

Homework 1

Edited by

牛午甲 PB20111656

潘云石 PB20111657

石磊鑫 PB20111658

孙霄鹄 PB20111659

陈 昊 PB20051077

T1

- 风险
 1. 由一个政府来制定规则，在国际上不能保证其安全性，其他国家也不会认可；
 2. 政府选择一种加密系统，会在市场上直接造成垄断；
 3. 如果改系统出现了漏洞，整个使用该系统的人都会受到影响，扩大了损失范围。
- 降低风险
 1. 用户可选择不使用该系统；
 2. 设计系统时，采取多种加密方式，给用户选择；
 3. 使用开源的加密系统。

T2

Modes of Operation 的必要性

- Block Cipher 基本形式下只能一次加密固定大小的数据块，然而，在许多实际应用中，我们需要加密大于块大小的数据。
- Block Cipher 基本形式是确定性加密，不是 IND-CPA secure 的。

使用不同模式处理分组密码的原因：在实际应用中，需要加密的消息数据量是不定的，数据格式是多种多样的，因此需要不同的模式来进行加密，以提高整体的安全性。
- CBC 模式：对于第一个分组首先构造一个长度为分组长度的初始向量 IV，然后与第一个明文分组进行异或形成第一个密文分组。后续明文分组都与前一个形成的密文分组进行异或后加密。解密将上述过程逆过来即可。
- CTR 模式：CTR 模式通过对逐次累加的计数器 Nonce 加密来生成密钥分组，与明文分组异或产生密文分组。解密时将密文和明文分组位置对调即可。

CTR 与 CBC 模式的对比

| | CBC | CTR |
|---------------|---------------|---------------|
| 加密类型 | 非确定性（随机化）加密方案 | 非确定性（随机化）加密方案 |
| 能否并行 | 解密可以并行，加密只能串行 | 加解密均可并行 |
| Padding | 必需 | 不必需 |
| IV/Nonce 能否重用 | 不能，如果重用泄露消息较少 | 不能，如果重用泄露消息较多 |

注：上述非确定性（随机化）加密是指对于相同的明文也可能产生不同的密文，隐藏了明文的统计特性。

T3

- 产生哈希冲突的原因：通过哈希函数产生的哈希值是有限的，而当输入空间大于输出空间时，就会有不同的输入产生相同的输出哈希值。
- 哈希函数的抗碰撞性，即任何一个攻击者不能在多项式时间内找到 x_1 和 x_2 使得 $x_1 \neq x_2$, 且 $H(x_1) = H(x_2)$ 。

T4

以*.ustc.edu.cn举例
可查看其有效期:



常规(G) 详细信息(D)

