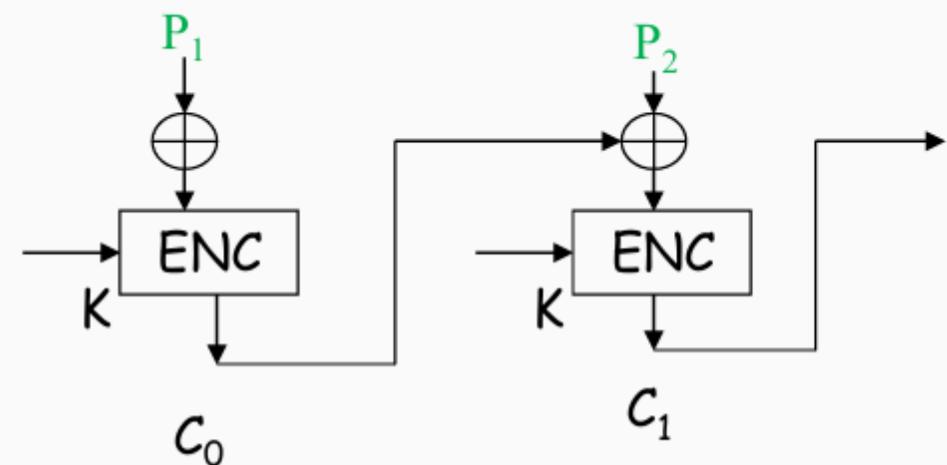
P'

$$C_0 = E(K, IV(00) \oplus P_1),$$

$$T' = MAC = C_1 = E(K, C_0 \oplus P_2)$$

P''

$$T'' = MAC = C_0 = E(K, IV(00) \oplus P_3)$$



P'''

$$C_0 = E(K, IV(00) \oplus P_1),$$

$$T' = C_1 = E(K, C_0 \oplus P_2)$$

$$C_2 = E(K, C_1 \oplus (P_3 \oplus T')) = E(K, T' \oplus (P_3 \oplus T')) = E(K, P_3) \text{ since } T' = C_1$$

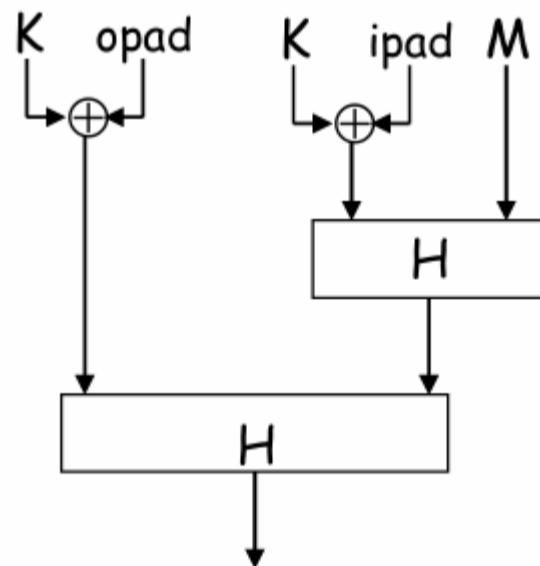
So  $T''' = C_2 = T''$

New message is  $P_1 P_2 P_3 \oplus T'$  with valid MAC  $T''' = T''$

## HMAC

由于哈希函数的高效和快速，以及给出良好输出长度。因此，有人建议我们用哈希而不是对称加密来构建 MAC。

我们需要将密钥整合进哈希函数计算中



$K \oplus opad$  and  $K \oplus ipad$  can be viewed as two generated keys.

$K \oplus ipad$  and message hash result are then hashed again with  $K \oplus opad$ .

## 6.Authentication

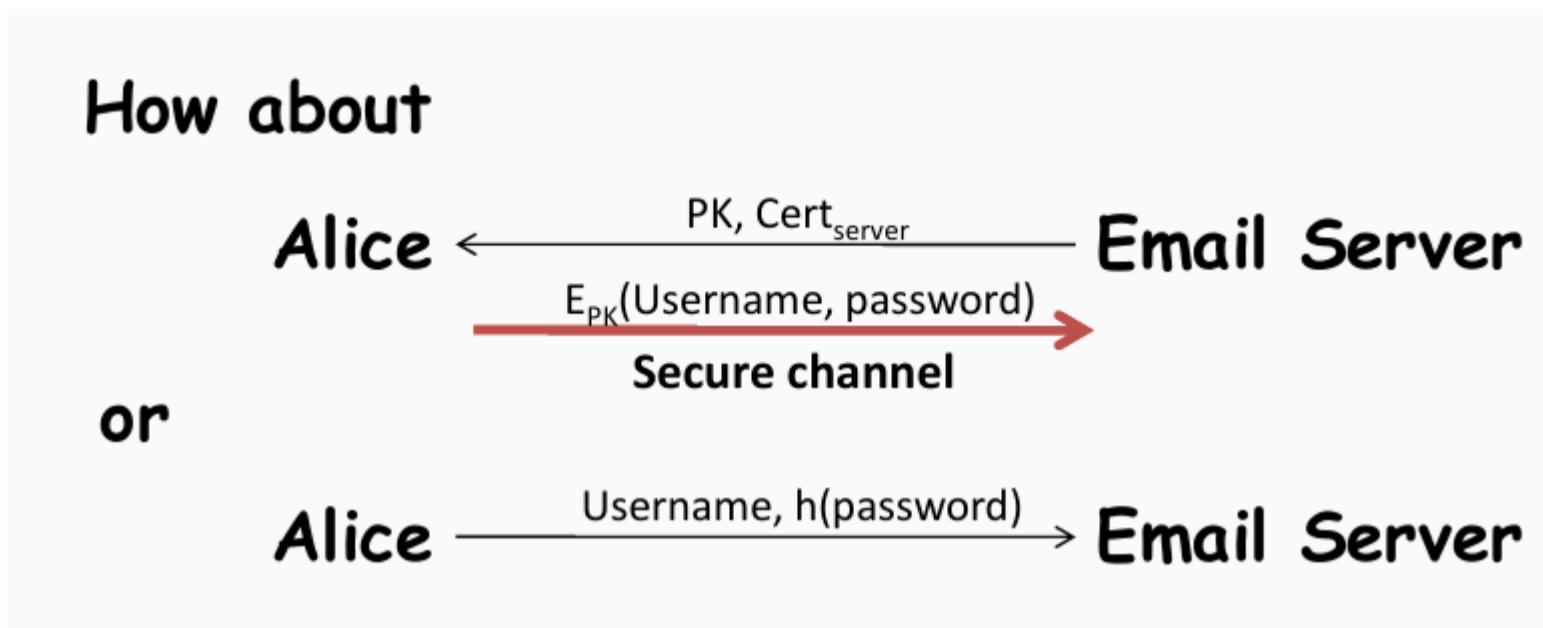
身份验证在封闭的物理环境中很简单，而在开放的网络环境中比较困难

### Introduction

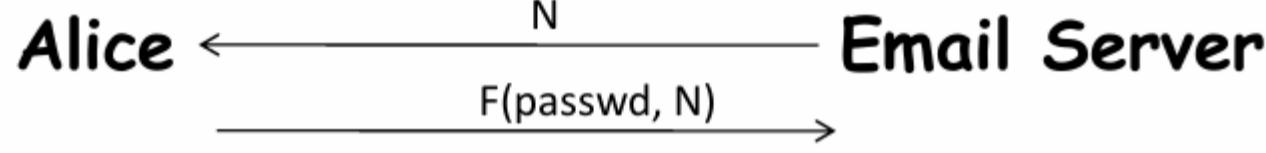
在开放环境中可能存在窃听者 (eavesdroppers)



如果窃听者窃取了用户名和密码，那么就可以登录Alice的账号



即使采用加密或hash，窃听者仍然可以在不得知密码的情况下通过重放值登录到服务器



一个可行的方法是引入nonce值N，其为一个仅使用一次的值，可以不为随机值

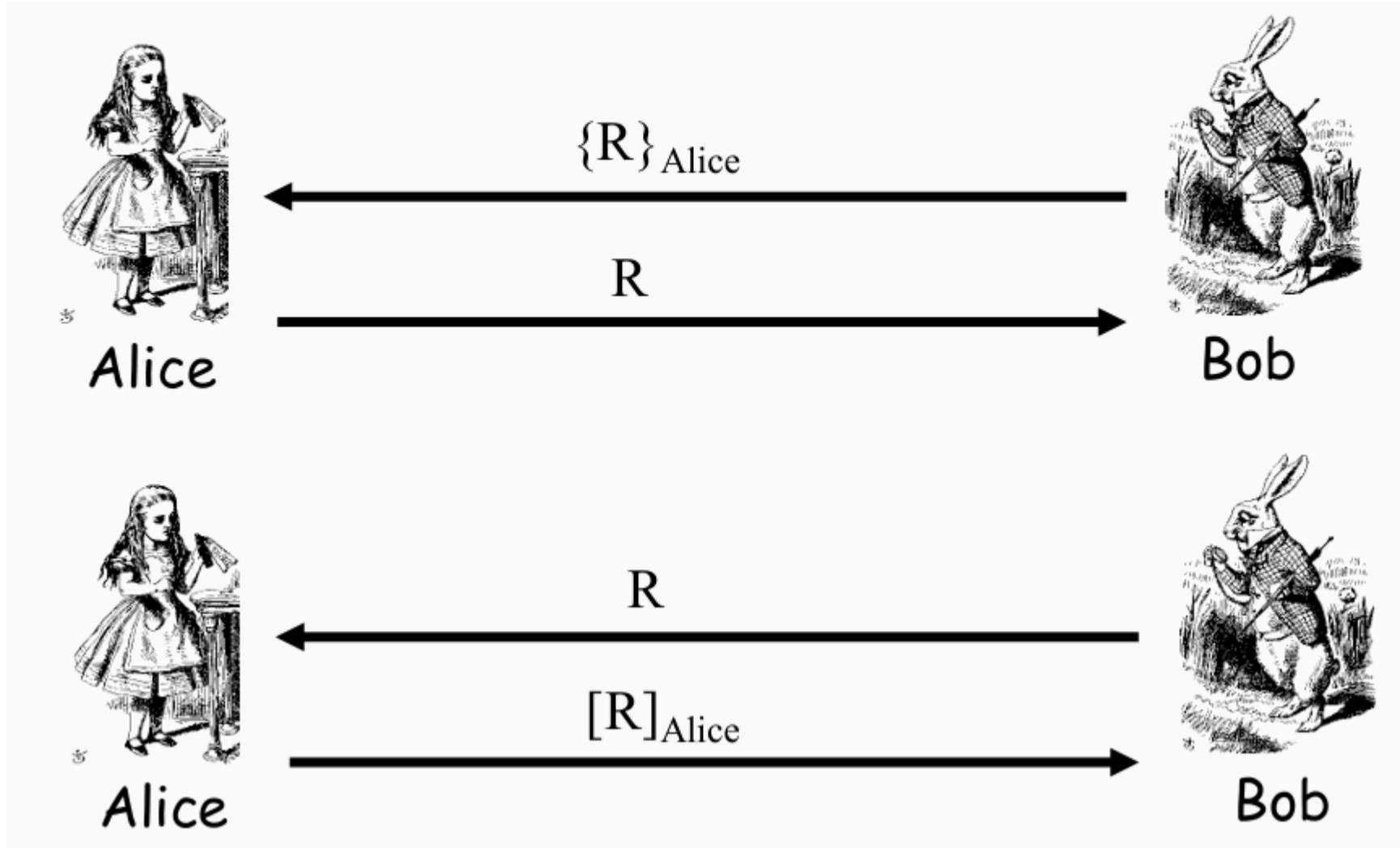
使用单向函数F（可以为hash或块加密）立即计算出值并交给服务器，以此完成登录

由于这个过程涉及到立刻产生的值N，所以窃听者无法重放

我们称N为freshness

**freshness+数据源身份验证=实体身份验证**

接下来设 $\{M\}_{Alice}$ 表示用Alice的公钥进行加密， $[M]_{Alice}$ 表示用Alice的私钥进行签名



图中两种情况都可以确认是Alice进行发送

## Entity authentication

定义：

claimant: 一个声明身份的实体

principal: claimant声明的身份

verifier: 验证声明的实体

实体验证协议是指在claimant和verifier之间传递的一系列消息，旨在确认claimant的身份

Unilateral authentication: 单方面身份验证，只为一个实体向另一方提供身份验证

Mutual authentication: 相互身份验证，双方互相验证身份

verifier只需要发送和接收消息来验证数据，他需要确定：

- 消息来自于真正的principal (数据源身份验证)
- 消息是刚刚产生的 (freshness)

MAC消息验证码和数字签名都可以提供数据源验证

而加密必须使用额外机制来提供验证：

加密前给消息加入MDC (manipulation detection code)， MDC是一个已知的数据。随后加密并发送

如果接收方解密得到的MDC正确，则证明发送方身份正确

## Freshness

有两种方法验证消息是否刚刚产生：

- freshness
- 时间戳 time stamp

使用时间戳来验证消息时效性要求双方具有一个同步的时钟，但提供这样的时钟不容易  
由于网络延迟等原因，通常会考虑设置一个窗口，在这样的接受窗口内接收时间戳则验证时效性通过  
但需要记录下已接收的消息，并拒绝重复消息的接收以防止攻击者在短时间内重放

为避免实际时间戳的缺陷，可以考虑逻辑时间戳，即序号体系：

每个实体都保存一个计数器，记录从特定实体接收和发送的消息数量

$N_{AB}$ 表示A发送给B的消息数量 (A和B都知道该值)