证书层次结构

▼ USERTrust RSA Certification Authority

▼ ZeroSSL RSA Domain Secure Site CA

*.ustc.edu.cn

证书字段

▼ 有效期

不早于

不晚于

主题

▼ 使用者公钥信息

使用者公钥算法

使用者公钥

▼ 扩展

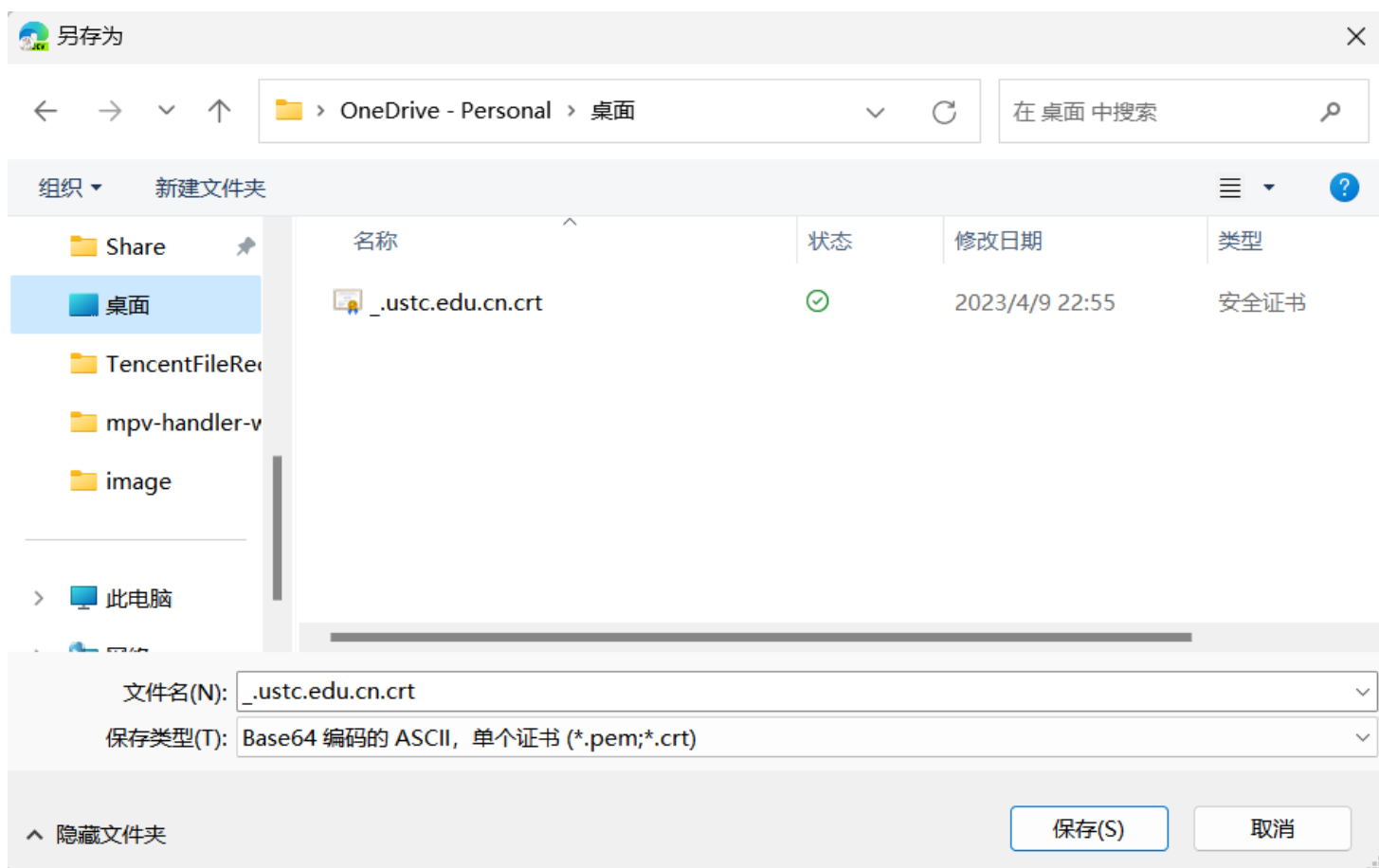
证书颁发机构名称

字段值

2023/5/31 GMT+8 07:59:59

导出(X)...

将其导出为文件:



使用 `certutil -dump` 查看:

```
PS C:\Users\ASUS\OneDrive\Desktop> certutil -dump .\_.ustc.edu.cn.crt
```

X509 证书:

版本: 3

序列号: d9255329a66a8a649acb5e278dd34aec

签名算法:

算法 ObjectId: 1.2.840.113549.1.1.12 sha384RSA

算法参数:

05 00

颁发者:

CN=ZeroSSL RSA Domain Secure Site CA

O=ZeroSSL

C=AT

名称哈希(sha1): 082e3ff9058cfe8a7c18bd13efdf1d1660707a6b

名称哈希(md5): ab1639dd9160fab0f92496ffe91dc2aa

NotBefore: 2023/3/1 8:00

NotAfter: 2023/5/31 7:59

使用者:

CN=*.ustc.edu.cn

名称哈希(sha1): 834dfe3175b0f398066d2e769669b268ac95867d

名称哈希(md5): 4a34771eaf248b73259f03d87f4ccc96

公钥算法:

算法 ObjectId: 1.2.840.113549.1.1.1 RSA

算法参数:

05 00

01f0 b9 75 ae 96 20 d6 6f 67 d0 c7 e9 8c 30 ec eb 33

非根证书

密钥 Id 哈希(rfc-sha1): 344f15a71f03a79a2f50d2caca3ce2a8371e2114

密钥 Id 哈希(sha1): fc846b9674d264639cd2234ad0948b0000a707d8

密钥 Id 哈希(bcrypt-sha1): bdf11100161aa37580e3d28183e306e4d75a7244

密钥 Id 哈希(bcrypt-sha256): bf489eb721fd7e0912d836b23da6aa3896171cb1eb45d770ac97e8ea8374011b

密钥 Id 哈希(md5): d41a567327aa0272d7b4f06f7c9c0611

密钥 Id 哈希(sha256): e06d21d4640940f0e99391d6db09af9c6ce8390210d197c973e91d7b93d23ca6

密钥 Id 哈希(pin-sha256): d10fM6VQyia4oSirPAzLWseTwpxi0euheJtMBDM9x1s=

密钥 Id 哈希(pin-sha256-hex): 775d1f33a550ca26b8a128ab3c0ccb5ac793c29c62d1eba1789b4c04333dc75b

证书哈希(md5): 05d2565b083ae851cb1f6b4944fcd24d

证书哈希(sha1): 63374089d848278f1d9e3ff0a56d449f29cb90db

证书哈希(sha256): 9c1302979b3b941568effc6f7e10b9384cde931ff56e4b2f4b4258d181e44743

签名哈希: 19b5b1eab5eb92e30cf41102d1d357a3131a26fdb43b5926158d959490e88d8afbcacc17da4979919a78add90fd4c38d

CertUtil: -dump 命令成功完成。

PS C:\Users\ASUS\OneDrive\Desktop> |

SHA1和SHA256与浏览器查看一致:

常规(G)

详细信息(D)

颁发给

| | |
|----------|---------------|
| 公用名(CN) | *.ustc.edu.cn |
| 组织(O) | <不是证书的一部分> |
| 组织单位(OU) | <不是证书的一部分> |

颁发者

| | |
|----------|-----------------------------------|
| 公用名(CN) | ZeroSSL RSA Domain Secure Site CA |
| 组织(O) | ZeroSSL |
| 组织单位(OU) | <不是证书的一部分> |

有效期

| | |
|------|------------------------|
| 颁发日期 | 2023年3月1日星期三 08:00:00 |
| 到期日期 | 2023年5月31日星期三 07:59:59 |

指纹

| | |
|------------|--|
| SHA-256 指纹 | 9C 13 02 97 9B 3B 94 15 68 EF FC 6F 7E 10 B9 38 4C DE 93 1F F5 6E 4B 2F 4B 42 58 D1 81 E4 47 43 |
| SHA-1 指纹 | 63 37 40 89 D8 48 27 8F 1D 9E 3F F0 A5 6D 44 9F 29 CB 90 DB |

1. <https://ustc.edu.cn/>

1. (root) USERTrust RSA Certification Authority

- expired: 2038/1/19 GMT+8 07:59:59
- MD5: 1bfe69d191b71933a372a80fe155e5b5
- SHA256: e793c9b02fd8aa13e21c31228accb08119643b749c898964b1746d46c3d4cbd2

2. ZeroSSL RSA Domain Secure Site CA

- expired: 2030/1/30 GMT+8 07:59:59
- MD5: 58aa23107c8d5aedeabd0d5e32578592
- SHA256: 21acc1dbd6944f9ac18c782cb5c328d6c2821c6b63731fa3b8987f5625de8a0d

3. ustc.edu.cn

- expired: 2023/6/15 GMT+8 07:59:59
- MD5: 044a344e9ba6146e4f5db7128ced4269
- SHA256: 8d617c13fa32c7a2e1581c28086648cd698107fe7b57517e1213916e42375852