每当A发送给B一个消息时， $N_{AB}$ 都增加1

对于序号为n的消息，如果：

- $n \geq N_{AB}$ , 消息是新鲜的 (如果n大于则说明可能有丢包)
- $n < N_{AB}$ , 消息不新鲜，丢弃

另一种方法是使用新鲜值nonce，一个一次性数字

Alice生成一个全新随机数发送给B，B在其回复中包含该值证明消息是新鲜的

nonce一定要确保是从未使用过的，如果有重复那么可能就会面临重放攻击

## Authentication attacks

身份验证协议的安全属性通常由其能抵抗特定攻击的能力来定义

masquerade attack (伪装攻击)：攻击者直接尝试生成消息来伪装成其他人

源身份验证机制提供防护

replay attack (重放攻击)：旧消息重放发送给验证者

freshness提供防护

reflection attack (反射攻击)：验证者生成的数据被重新发送回他自己

通过显示消息的标识符来防护

## Samples

A, B: 实体A和B的标识符

$T_A$ : A生成的时间戳

$R_A$ : A生成的随机值

$K_{AB}$ : A与B之间的共享密钥

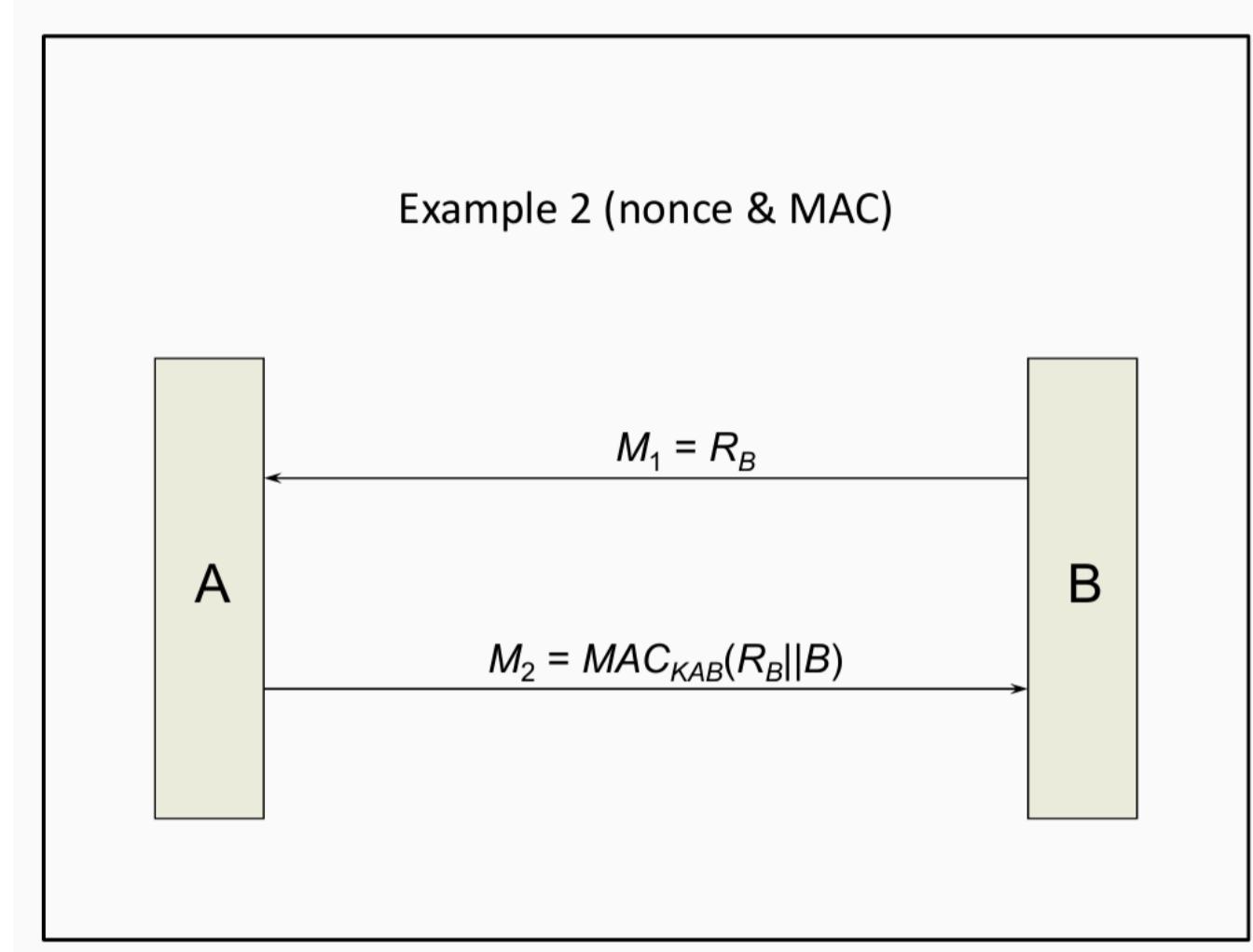
Text: 一段文本，MDC属于此类

$Enc_{A_{AB}}(X)$ : 使用共享密钥加密X

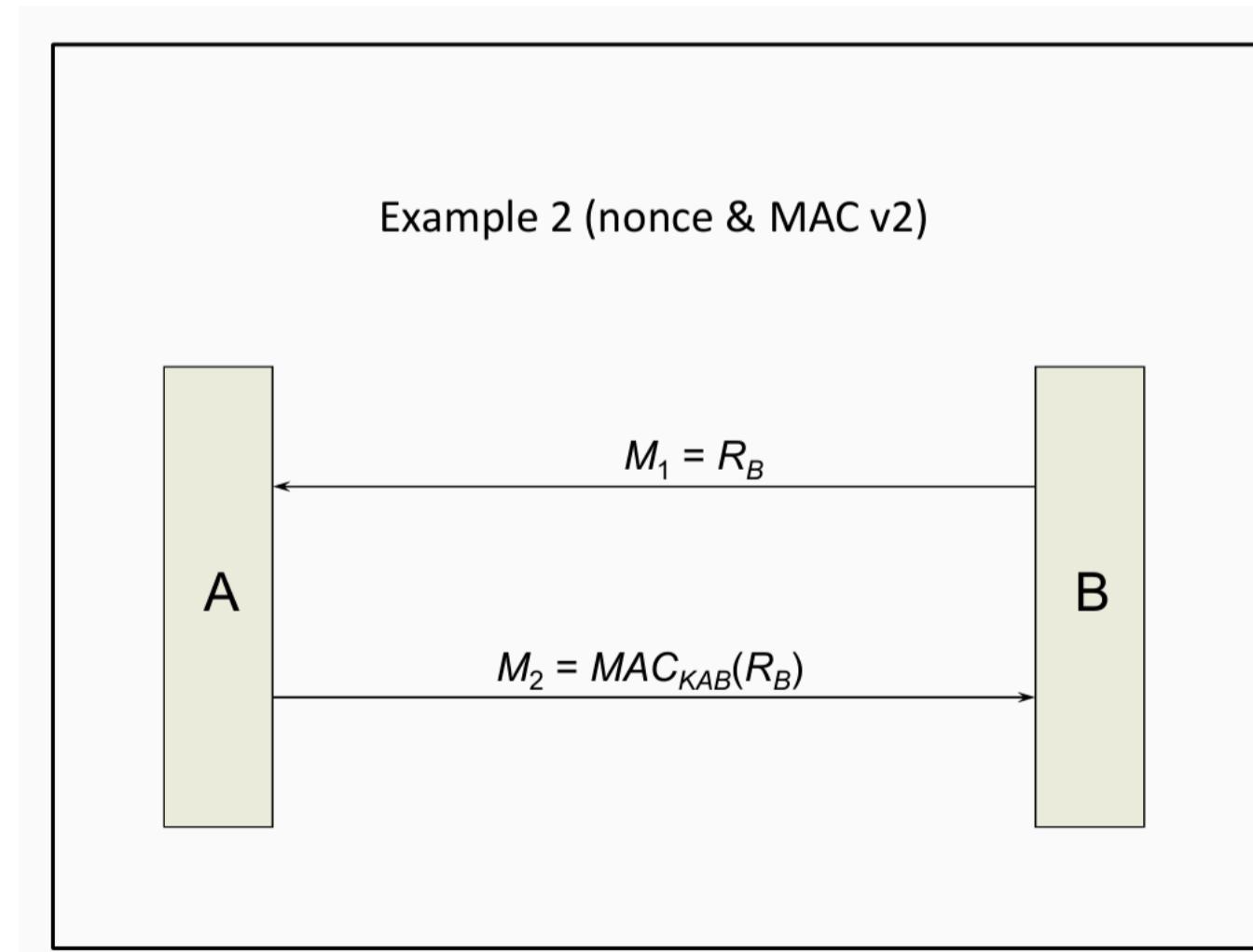
$MAC_{K_{AB}}(X)$ : 使用共享密钥对X生成消息验证码

$Sig_A(X)$ : 由A对X进行加密

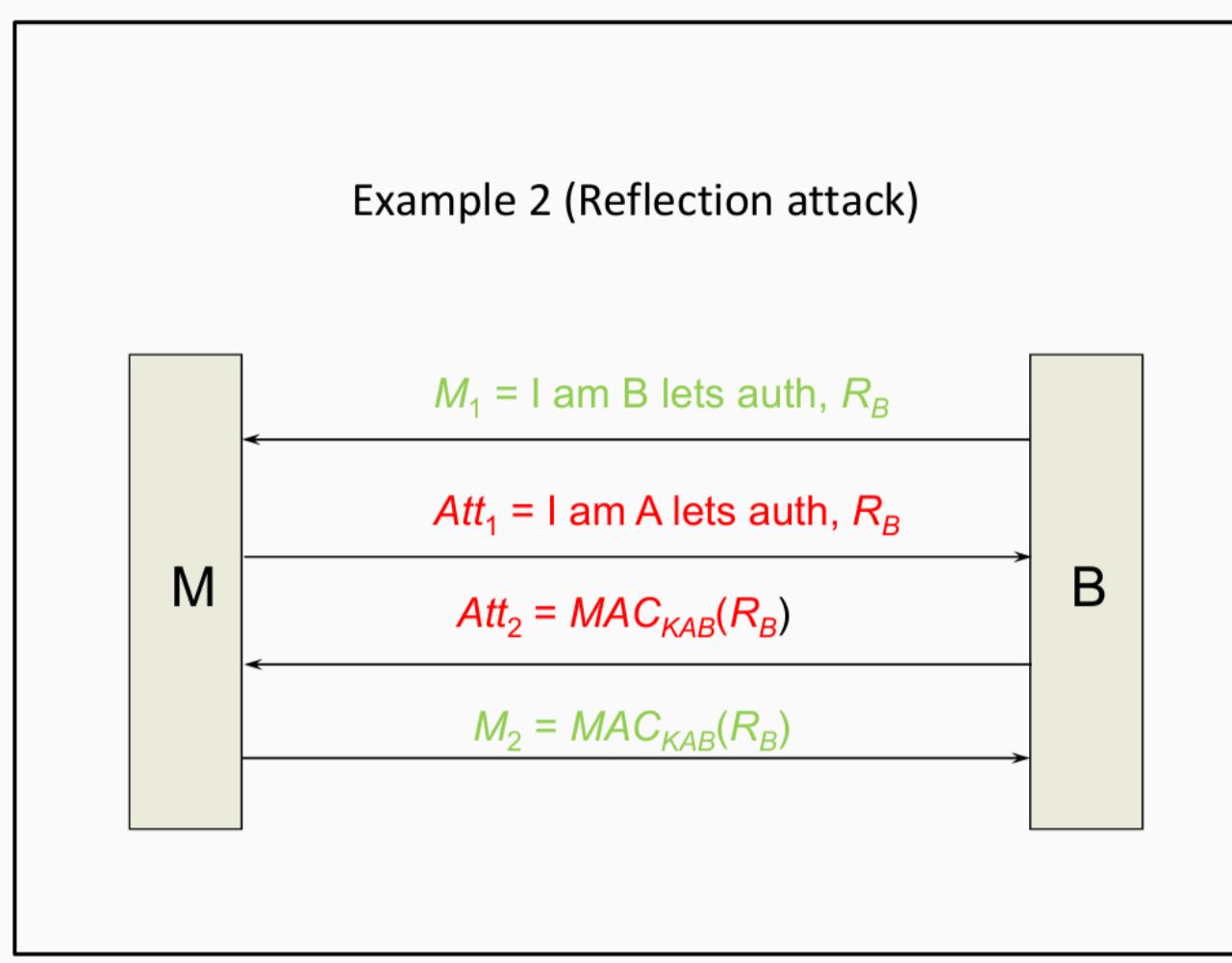
使用标识符来防止反射攻击



如果没有标识符的存在

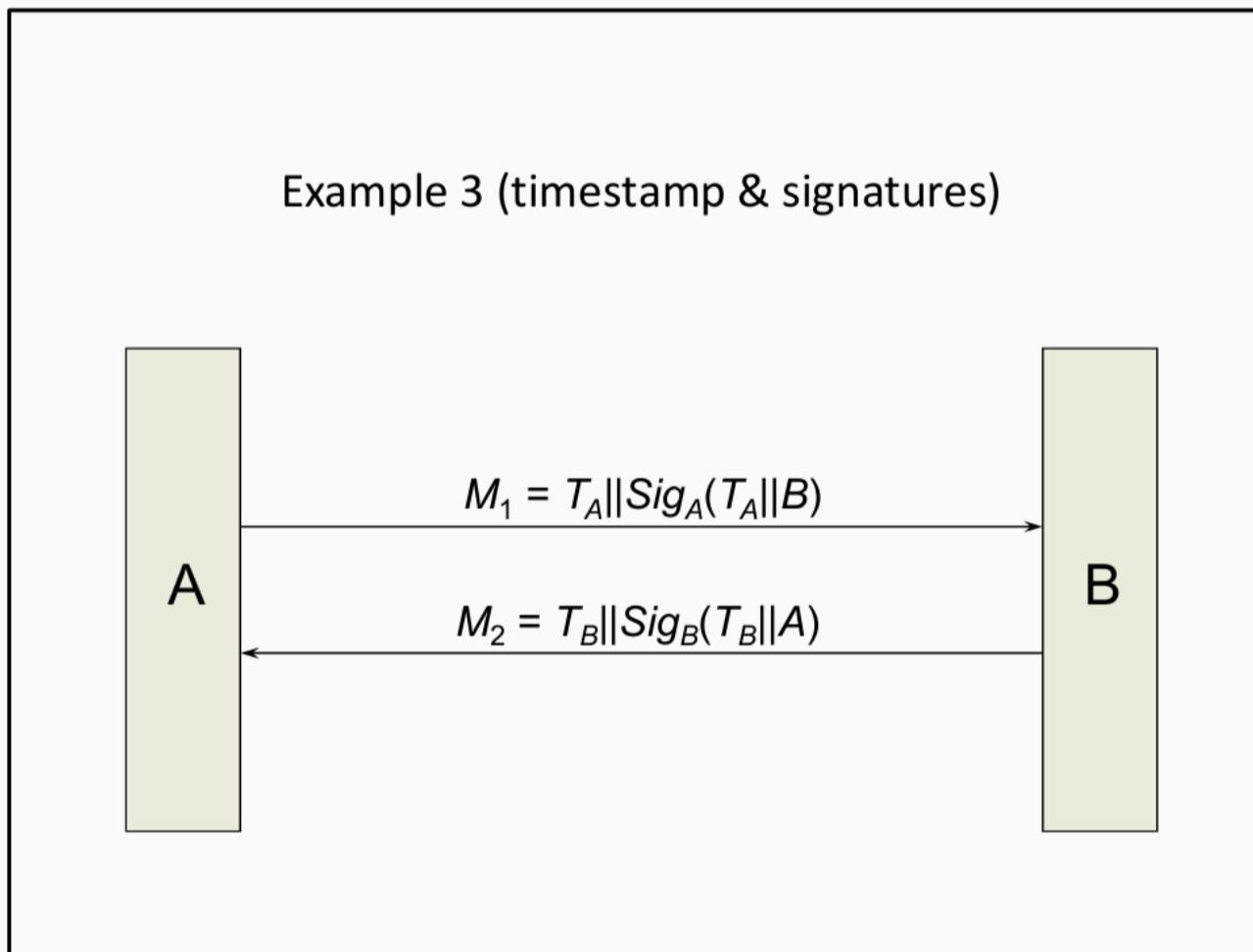


那么就会存在反射攻击:



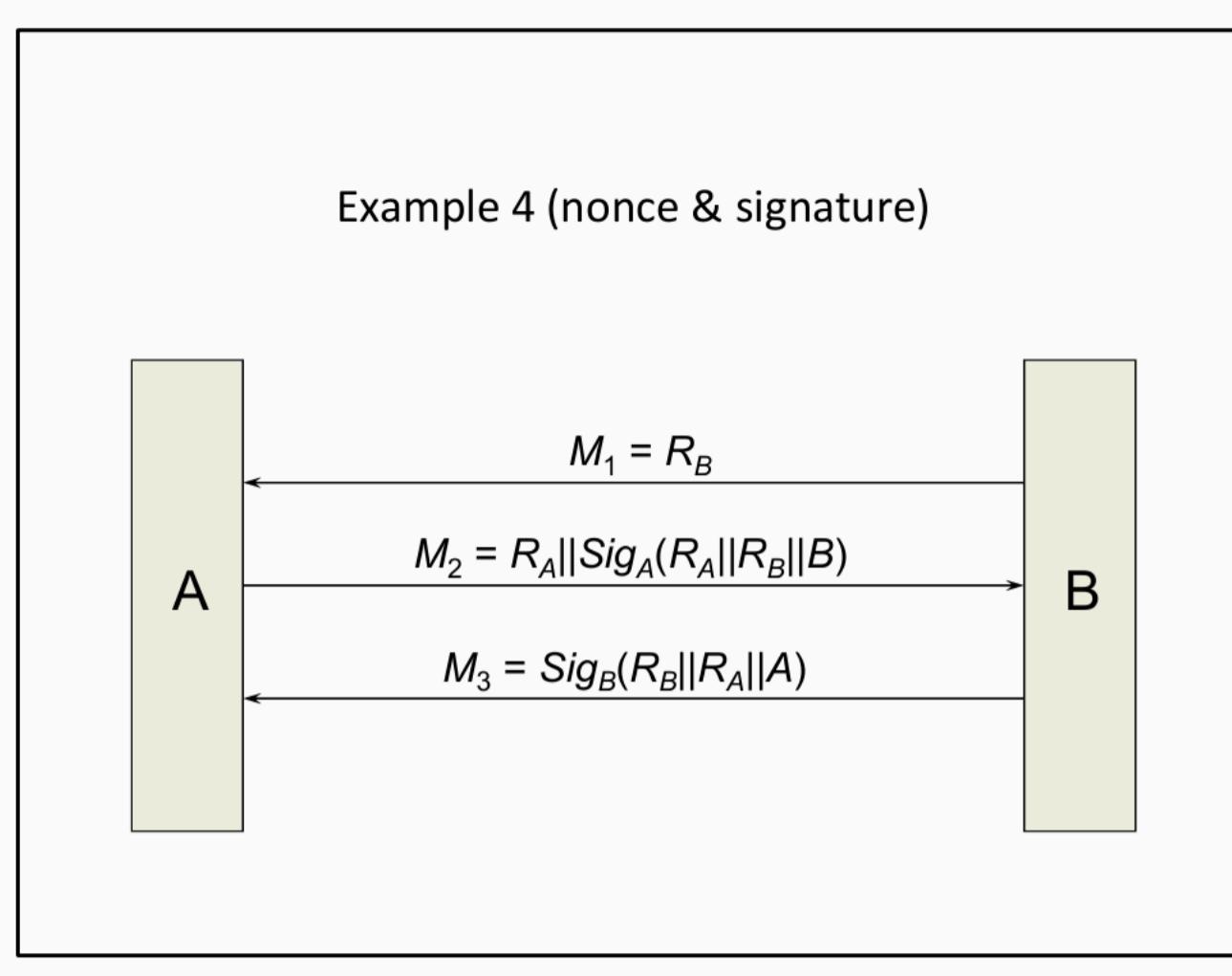
然而如果将MAC替换成Sig，那么不需要标识符也可以防止反射攻击，因为签名本身就有标识的作用

以下协议既提供了来源认证又提供了freshness



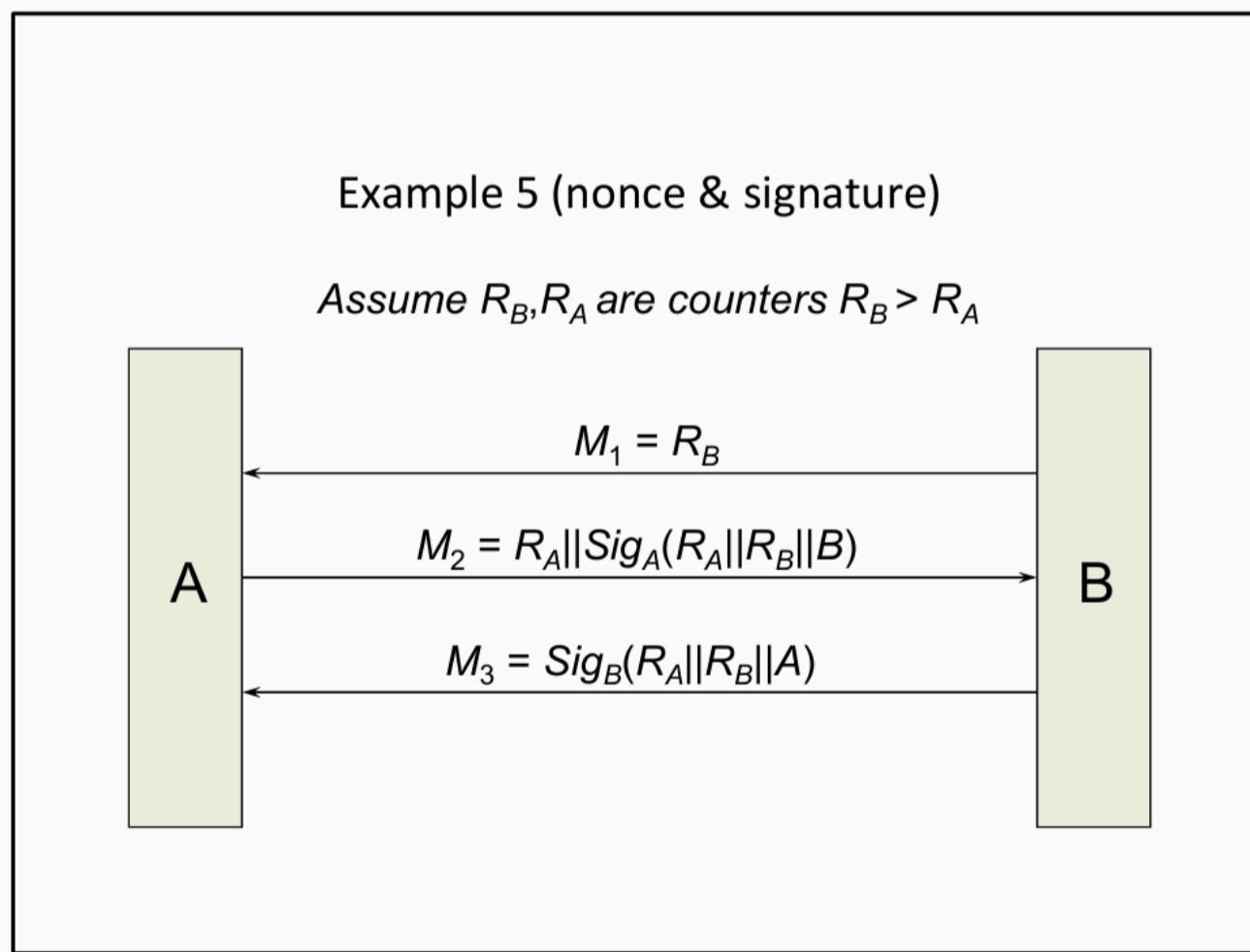
假如没有A或者B这样的标识符会怎么样？

并不影响A和B之间的对话，但攻击者可以立刻窃取A发给B的消息，然后发给B以外的其他人来假装是A



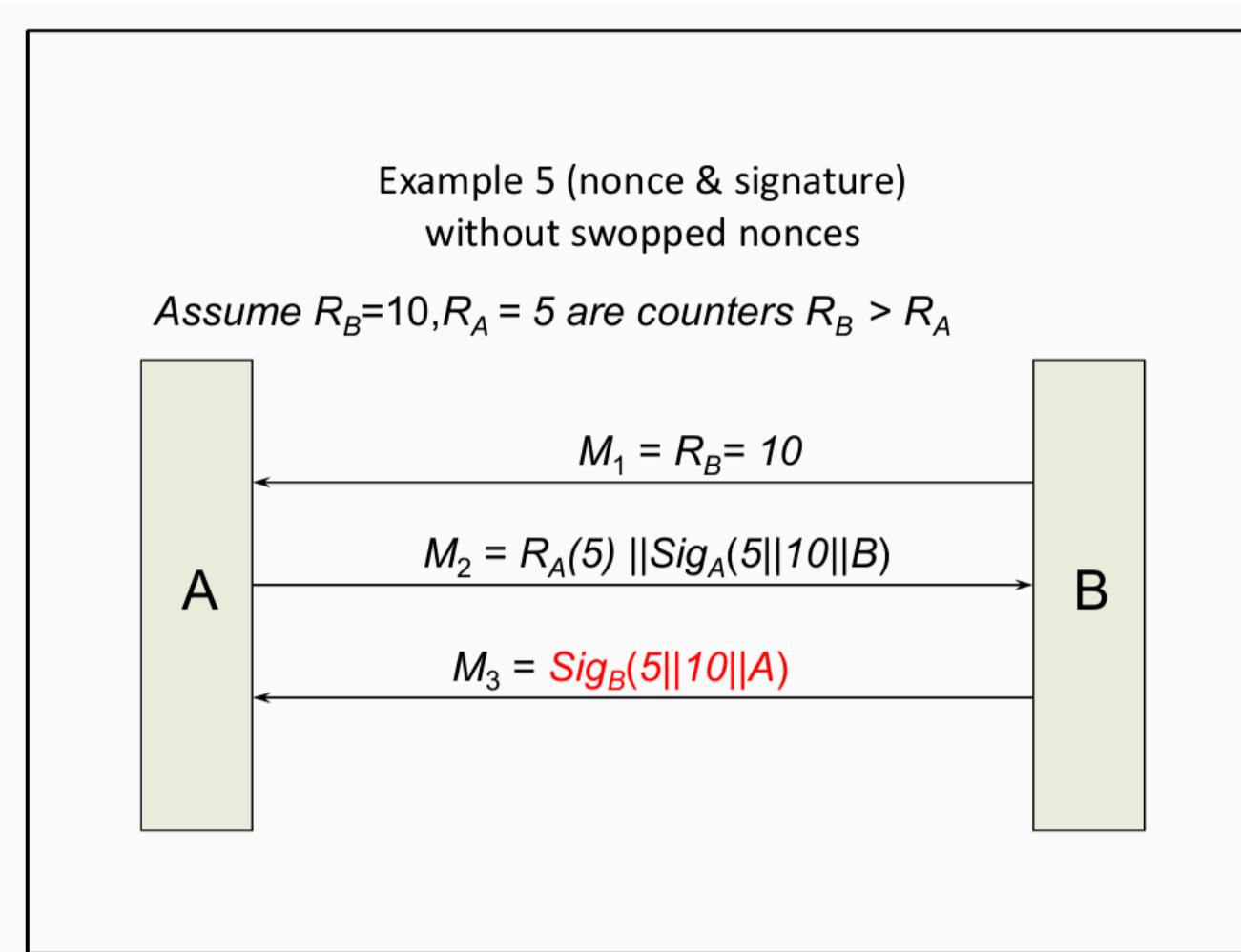
以上例子是使用随机数来提供实体验证的方法

上图中的 $Sig_A(R_A || R_B || B)$ 指示了消息的流动方向，假如其没有出于这个考虑，那么则会变成：

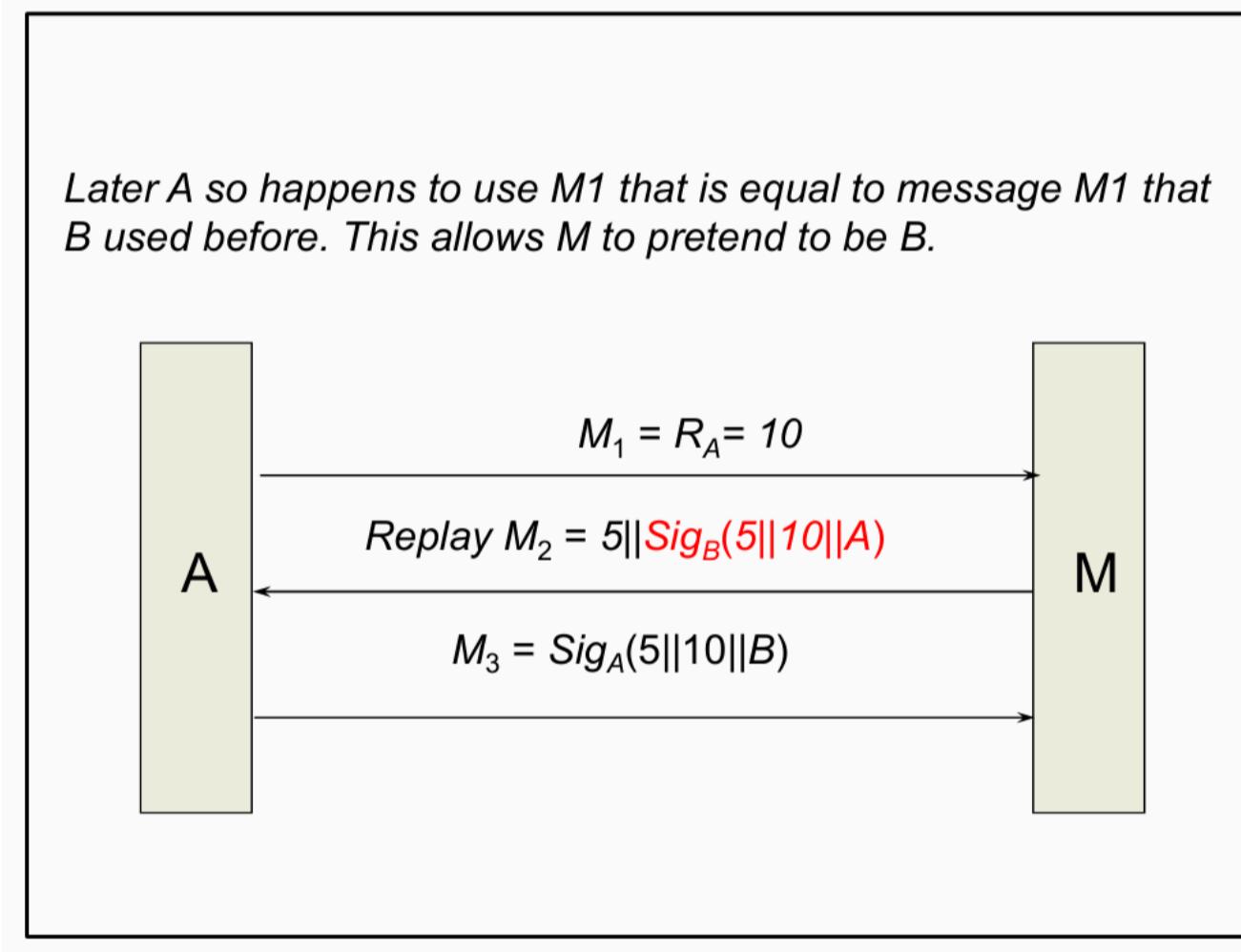


要注意的是，R仅仅代表A不会重复使用自己之前使用过的 $R_A$ ，不代表他之后不会使用B使用过的 $R_B$

假设A与B通过以上协议进行过验证



那么若之后攻击者尝试向A发起验证，A使用了使用过的 $R_B=10$ ，那么攻击者就可以通过截获 $M_3$ 并通过以下方法伪装成B



## 7. Key Management

Key management指的是生成密钥，使用密钥，存储密钥等一系列过程

### Symmetric-key protocols

2 main models:

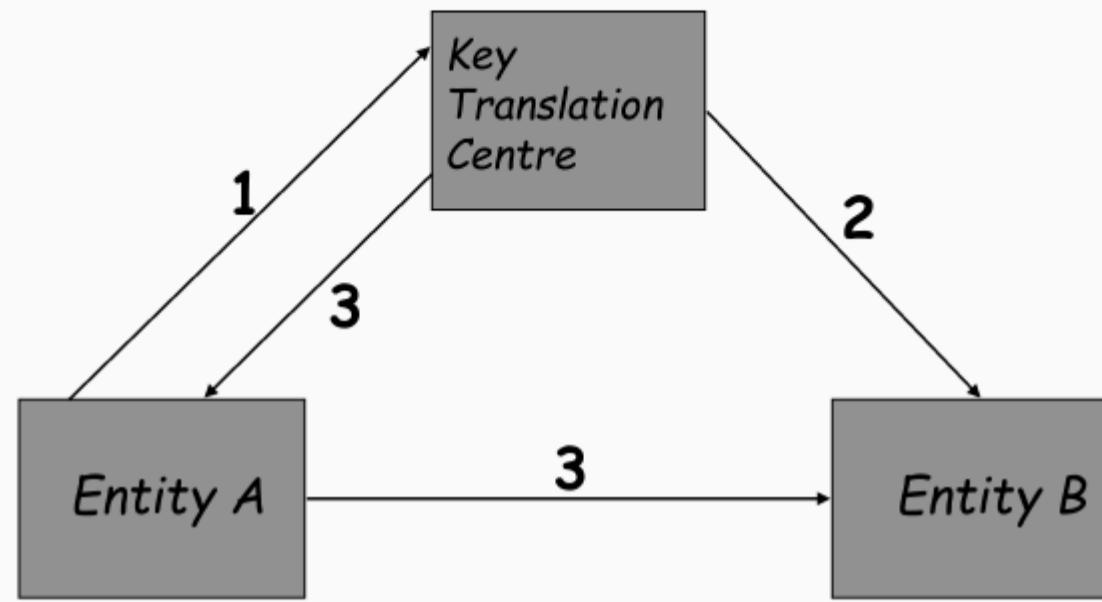
- Use of a Key Distribution Centre (KDC)
- Use of a Key Translation Centre (KTC)

根据密钥的情况可以分为使用非对称密钥或对称密钥，KDC和KTC用于对称密钥

Key Translation Centre (KTC):一个受信任的实体，用于在实体之间转运密钥

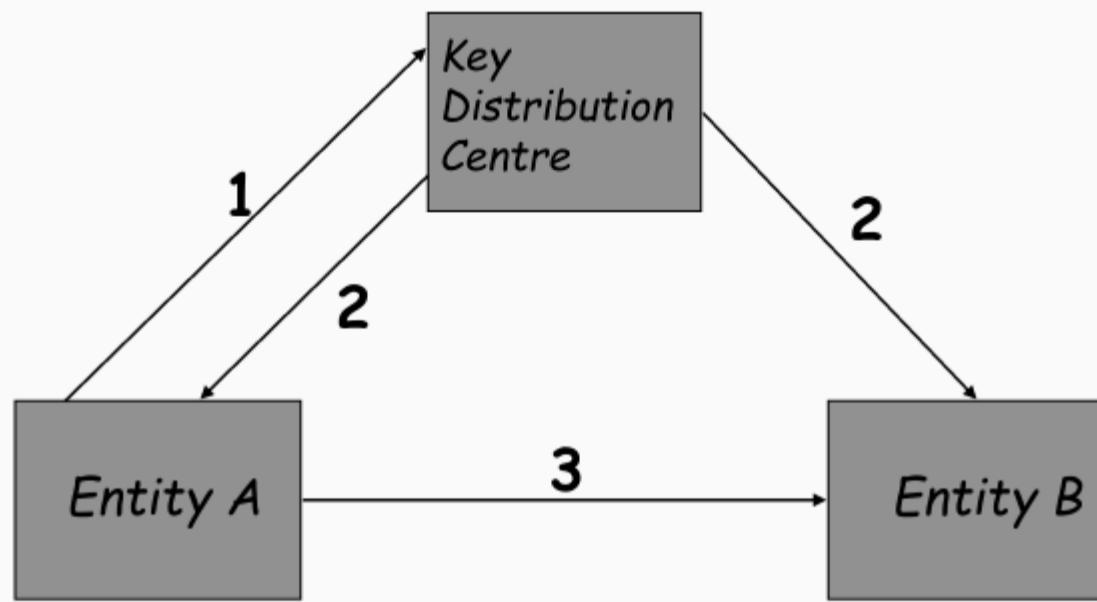
Key Distribution Centre (KDC):一个受信任的实体，用于生成密钥并在实体间分发

## Key Translation Centre



1. KTC接收A发来的密钥，使用它与A的共享密钥解密，然后使用它与B的共享密钥加密后发给B
2. B解密后，A与B使用密钥进行通信

## Key Distribution Centre



1. A请求KDC生成并分发A与B之间使用的密钥
2. 随后它将加密后的密钥发送给A和B，A和B各自解密后使用密钥进行通信（若AB共享有密钥，则只需要发送一个版本；如果它们分别与KDC共享，则需要各自发送一个版本）

Key establishment: 密钥建立。指让密钥供多方使用的通用术语

Key Agreement: 密钥协议。以两个实体都没有密钥控制权的方法建立密钥

Key transport: 密钥传输。将密钥从一个实体运输到另一个实体的过程

以上两者的区别是密钥控制权 Key Control：能够选择密钥值的能力

Key Confirmation：密钥确认。确保另一个实体能够获得特定密钥（通常为刚建立的密钥）

我们可以为Key Establishment设置两个需求：

- 隐式密钥认证 (Implicit Key Authentication)：向A保证只有另一个被识别的实体B能够拥有密钥。这是基本要求
- 显式密钥认证 (Explicit Key Authentication)：不但向A保证只有另一个被识别的实体B能够拥有密钥，还为其提供证据。这是更严格的要求