2. <https://www.12306.cn>

1. (root)CFCA EV ROOT

- expired: 2029/12/31 GMT+8 11:07:01
- MD5: 74e1b6ed267a7a44303394ab7b278130
- SHA256: 5cc3d78e4e1d5e45547a04e6873e64f90cf9536d1ccc2ef800f355c4c5fd70fd

2. CFCA OV OCA

- expired: 2029/12/25 GMT+8 10:02:56
- MD5: fe5a836040d65c90df8131b67f3cf95f
- SHA256: f07bbbde076f9b40c57cc4befede97ca1f53b9ae147f035d284cbf53f3432fb8

3. *.12306.cn

- expired: 2023/10/24 GMT+8 09:49:31
- MD5: b0bc74341ae1ca53e27954560a7e248f
- SHA256: c1b1ba32d88ffaa390a7356c56730443033462f3acf324a4fb4a65364ede1337

3. www.bing.com

1. (root)Baltimore CyberTrust Root

- expired: 2025/5/13 GMT+8 07:59:00
- MD5: acb694a59c17e0d791529bb19706a6e4
- SHA256: 16af57a9f676b0ab126095aa5ebadef22ab31119d644ac95cd4b93dbf3f26aeb

2. Microsoft RSA TLS CA 02

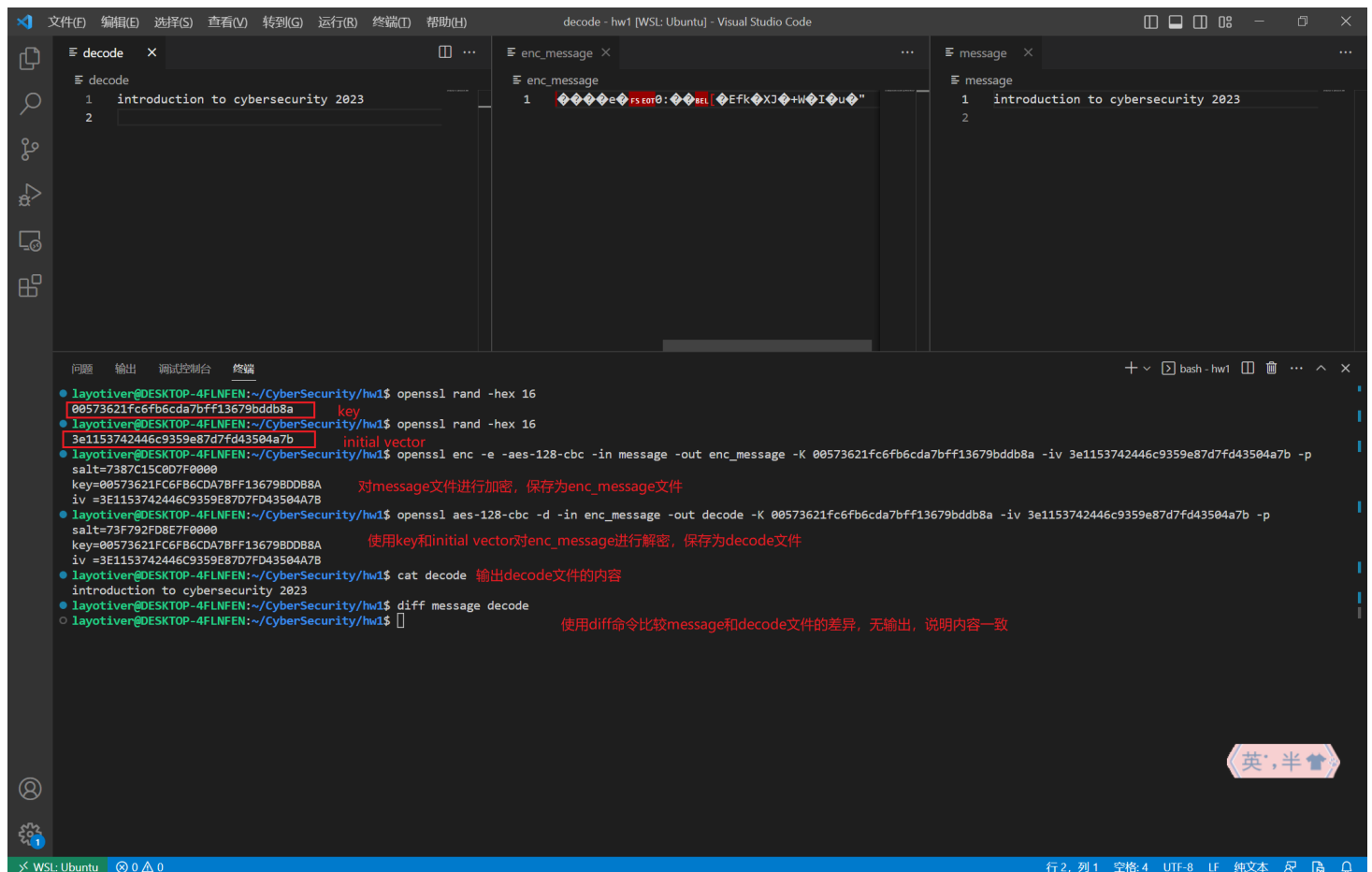
- expired: 2024/10/8 GMT+8 15:00:00
- MD5: 65d17ecae5798c79db8e840fe98a53b9
- SHA256: 05e4005db0c382f3bd66b47729e9011577601bf6f7b287e9a52ced710d258346