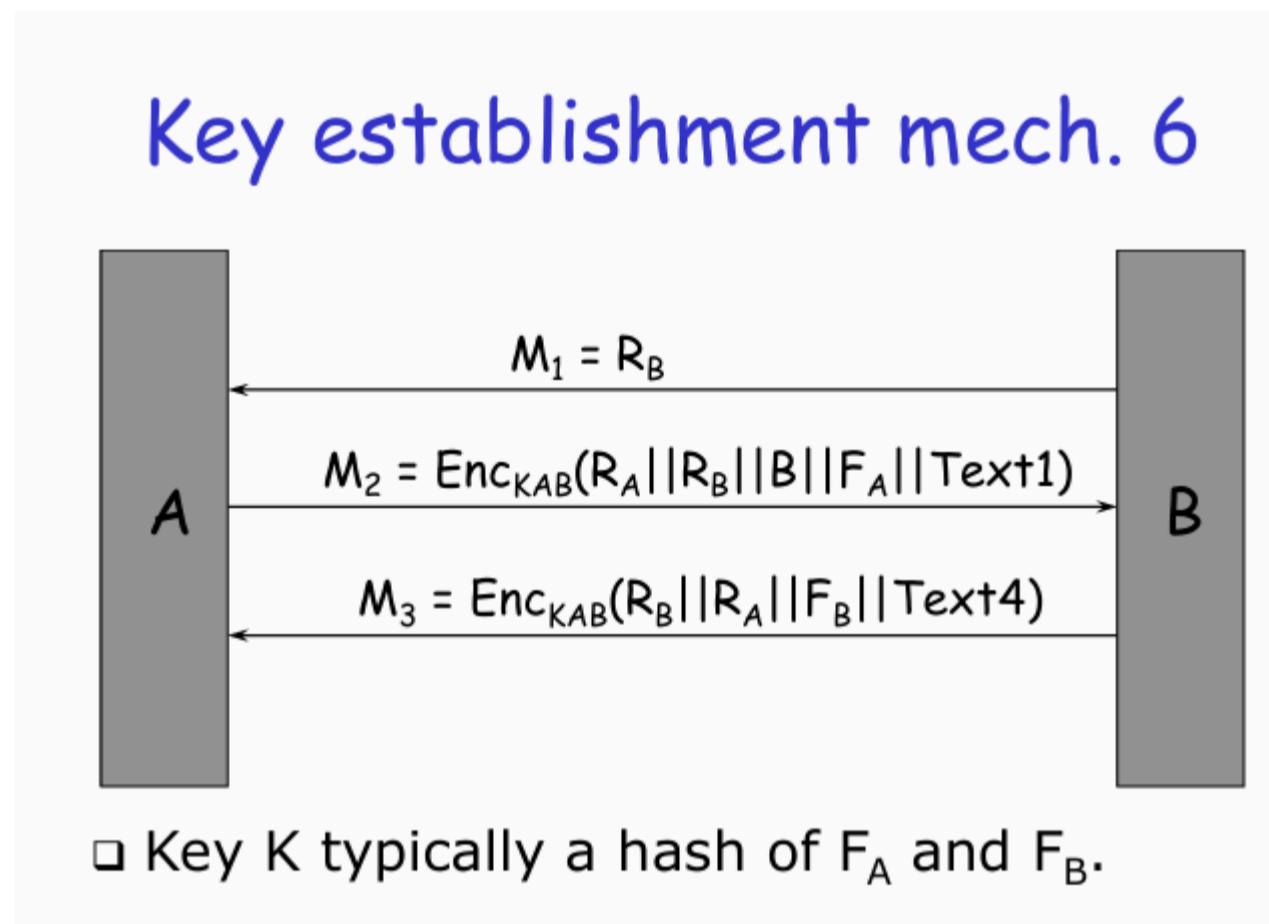
所以只需要确定B是否使用了新建立的密钥，即可得知是否显式，否则是隐式

分析以下机制（无需记忆）

A, B共享密钥K<sub>AB</sub>

$R_A R_B$ 都是nonce

$F_A F_B$ 都包含密钥材料



以上过程中：

生成新密钥K

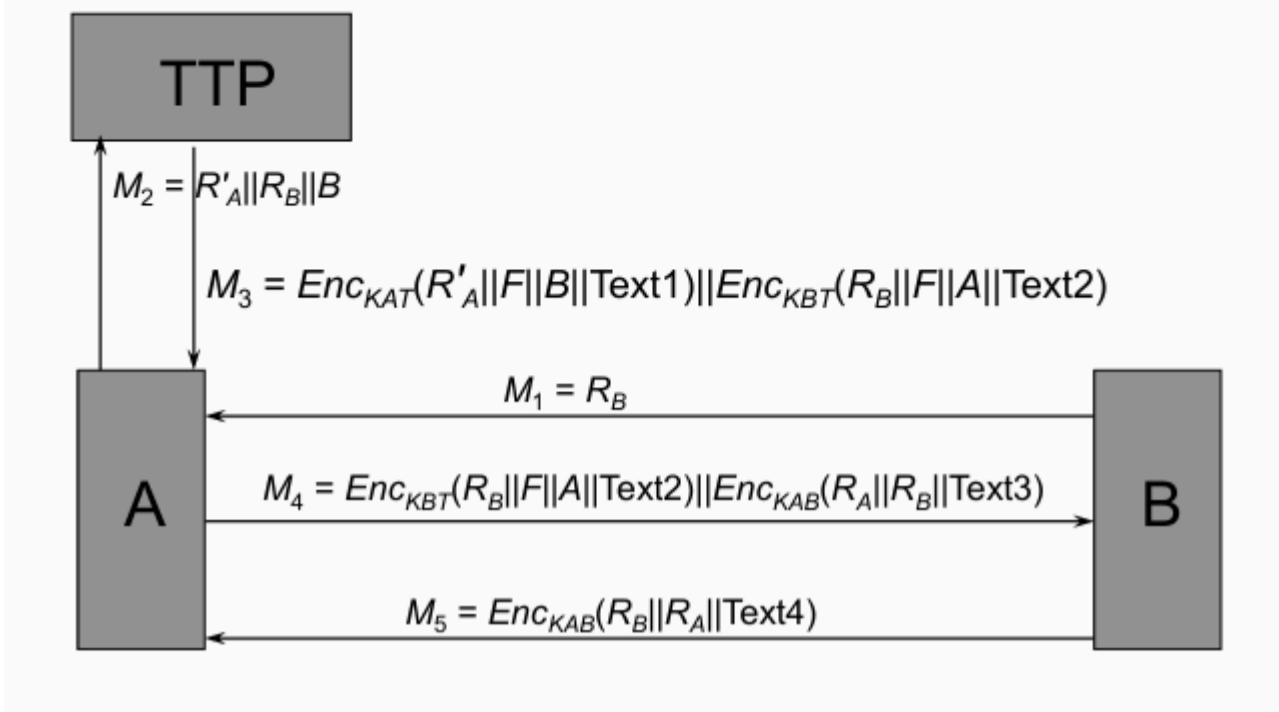
A与B完成相互身份认证（R保证freshness，使用K<sub>AB</sub>使得双方完成数据来源验证）

双方都没有密钥控制，所以是Key agreement

假如 $F_A$ 或 $F_B$ 不包含数据，那么包含数据那一方就有控制权

显式密钥认证和Key Confirmation都没有，因为没有使用到生成的密钥K

## Key establishment mech. 9



该过程用于A和B建立共享密钥KAB

使用Nonce提供freshness

TTP发送F给A后，A根据F生成KAB

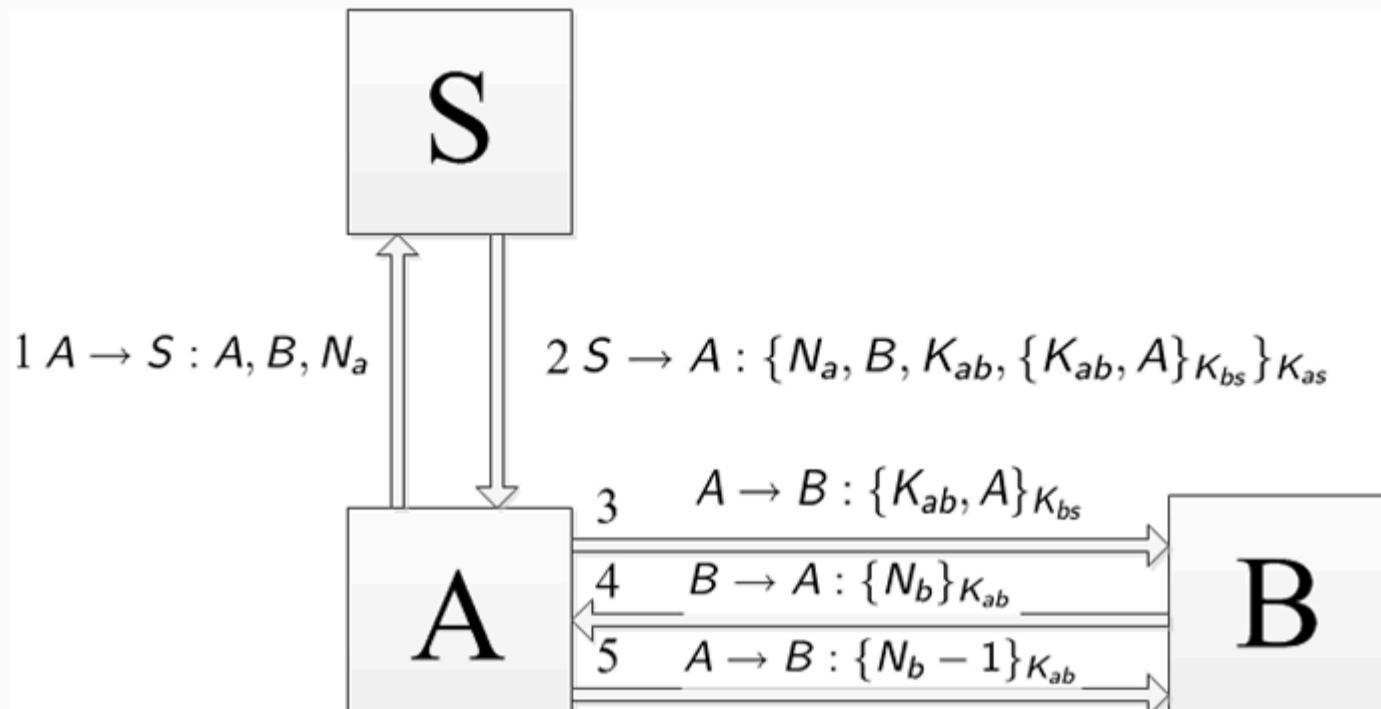
发送给B后，B也通过同个过程生成KAB

$M_5$ 中 $R_B, R_A$ 与M4中相比调换了顺序，避免了反射攻击

并且双方分别使用了KAB做了加密，既保证了身份验证，又确保了显式密钥认证

但只有TTP具有密钥控制

## Remember from last lecture....



A认证了S的身份但S没有认证A的身份

B认证了A的身份但A没有认证B的身份

显式密钥认证

S具有密钥控制

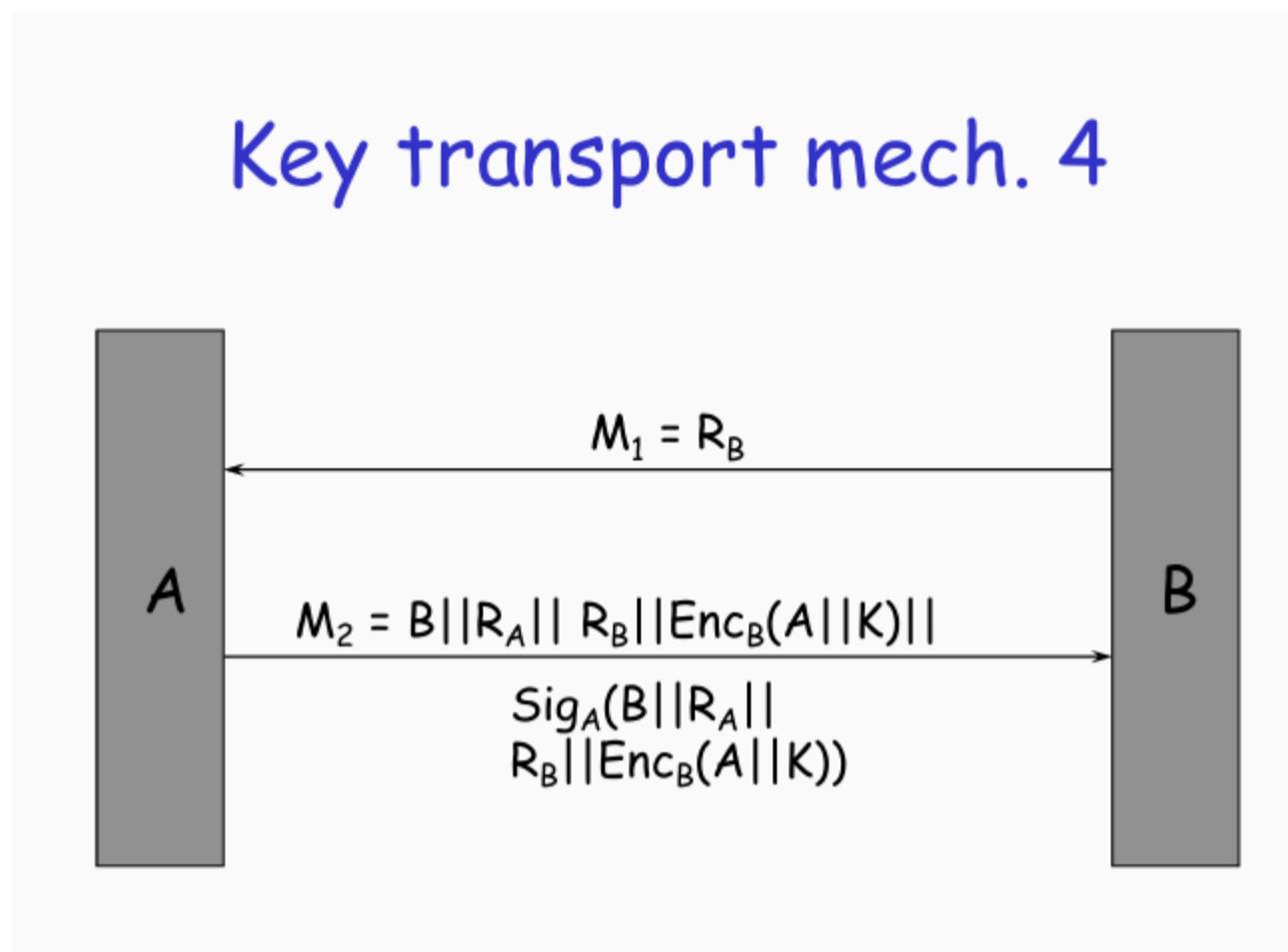
## Public-key protocols

公钥协议可分成

- 密钥传输协议 Key Transport protocols: 通常设计公钥加密和数字签名
- 密钥协议 Key Agreement protocols: 间接指定, 主要基于Diffi-Hellman

注意, 由于这里考虑的是公钥加密, 因此  $Enc_A(X)$  可以由任何实体进行

另外, 公钥加密中MDC没有作用, 因为任何人都可以加密, 无法提供身份验证



B验证了A的身份

A决定密钥值, Key Transportation

隐式密钥认证

## Key hierarchies

密钥通常按层次来组织, 上层密钥用于保护或生成下层密钥

最顶层密钥是主密钥, 必须好好保护

最底层密钥是会话密钥, 仅用于为单次对话提供数据安全

密钥被使用的越多, 就越容易泄露, 因此生命周期越短

会话密钥容易被攻击但由于常常变化, 问题不大

主密钥使用频率低, 生命周期长

## Public Key (Certificate) Management

在公钥加密体系中，验证公钥很重要

假如M声称自己是A并用自己公钥发送给B，那么B就无法辨别到底哪个公钥才是真正的A的公钥

以上问题对于数字签名同样适用，必须确保用来解签的公钥确实是正确实体的公钥才行

解决方案是引入数字证书 Digital Certificate

系统中有一个可信实体CA，其公钥被所有人知道

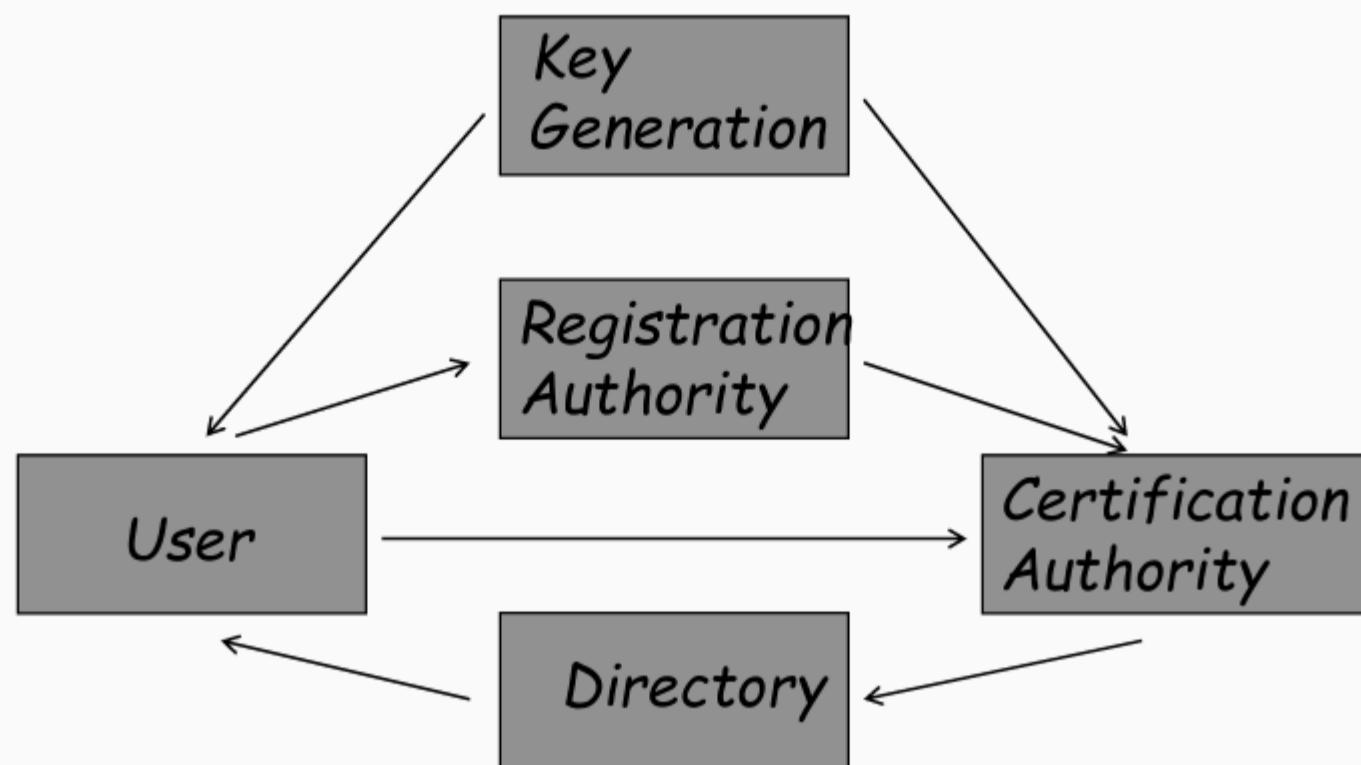
对每个公钥持有者，CA通过颁发证书包含以下条目来认证其公钥确实有效：

- 公钥持有者身份
- 公钥
- 证书有效期
- CA签名

在颁发证书前，CA要负责：

- 识别实体身份
- 生成或验证用户的密钥
- 确保用户密钥对的质量
- 保护用户的私钥
- 验证用户是否知道其声明的公钥对应的密钥

## Who is involved?



用户决定生成密钥和证书，则：

1. 寻找密钥生成机构KGF来生成密钥对
2. KGF把密钥对发送给用户，公钥发给CA
3. 用户向RA证明其身份，RA将证明发给CA
4. 用户直接和CA交流，表示需要证书
5. CA为其生成证书

注意，给用户呈现证书并不能证明呈现者的身份，证书只能证明其上的ID和公钥

CRL (Certification Revocation Lists)：证书吊销列表，其内包含了将要吊销的证书序号

为什么需要吊销证书？

可能密钥被泄露，可能拥有者主动更改密钥

## PKI

Public Key Infrastructure 公钥基础设施，包含安全使用公钥加密所需的所有成分：

- 密钥生成和密钥管理
- CA, 数字证书
- CRLs

PKI没有标准模型，这里只考虑PKI信任模型 PKI Trust Model

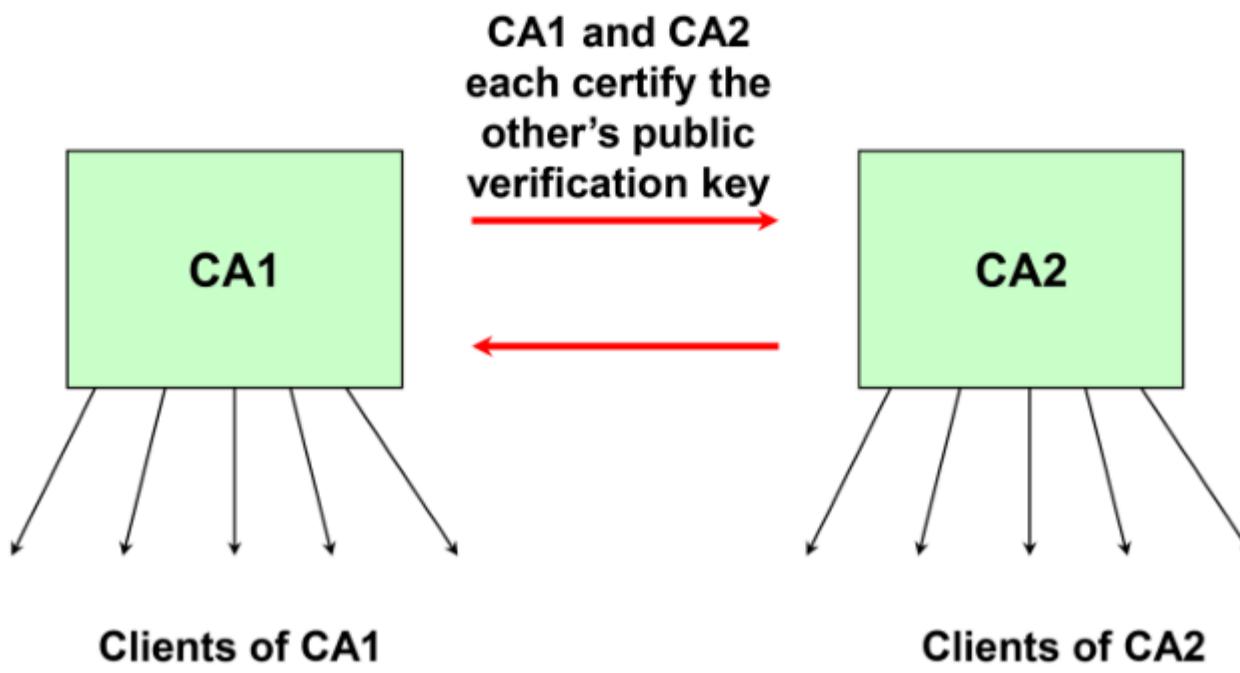
该模型：

- 多个受信任CA
- 正在被使用
- 浏览器可能内置数十个根CA的公钥
- 用户决定信任哪些CA（默认信任浏览器信任的CA）

我们通过使用签署该证书的CA的证书来一路验证，直到到达绝对可信的根CA

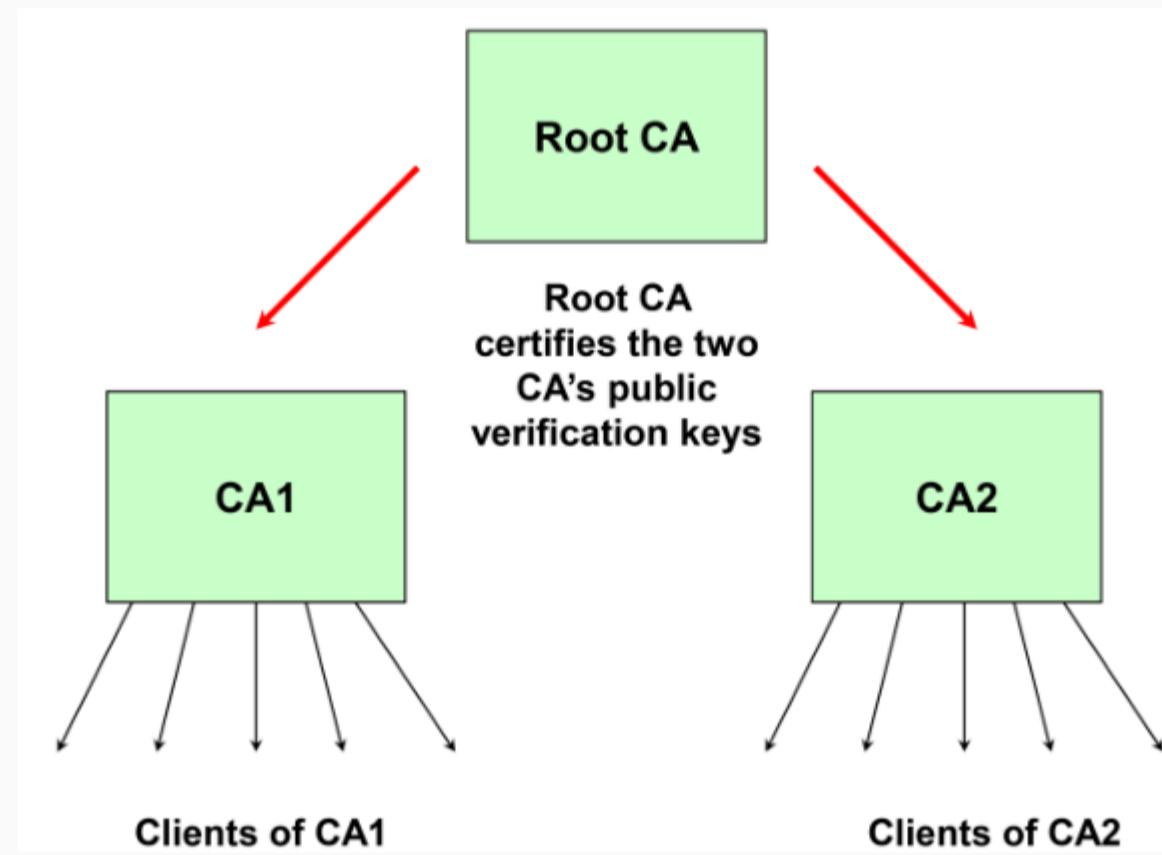
那么证书之间是如何认证的呢？通常通过交叉认证Cross Certification或自认证 Self Certification

## Cross certification



双方分别验证各自的公共验证密钥

## Certificate Hierarchy

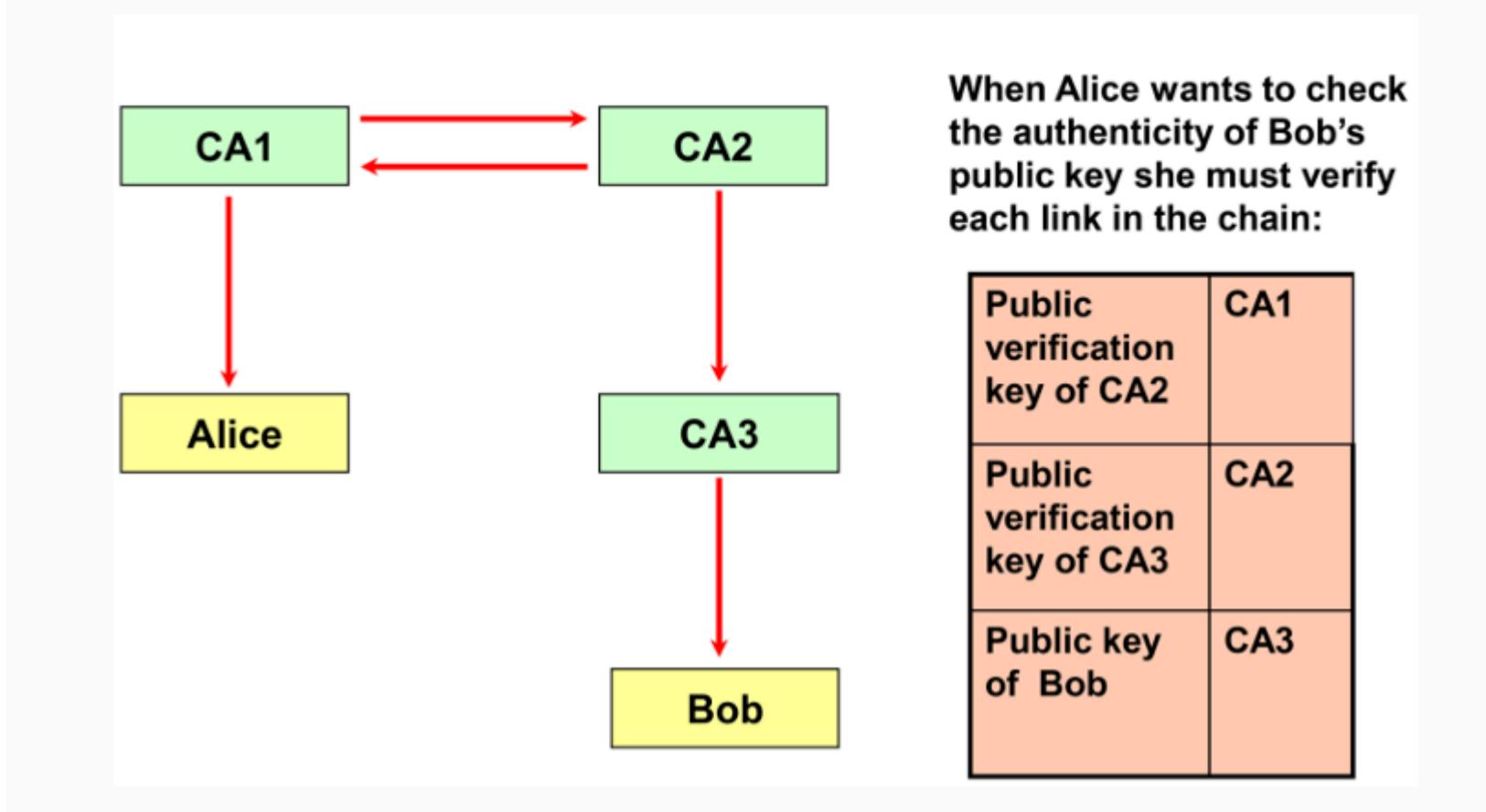


45

或者通过高等级的根CA来对低级的CA提供验证

最高级的根密钥证书可能会自验证，我们通过其证书上验证的公钥来对其签名验证（我们相信根CA是诚实的）

# Certificate Chains



验证链：

A想要验证B的公钥，B的公钥由CA3的证书保证

A使用CA3的公钥验证B拿到的证书

A使用CA2的公钥验证CA3的证书

A使用CA1的公钥验证CA2的证书

A信任CA1

## 8. Compsec

Access control分成两种：

- Authentication: Who are you
- Authorization: What are you allowed to

如何做认证？

3个因素：

- Something you know
- Something you have
- Something you are

在实际应用中，常常使用密码来完成认证

但用户创建的密码往往不是完全随机的

## Password File

我们需要在设备中保存用户的密码文件，否则无法验证用户身份

但直接保存明文密码文件很愚蠢

所以我们考虑对密码使用hash

存储 $y = h(password)$ 在密码文件内

这样在输入密码时就可以做hash，随后验证是否相等、

但是如果攻击者拿到了密码文件，那么他就可以猜想密码 $x$ ，随后验证是否有 $y = h(x)$

如果有，那么攻击者成功拿到密码

以上思路对应字典攻击

攻击者预先计算常见密码字典中所有 $x$ 对应的哈希值

随后假如攻击者拿到了密码文件，那么他就可以一一比对其内的哈希值和他预先计算的值是否相同

这样的攻击方法每次攻击耗时都相同，不受密码长度影响

如何应对字典攻击：salt

对密码password，选定随机的s，计算

$y = h(password, s)$

随后存储 $(y, s)$ 在密码文件中

注：salt值不需要保密

salt值的存在使得攻击者需要为每个可能的salt值，针对每个可能密码都计算一次hash，极大增加了计算量

例如：可能的密码有1000，可能的salt有100，那么在引入salt前，只需要计算1000次hash，引入后需要计算 $1000 * 100$ 次

## Phishing attack

攻击者伪装masquerades成值得信赖的实体

诱导用户进行一些暴露自身敏感信息的操作

钓鱼网页和邮件等会伪装得看起来合法

难以预防，只能通过教育用户来实现

难以定位到钓鱼网站的位置，因为网站可能经过了许多代理

## 2-factor Authentication

双重认证：使用以下3点中的2点来进行认证：

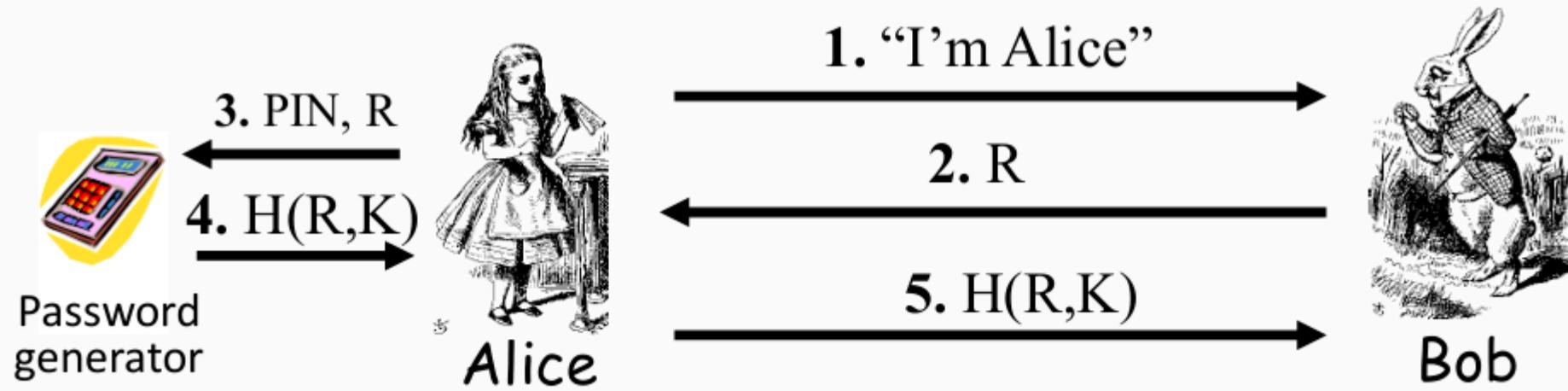
1. Something you know
2. Something you have
3. Something you are

使用双重认证，即使其中一者失败了还有另一者兜底

最常见的2FA:

密码+用户持有的某样东西

## Password Generator

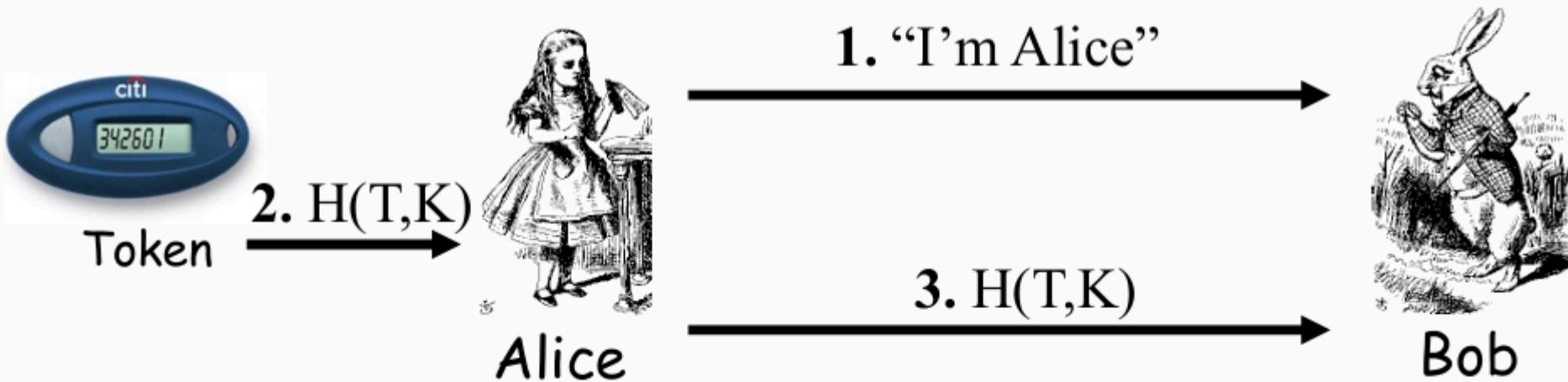


- Alice gets “challenge” R from Bob
- Alice enters R into password generator
- Alice sends “response” back to Bob
- Alice **has** pwd generator and **knows** the PIN
- K is only known to Bob and Password Generator, but not to Alice!

该协议中对A进行了身份验证，因为用到了只有她才持有的PIN

## Dynamic Password Token

(a.k.a. Time-Based One-Time Password or T-OTP)



- Timestamp T is the “challenge” (*yes, the Token has an internal clock*)
- K is only known to Bob and the Token, but not to Alice, or we say “*not necessary for Alice to know*”
- Alice sends “response” H(T, K) back to Bob
- Alice **has** the Token
- **Time synchronization** between the token and Bob is required.

该协议没有对Alice进行身份验证，而是默认只有Alice才持有token

## Firewalls

防火墙用于规定什么内容可以进入内部网络，提供访问控制access control的作用

防火墙可以分为以下类型：

数据包过滤器 Packet filter:

- 运行在网络层
- 根据源IP，端口或目标IP，端口以及标识符flags进行过滤（作为网络层的过滤器，其可以查看标识符但不知道具体含义）
- 优点：速度快
- 缺点：无状态，无法查看TCP连接，无法过滤应用层数据

状态数据包过滤器 Stateful Packet Filter:

- 运行在传输层
- 在数据包过滤器的基础上增加了状态信息
- 可以记住TCP连接，flags甚至是UDP信息
- 优点：可以做任何数据包过滤器可以做到的事，可以跟踪现存的连接
- 缺点：无法查看应用层数据，比数据包过滤器慢

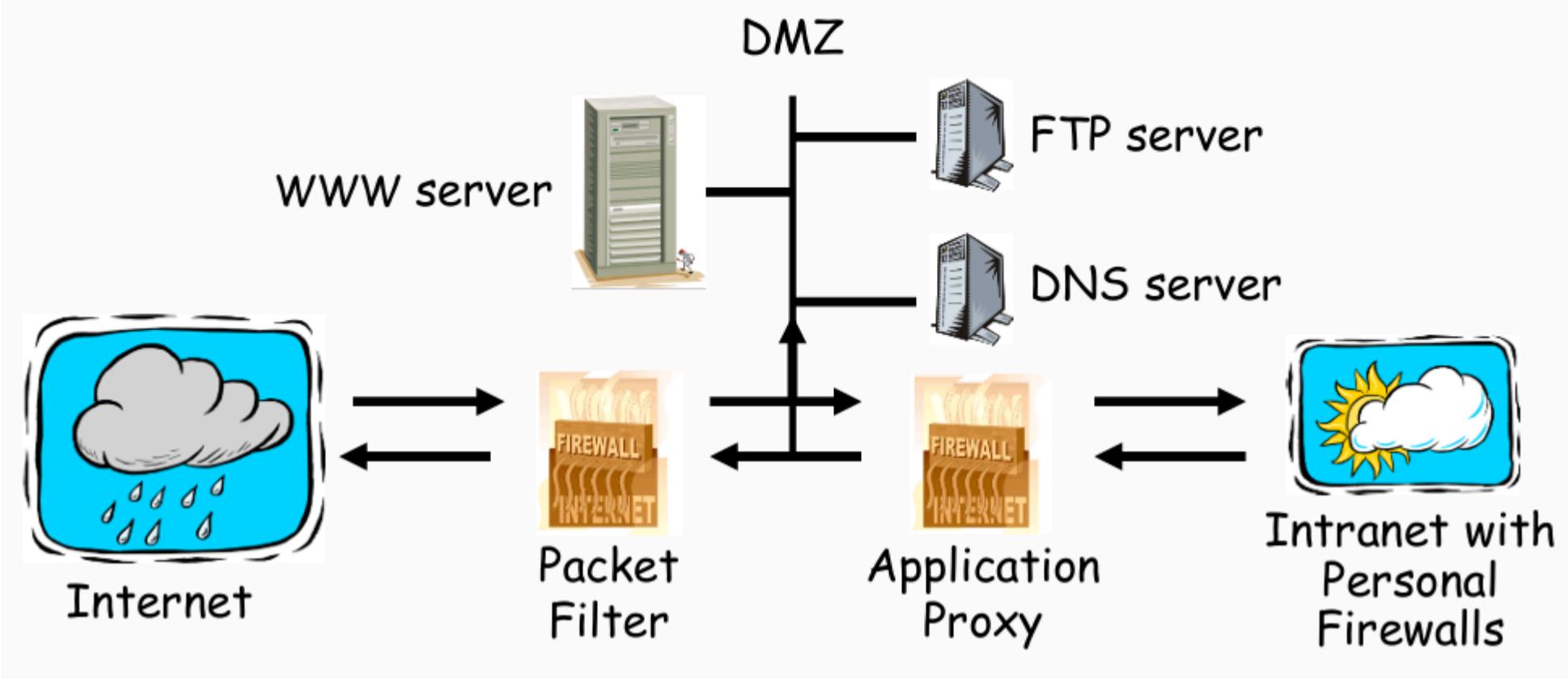
应用代理 Application Proxy

- 运行在应用层
- 应用代理代表用户本身提前查看传入的应用层数据，验证传入之前的数据是否安全
- 优点：可以查看连接和应用数据，在应用层面过滤不良数据
- 缺点：速度慢