3. www.bing.com

- expired: 2023/8/16 GMT+8 11:47:45
- MD5: a59bd11849791d57c1093b57bb9c95f7
- SHA256: e6a984d3550b10540d1429007afb4790af75969382e5403441d6da36963dc35f

T5

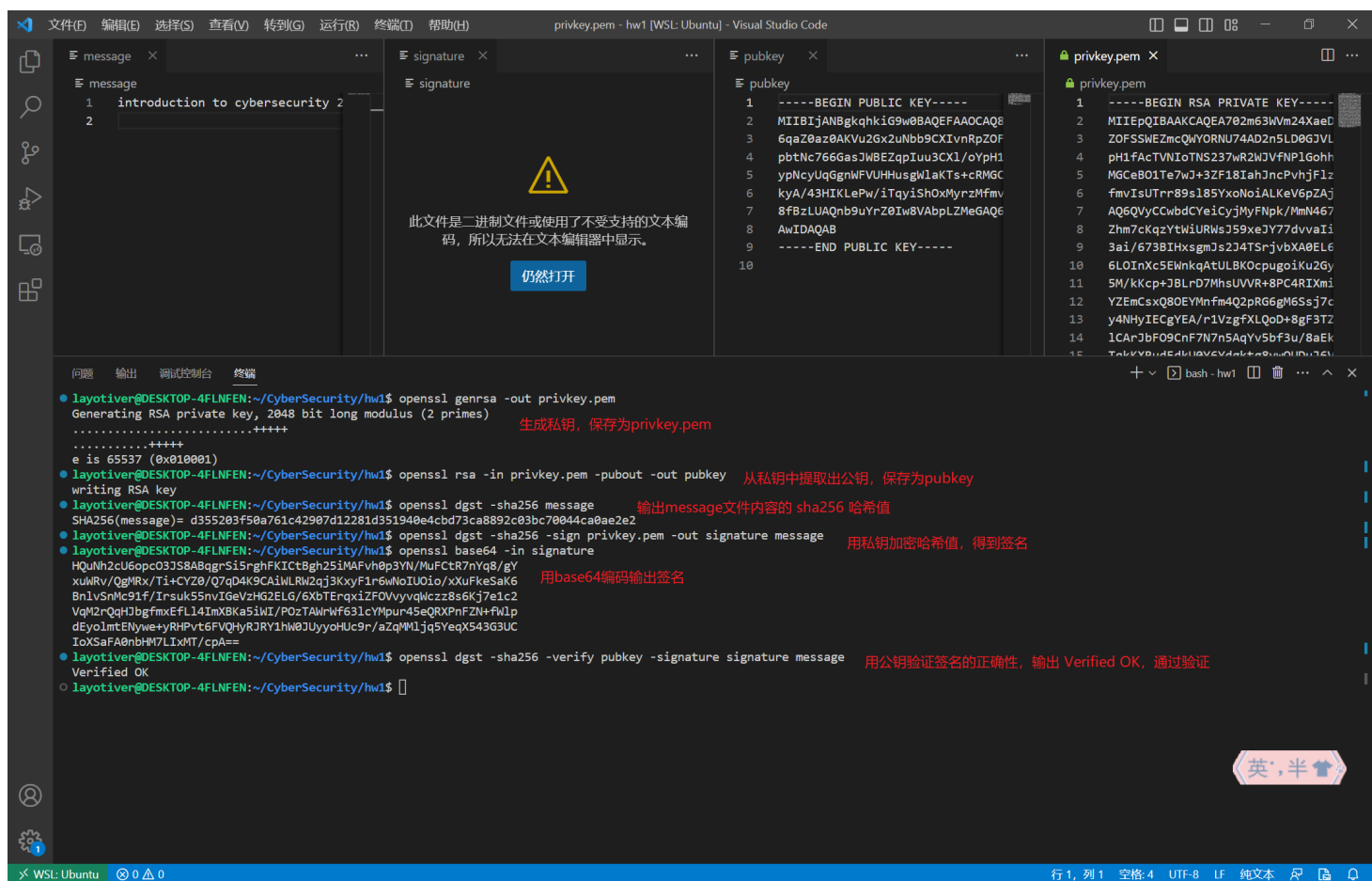
(a) and (b)



```
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ openssl rand -hex 16
00573621fc6fb6cda7bff13679bddb8a
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ openssl rand -hex 16
3e1153742446c9359e87d7fd43504a7b
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ openssl enc -e -aes-128-cbc -in message -out enc_message -K 00573621fc6fb6cda7bff13679bddb8a -iv 3e1153742446c9359e87d7fd43504a7b -p
salt=7387C15C0D7F0000
key=00573621FC6FB6CDA7BFF13679BDDDB8A
iv =3E1153742446C9359E87D7FD43504A7B
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ openssl aes-128-cbc -d -in enc_message -out decode -K 00573621fc6fb6cda7bff13679bddb8a -iv 3e1153742446c9359e87d7fd43504a7b -p