个人防火墙 Personal Firewall

- 用于防护单用户或家庭网络
- 可以基于所有以上三种防火墙构建

## □ Example security architecture



对于极大量的数据流入情况例如流向服务器，最好不要采用application proxy

但在流向个人网络之前，仍然应该通过第二道防火墙（application proxy）

## Malicious Programs

### 陷门/后门 Trapdoor/Backdoor

开发人员留下的一个入口，为其将来修改或访问未经授权的数据提供了可能

对策：自行开发程序

### 逻辑炸弹 Logic Bombs

嵌入在合法程序中的入口，触发时会引爆explode，破坏或删除文件

### 特洛伊木马 Trojan Horses

一段隐藏的代码，使得程序在运行正常功能之外还会运行隐藏功能例如收集用户的密码

### 病毒 Virus

通过修改其他程序来实现感染infect，可引起巨大增长

### 细菌 Bacteria

不销毁任何程序或文件的程序，只会不断自我复制，消耗资源（availability attack）

### 蠕虫 Worms

可视为病毒的扩展extension，通过网络感染网络中的其余设备

## 9. Network

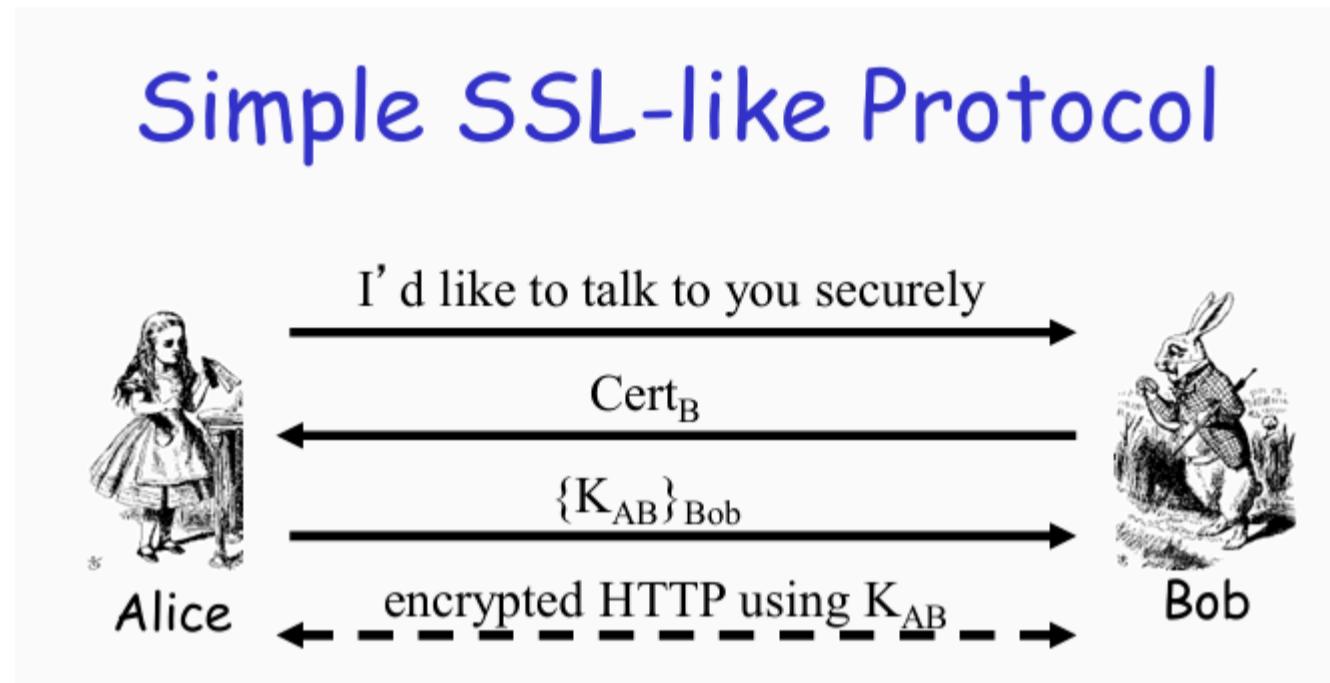
### Secure Socket Layer (SSL)

套接字socket位于应用层和传输层之间，因此SSL作用于HTTP和TCP

SSL是用于Internet中大多数安全交易的协议

其主要提供的安全服务包括：

- 单向身份认证（用户确认对方的确是电商网站，而电商网站不在意用户是谁）
- 数据机密性（信用卡数据信息）
- 但SSL也能提供双向认证，对于服务器之间的情况



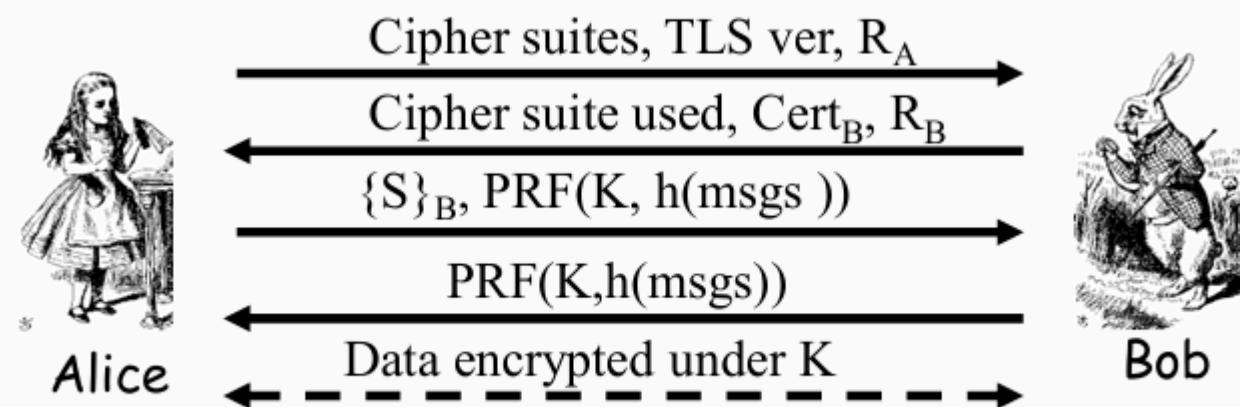
SSL大致通过以上模式工作：

- B发送证书，包含其公钥
- A发送用B的公钥加密的共享密钥给B
- B用其私钥解密得到共享密钥，两者用该密钥开启会话

以上过程中，A认证了B的身份（身份验证）

对HTTP数据进行了加密（数据机密性）

## Simplified SSL Handshake Protocol



- S is randomly chosen by Alice
- K = h(S, R<sub>A</sub>, R<sub>B</sub>)
- msgs = all previous messages

以上是简化的SSL握手过程：

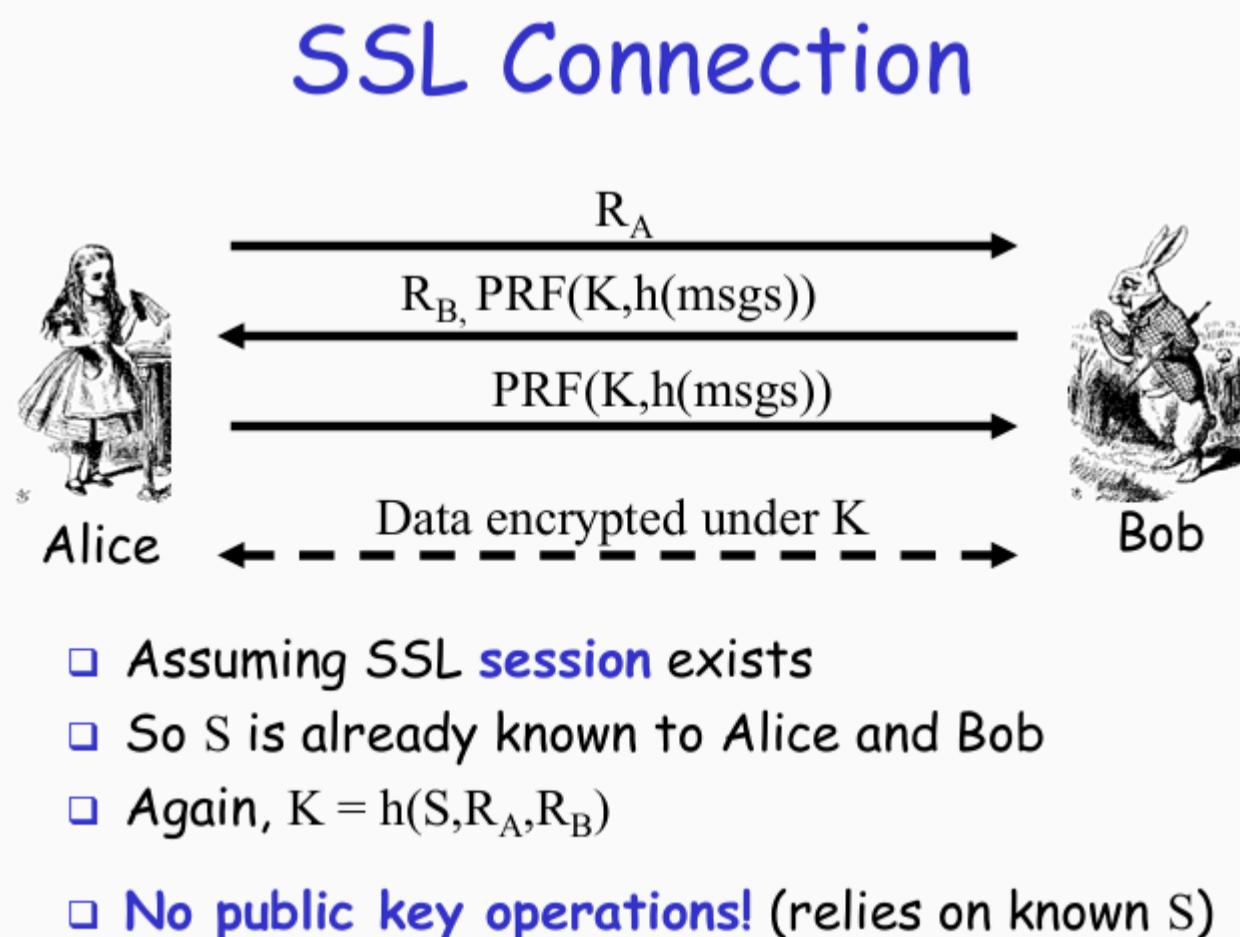
1. A发送支持的加密套件, TLS版本和随机数RA
2. B回复选择的加密套件, B的证书和随机数RB
3. A随机选则S并加密, 用生成的 $K = h(S, R_A, R_B)$ 使用PRF对之前的所有消息做hash后生成随机序列
4. B解密得到S, 然后做类似过程发给A
5. 随后双方可以使用密钥K进行对话

这一过程中A对B的身份进行了验证

双方都具有Key Control

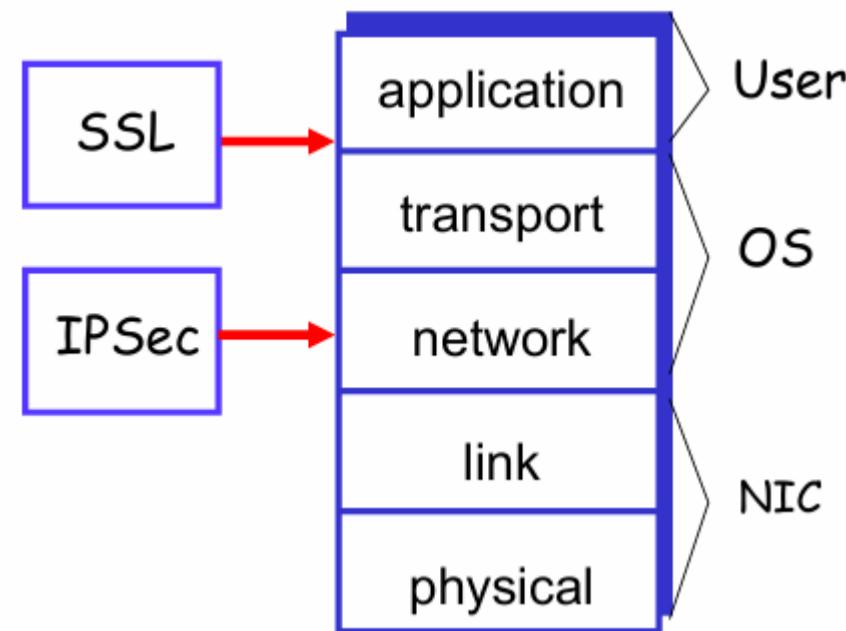
双方都有Key Confirmation

由于SSL建立连接很昂贵, 双方可以在存在SSL会话的情况下基于会话信息建立新的SSL连接:



以上过程可以基于已有的S生成新的会话密钥K, 这一过程不涉及到对公钥的运算

# IPsec



IPsec存在于网络层，并且对应用层透明可见

IPsec存在两个部分：

- IKE (Internet Key Exchange) : 建立会话密钥
- ESP (Encapsulating Security Payload) /AH (Authentication Header) : 安全通道如何工作

在SSL中我们只讨论了关于建立共享密钥的部分

## IKE phase1

用来建立会话密钥

包含3种方式：

- 基于公钥加密 Public key encryption based
- 基于签名 Signature based
- 基于对称密钥 Symmetric key based

其中每种方式包含2种模式

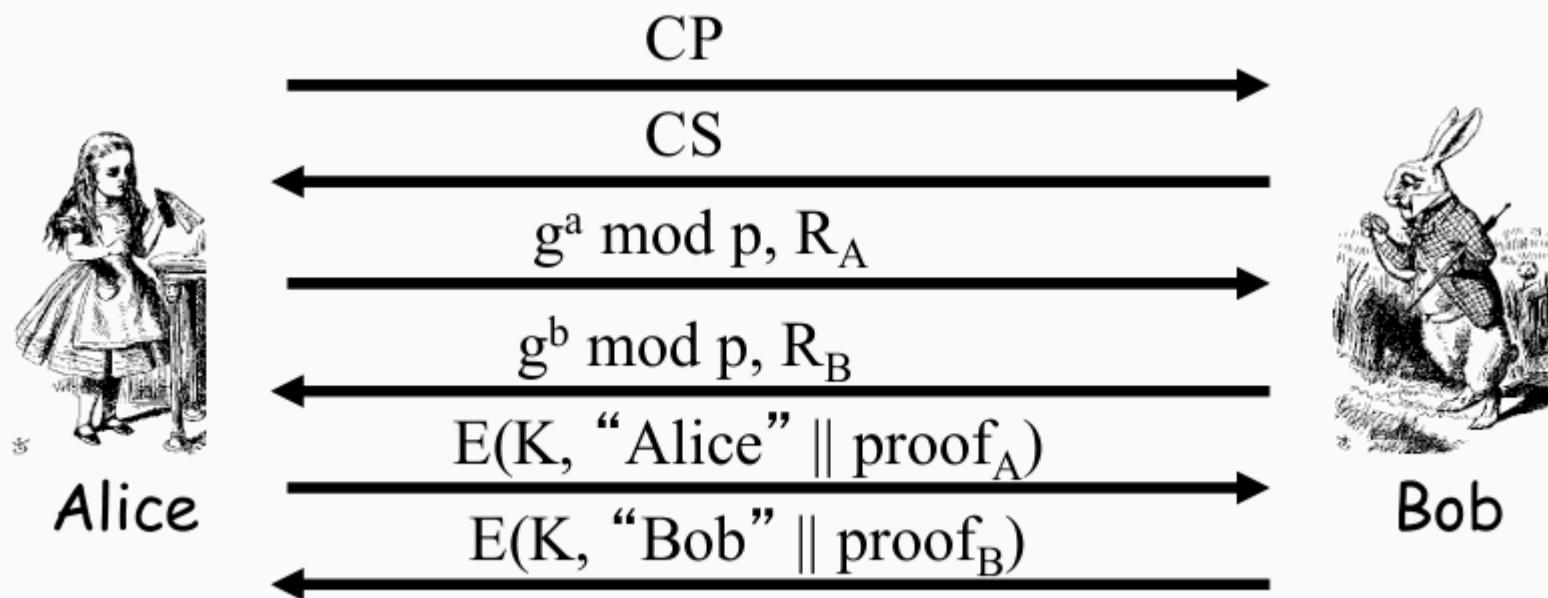
- Main Mode
- Aggressive Mode

所有总共有6种变量

不需要记忆所有的细节，但见到协议要能看出其背后的原理，以及知道该方式下的Main Mode和Aggressive Mode的区别

首先要集中，每种变量都基于Diffie-Hellman密钥交换

# IKE Phase 1: Signature Based (Main Mode)



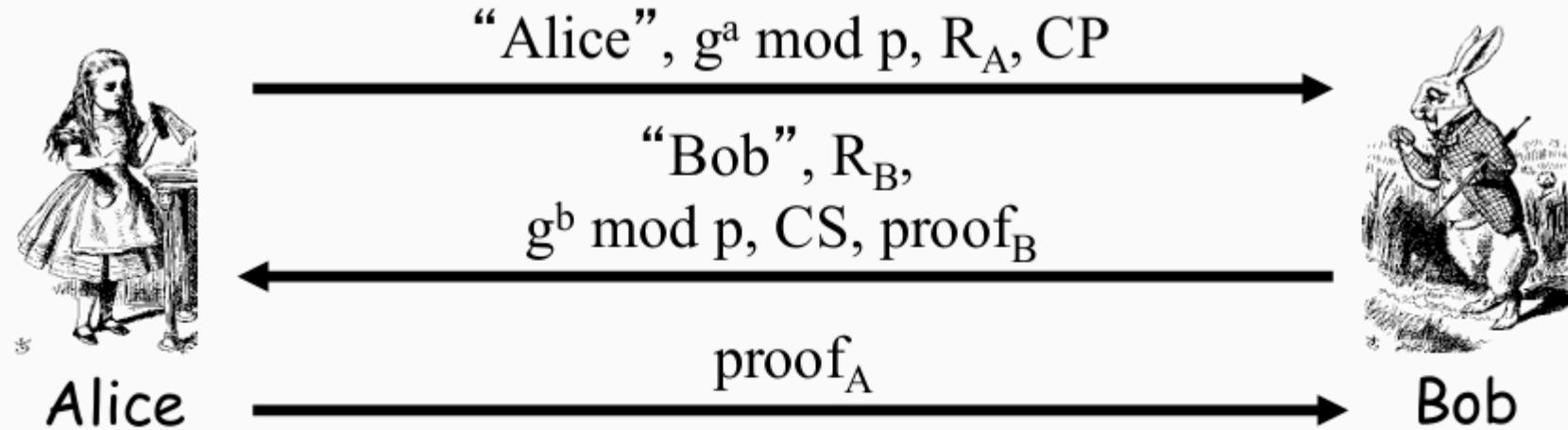
- CP = crypto proposed, CS = crypto selected
- $K = h(g^{ab} \text{ mod } p, R_A, R_B)$
- SKEYID =  $h(R_A, R_B, g^{ab} \text{ mod } p)$
- $\text{proof}_A = [h(\text{SKEYID}, g^a \text{ mod } p, g^b \text{ mod } p, \text{CP}, "Alice")]_{\text{Alice}}$

该方法中签名体现在proof的生成中，运用到了签名

两者的ID都被加密后传输，都是私密的private

涉及到显式密钥认证，都是用生成的密钥K进行了加密

# IKE Phase 1: Signature Based (Aggressive Mode)



- Main difference from main mode
  - Not trying to protect identities
  - Cannot negotiate g or p

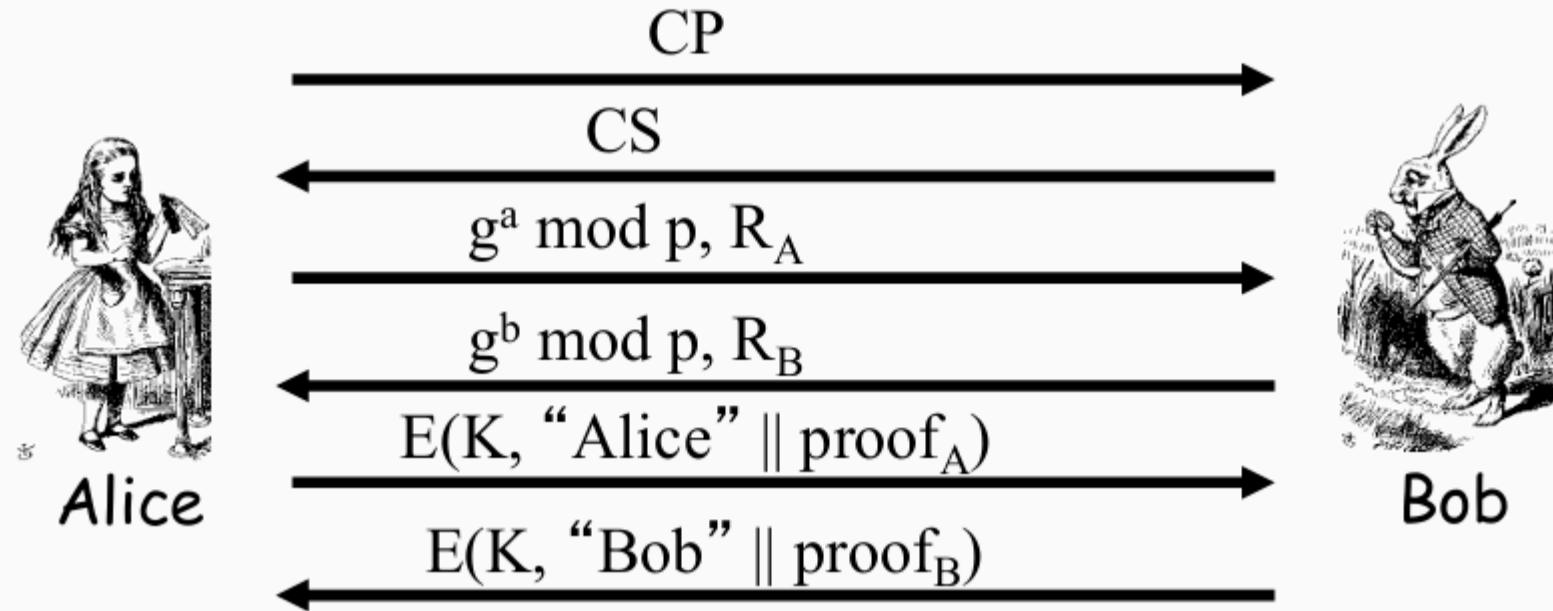
没有显式密钥认证

没有提前协商g或p，直接由A选择

(以上也是所有aggressive和main的不同之处)

与Main Mode相比，个人ID不被保护

## IKE Phase 1: Symmetric Key Based (Main Mode)



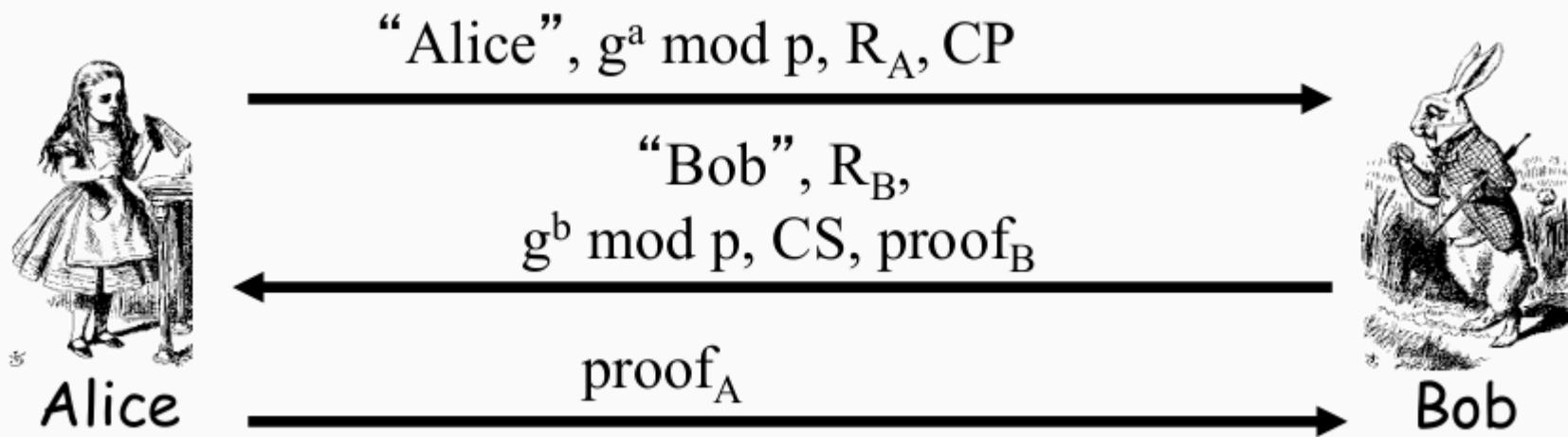
- $K_{AB} = \text{symmetric key shared in advance}$
- $K = h(g^{ab} \text{ mod } p, R_A, R_B, K_{AB})$
- $\text{SKEYID} = h(K, g^{ab} \text{ mod } p)$
- $\text{proof}_A = h(\text{SKEYID}, g^a \text{ mod } p, g^b \text{ mod } p, \text{CP}, "Alice")$

此方法基于两者提前共享的对称密钥KAB

但存在一个问题：B必须确定他是在和A对话

否则他无法得知应使用哪个KAB来生成K，也就无法获得“Alice”来确定A的身份。因此“Alice”必须是IP地址

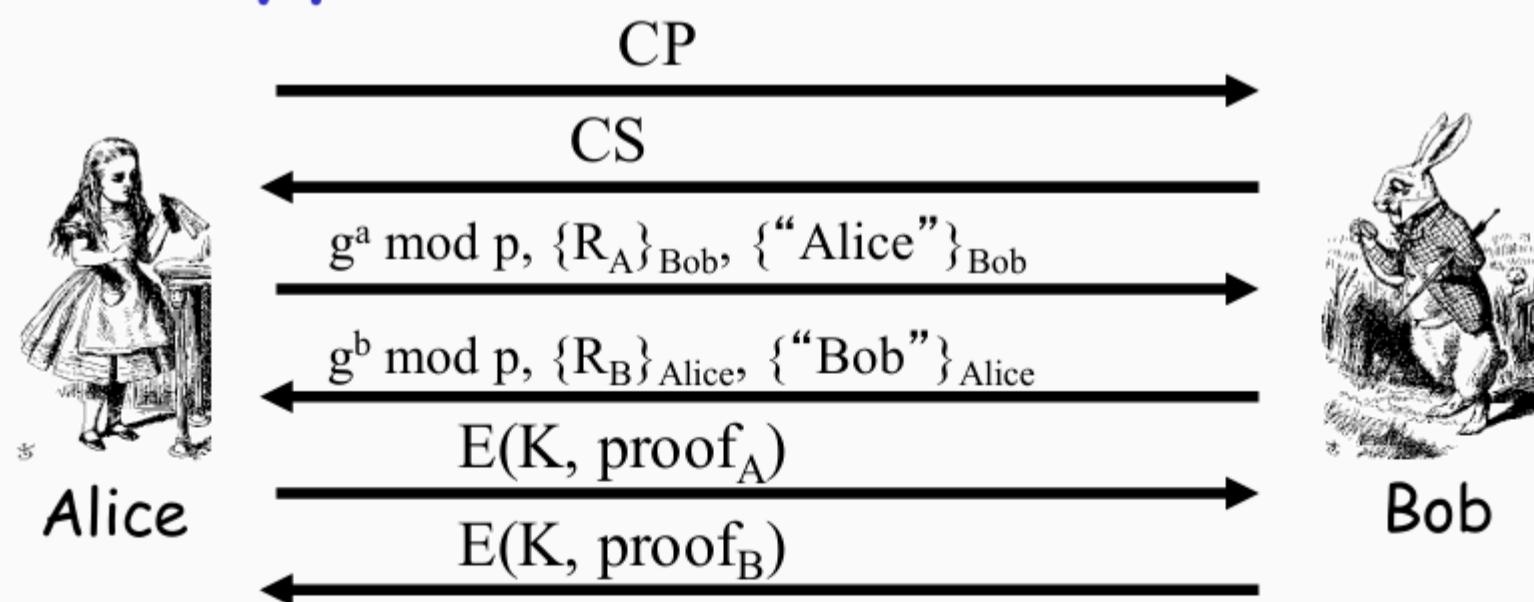
# IKE Phase 1: Symmetric Key Based (Aggressive Mode)



- Same format as digital signature aggressive mode
- Not trying to hide identities...
- As a result, does **not** have problems of main mode

aggressive模式中ID都是显式发送因此不存在Main Mode的问题

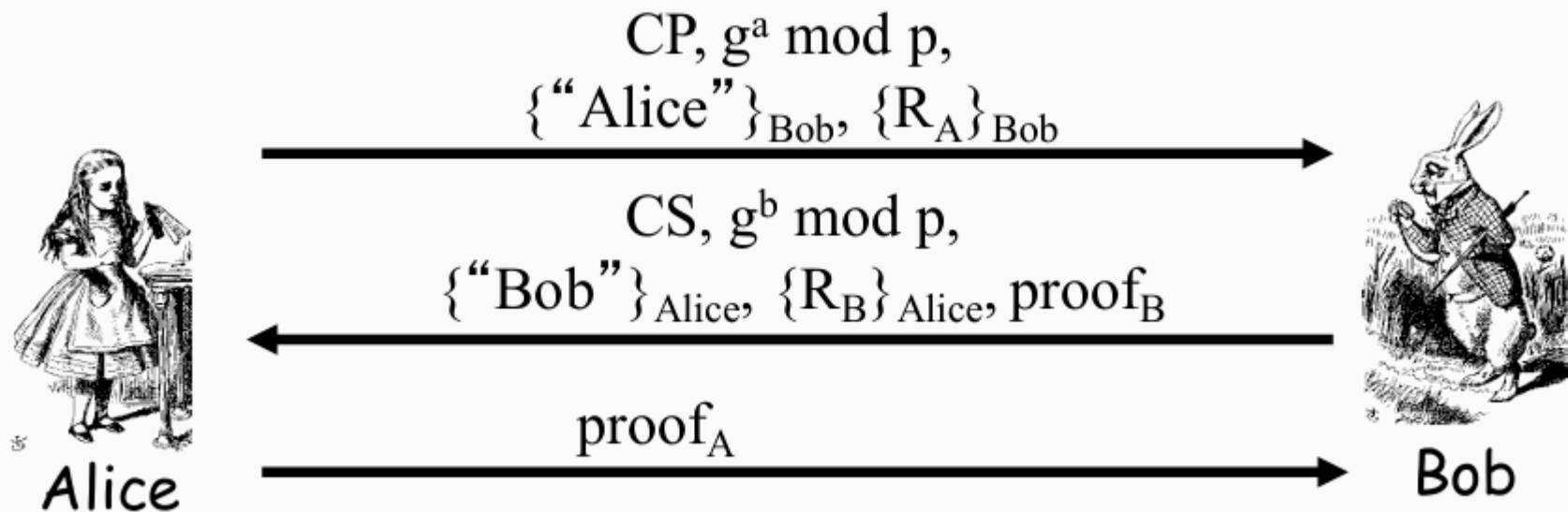
# IKE Phase 1: Public Key Encryption Based (Main Mode)



- $K = h(g^{ab} \text{ mod } p, R_A, R_B)$
- SKEYID =  $h(R_A, R_B, g^{ab} \text{ mod } p)$
- $\text{proof}_A = h(\text{SKEYID}, g^a \text{ mod } p, g^b \text{ mod } p, \text{CP}, \{"Alice"\})$

该模式使用双方公钥对ID进行了加密

# IKE Phase 1: Public Key Encryption Based (Aggressive Mode)



- $K, \text{proof}_A, \text{proof}_B$  computed as in main mode
- Note that identities are hidden
  - The only aggressive mode to hide identities
  - Then why have main mode?