salt=73F792FD8E7F0000
key=00573621FC6FB6CDA7BFF13679BDDDB8A
iv =3E1153742446C9359E87D7FD43504A7B
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ cat decode
introduction to cybersecurity 2023
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$ diff message decode
layotiver@DESKTOP-4FLNFFEN:~/CyberSecurity/hw1$
```

1. 使用 `openssl rand -hex 16` 命令生成16字节的密钥和初始向量，用16进制数表示。
2. 使用 `openssl enc -e -aes-128-cbc` 加密message文件，得到enc_message文件
3. 使用 `openssl aes-128-cbc -d` 命令解密enc_message文件，得到decode文件
4. 输出decode文件的内容，内容正确。比较decode和message的内容，内容一致

(c), (d), and (e)



1. 使用 `openssl genrsa` 命令生成私钥
2. 使用 `openssl rsa` 命令从私钥中提取出公钥
3. 使用 `openssl dgst -sha256` 命令生成message文件的sha256 哈希值
4. 使用 `openssl dgst -sha256 -sign` 命令用私钥加密哈希值，得到签名，保存为signature文件
5. 使用 `openssl base64` 命令用base64编码输出signature文件
6. 使用 `openssl dgst -sha256 -verify` 命令验证签名，输出Verified OK，通过验证