该方法是唯一一个隐藏ID的aggressive mode

## ESP and AH

该部分涉及会话通道安全性的保障

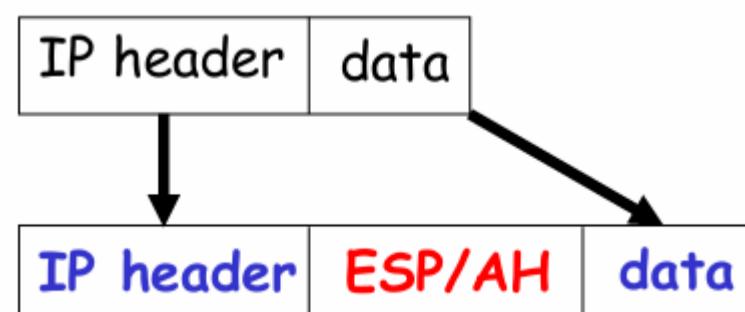
两种封装模式：

- 传输模式 Transport mode
- 通道模式 Tunnel mode

两种协议：

- AH：只支持消息验证（完整性和来源）
- ESP：支持加密以及基于加密的消息验证（完整性和来源）

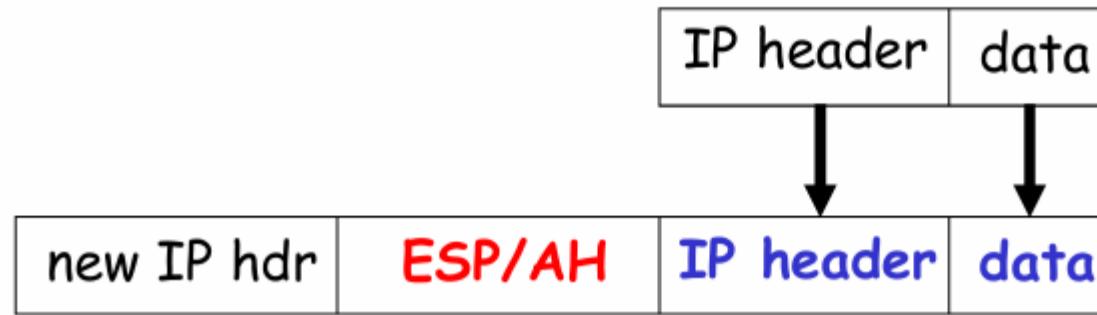
## Transport Mode



端到端

原始的header和数据被保留，可以看到发送者和接收者是谁

## Tunnel Mode

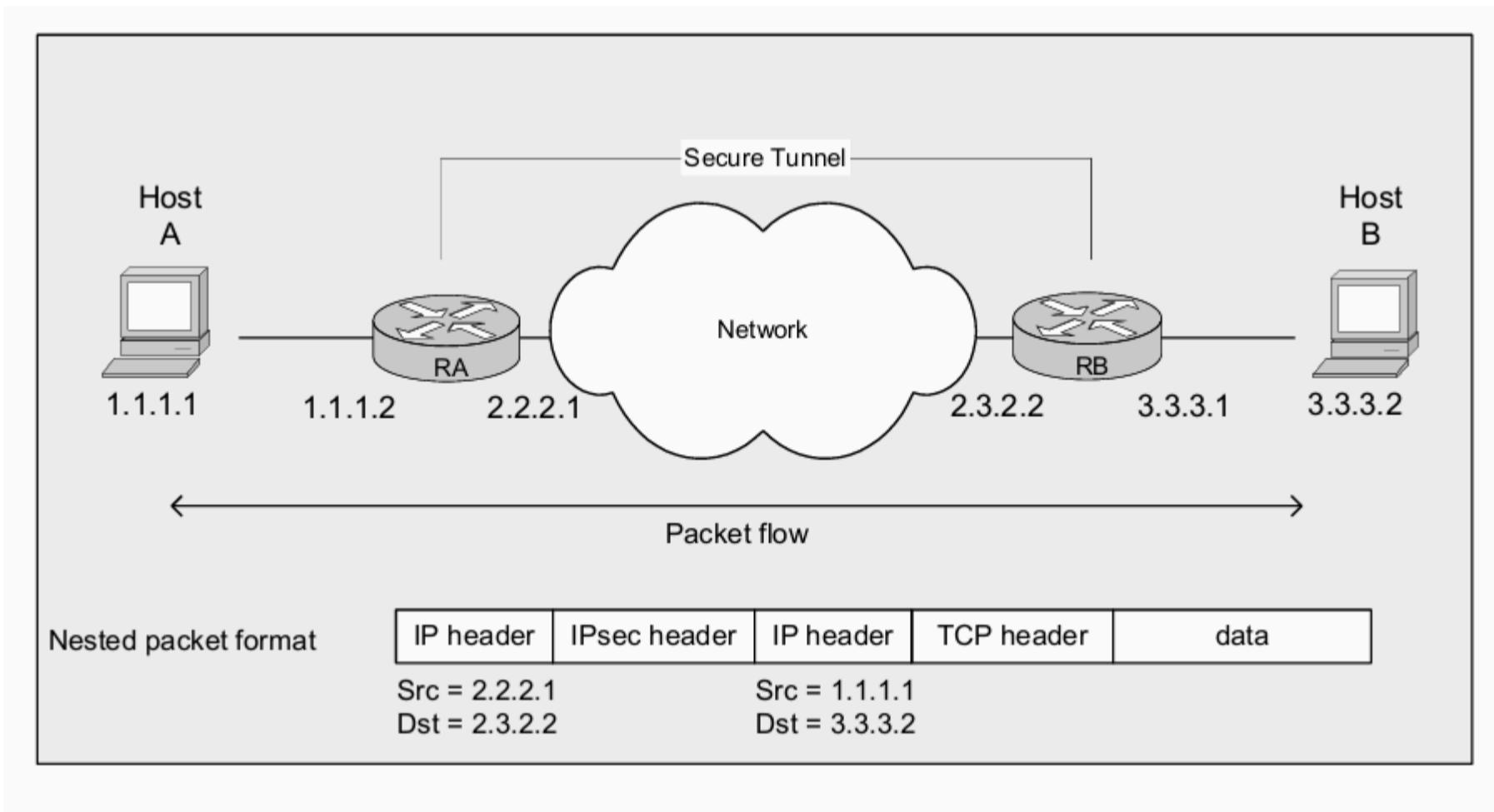


在原header的基础上添加新的header

该模式设计为网关到网关模式

也就是说新包头包含了源网关和目标网关的IP地址

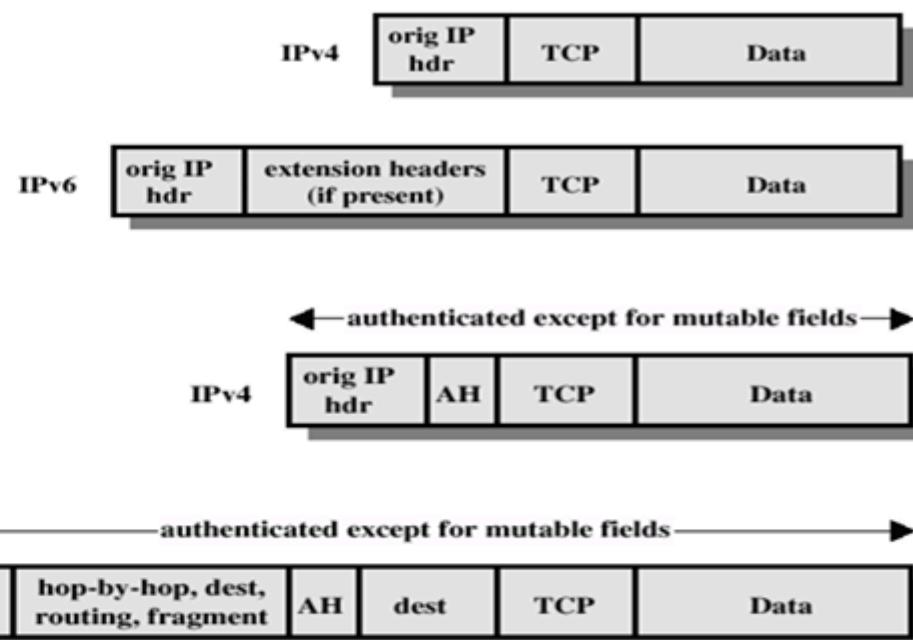
不能直接对IP header进行加密，否则路由器无法得知需要将数据包传输到哪里



A生成数据包，RA为其增加新的header，发送给RB，B de-capsulate后发给B

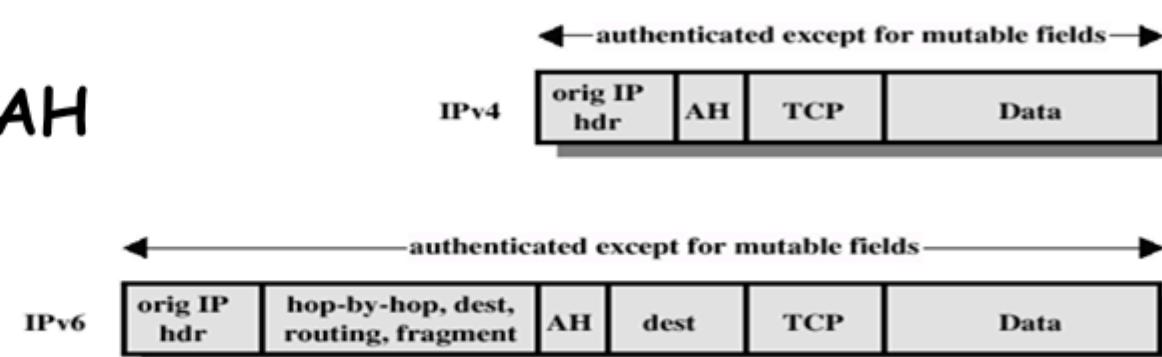
AH

- Original IP packets



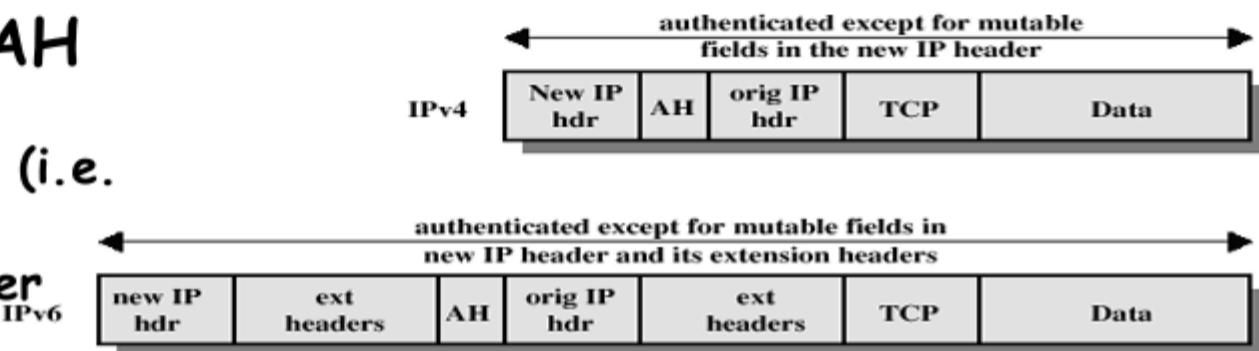
- Transport Mode AH

- Host-to-host authentication



- Tunnel Mode AH

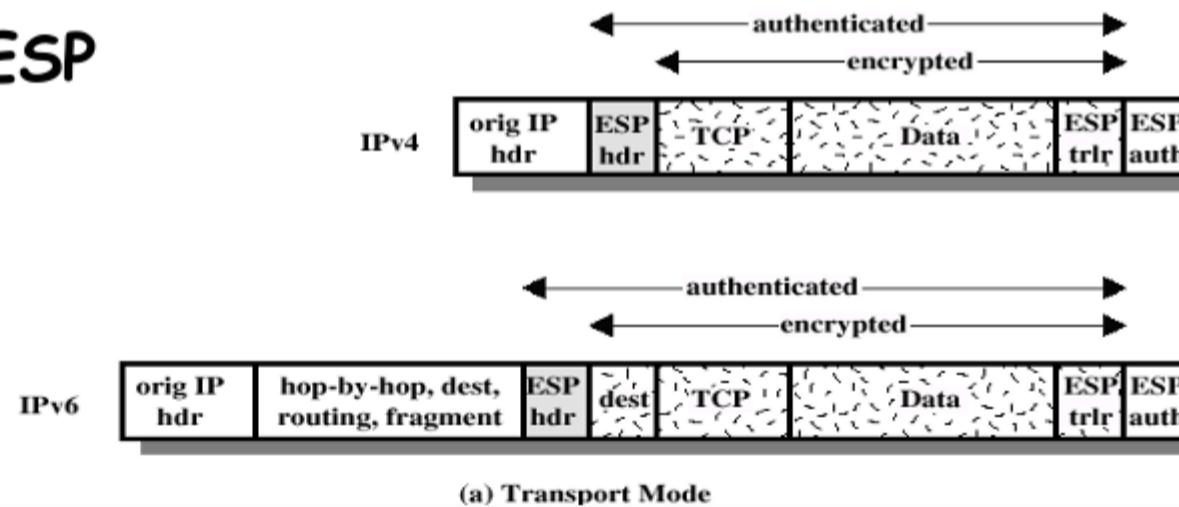
- Host-to-host
- Host-to-router (i.e. remote access)
- Router-to-router



AH为整个报文提供了完整性保护（除header中可变mutable字段）

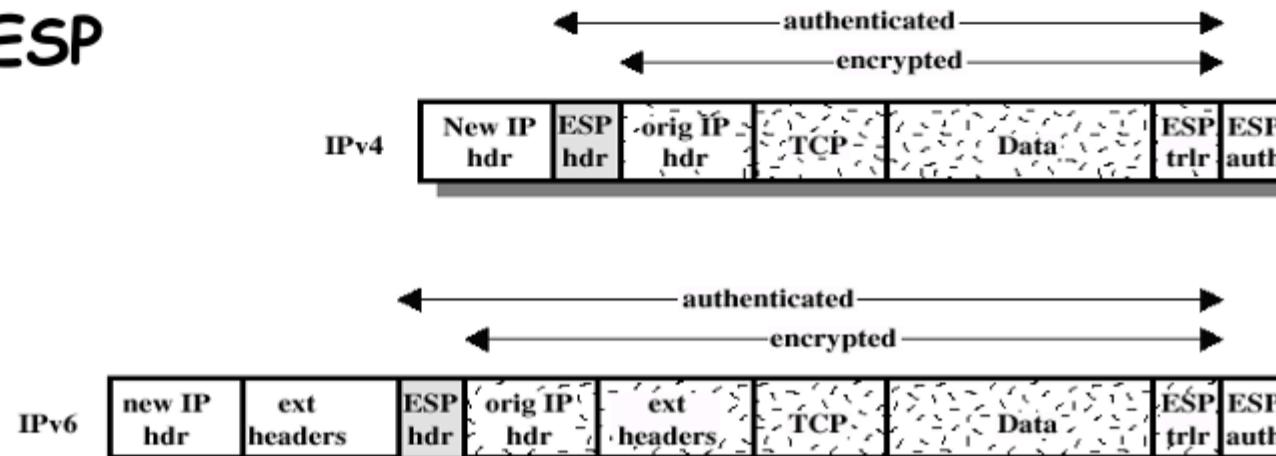
### ESP

- Transport Mode ESP



(a) Transport Mode

- Tunnel Mode ESP



(b) Tunnel Mode

ESP的原始IP头和数据被加密保护

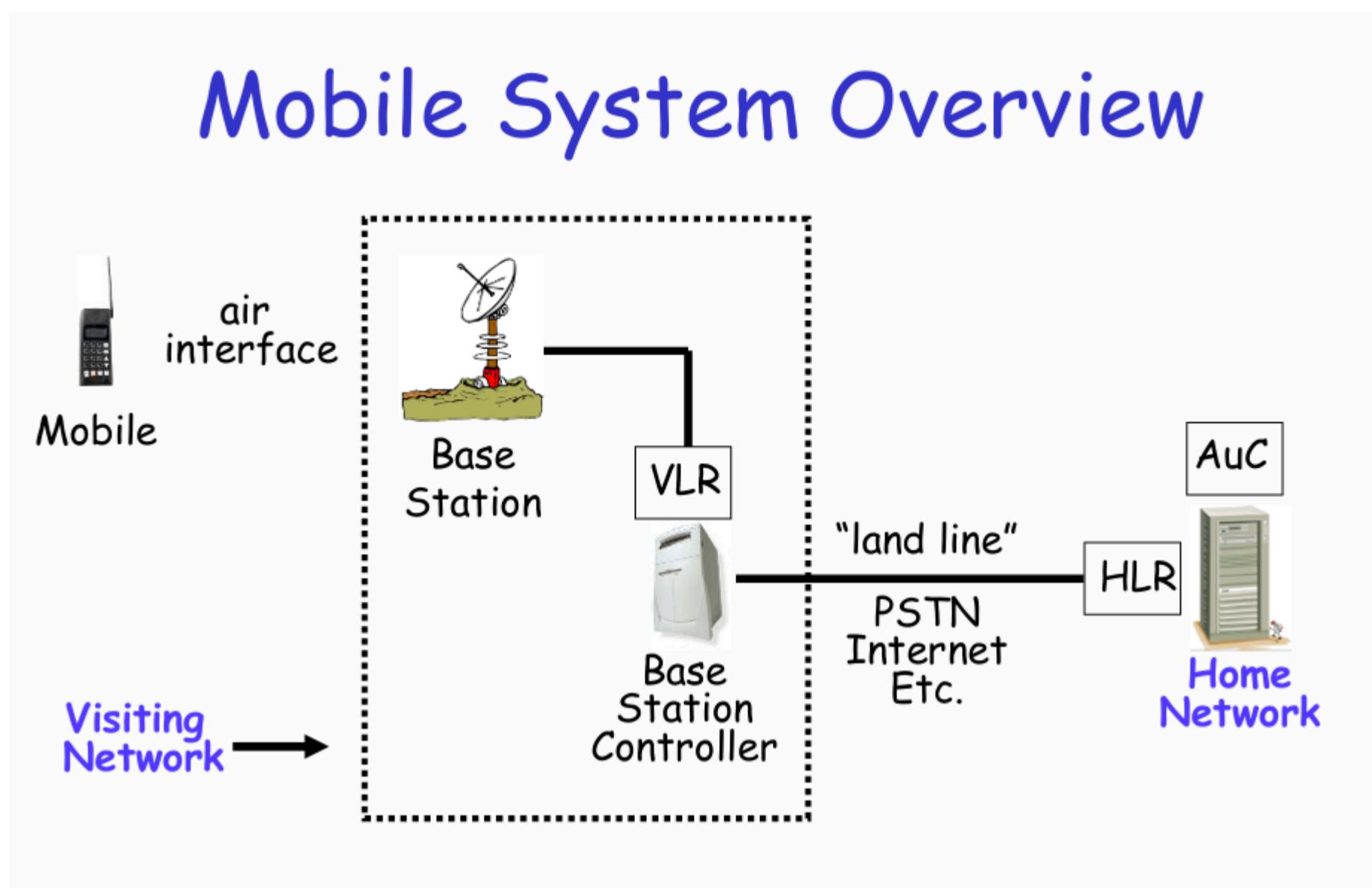
ESP在tunnel mode中对新IP头部不提供任何保护

同时也提供了完整性验证（除可变字段）

生成的新IP头便于路由器寻找目的地

## Mobile Network Security

在移动网络安全中，首先需明确两个世代：GSM(2G), 3GPP (3G)



3个实体：

VLR – Visiting Location Register (拜访位置寄存器): 临时数据库，避免了用户每次请求服务都需要和远方的HLR交互

HLR – Home Location Register (归属位置寄存器): 主要数据库，存储了移动用户的完整信息，通常位于运营商

AuC – Authentication Centre(鉴权中心): 安全认证数据库，保存用户的Ki，保障网络安全

每个移动用户都使用SIM，SIM卡就是安全模块 security module，包含IMSI (International Mobile Subscriber ID)，用户密钥Ki以防篡改

## GSM

### Authentication

Caller被基站base station认证（单向认证）

认证基于挑战-响应模式 challenge-response:

1. AuC生成RAND，并计算使用A3(一种hash) 计算XRES=A3(RAND,Ki)，发送给基站
2. 基站发送使用RAND当作挑战发给用户
3. 用户计算SRES=A3(RAND,Ki)发送给基站
4. 基站验证是否XRES=SRES

注：Ki在始终保存在AuC，不涉及传播

### Confidentiality

使用流密码A5对数据加密，涉及加密密钥Kc

以下过程中RAND的值和authentication中的值相等（认证后进行数据加密）

1. 计算  $Kc = A8(RAND, Ki)$ , A8也是hash。Kc和RAND一同发送给基站
2. 基站发送使用RAND当作挑战发给用户
3. 用户计算  $Kc = A8(RAND, Ki)$
4. 使用A5(Kc)生成密钥流

注：Ki在始终保存在AuC，不涉及传播

总的来说，GSM通过SRES和XRES提供认证，Kc提供保密性，两者都依赖于RAND

## 3GPP

3GPP相比于GSM更有所优化

- 双方认证（避免了假基站问题）
- 密钥（包含加密和完整性）都不可重复利用
- 更强大的加密算法AES

## WLAN Securi

无线网络是如何进行密钥管理的？

手动输入密钥

## WEP

40位或104位密钥，通过外部管理服务（如手动输入）分发给通信实体

使用RC4流密钥进行加密，并使用CRC（循环冗余校验）来验证完整性

缺点：

- 长期密钥 long term key容易被破解
- CRC提供的完整性验证是线性的，性能不佳
- RC4本身容易被破解，安全性不佳

## WPA

相比WEP，密钥频繁更新

但仍然使用RC4

## WPA2

使用会话密钥

使用AES加密算法

## Dos/DDos

Dos: 通过耗尽资源，阻止授权用户访问服务（攻击availability）

注：合法用户的大量使用也可能导致Dos

建立大量连接或消耗大量带宽都是Dos的手段

Dos攻击的一个关键是资源放大 Resource Amplification：攻击者消耗少部分精力消耗受害者大量资源

相比起Dos，DDos的攻击手段建立在许多bot的基础上

通过操控多台bot同时发起连接来消耗资源