T6 A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms

Summarize

ElGamal 公钥加密系统

- 算法描述
 - 密钥生成 (Alice)

- 选取一个素数 p , 以及模 p 原根 α (本质上是产生一个阶为 p 、生成元为 α 的循环群 G)
- 从 $\{1, 2, \dots, p-2\}$ 中选择一个随机数 x_A
- 计算 $y_A \equiv \alpha^{x_A} \pmod{p}$
- 公布公钥 (p, α, y_A) , 对于私钥为 x_A
- 加密 (Bob)
 - 将消息明文 M 表示为 $\{0, 1, \dots, p-1\}$ 中的一个整数 m
 - 从 $\{1, 2, \dots, p-2\}$ 中选择一个随机数 x_B
 - 计算 $y_B \equiv \alpha^{x_B} \pmod{p}$
 - 计算 $c \equiv my_A^{x_B} \pmod{p}$
 - 发送密文 (y_B, c) 给 Alice
- 解密 (Alice)
 - 计算 $m \equiv c \cdot y_B^{-x_A} \pmod{p}$
 - 恢复明文 M
- 性质
 - 基于 Diffie - Hellman 密钥分发方案实现, 该系统的安全性与 DH 密钥分发方案的安全性等价。
 - $p-1$ 至少要有一个大的素因子, 否则离散对数问题求解会变得很简单^[1]。
 - 密文长度是消息长度的两倍数量级。
 - 为了使 public file 尽可能简洁, p 和 α 可以在一组用户中共享。
 - 不能使用相同的值 k 来加密多个块, 因为如果 k 被使用了多次, 只要入侵者破解了消息的一个块 m , 就能够轻易计算其他块。
 - 攻破这个系统的难度和攻破 Diffie - Hellman 密钥分发方案的难度是一样的。
 - ElGamal 公钥加密系统 v.s. RSA 公钥加密系统
 - 必须使用和 RSA 系统中使用的数字相同的量级来获得相同级别的安全性。
 - public file 的大小比 RSA 的大。
 - 概率性加密方案: 由于加密操作的随机性 (随机数 k 的选取), 对同一消息加密两次, 将不会得到相同的密文。而 RSA 公钥系统是确定性加密方案。
 - 假设 p 的大小与 RSA 系统中 n 的大小相同。则该密文的大小是对应 RSA 密文大小的两倍。

ElGamal 签名方案

- 算法描述
 - 密钥生成
 - 选择一个素数 p , 以及模 p 原根 α
 - 从 $\{1, 2, \dots, p-2\}$ 中选择一个随机数 x
 - 计算 $y \equiv \alpha^x \pmod{p}$
 - 公布公钥 (y, α, p) , 私钥为 x , α 和 p 可以由一组用户共享。

- 签名生成
 - 设被签名消息为 m (可以是原始消息摘要, 表示为小于 p 的数值)
 - 选择一个随机数 k , 满足 $(k, p-1) = 1$, 计算: $r \equiv \alpha^k \pmod{p}$
 - 利用扩展 Euclidean 算法求解下列方程中的 s : $m \equiv rx + ks \pmod{p-1}$ 签名就是 (r, s) 。
- 签名验证
 - 验证者计算 $y^r r^s \pmod{p}$ 和 $\alpha^m \pmod{p}$
 - 若二者相等, 签名验证通过, 否则签名无效。
- 性质
 - 为了使 public file 尽可能简洁, p 和 α 可以在一组用户中共享。
 - 选择的 k 值不能使用超过一次。
 - ElGamal 签名方案 v.s. 使用 RSA 的数字签名
 - 两种方案的签名都是是消息长度的两倍。
 - ElGamal 签名的长度与 RSA 方案所需的长度相同, 并且是 Ong 和 Schnorr 发布的基于二次型的新签名方案的一半大小。
 - 签名需要一次幂运算和若干次乘法; 验证签名, 直观上需要三次幂运算, A. Shamir 提出了只需要1.875次幂运算的改进算法。

可能的攻击方法 (个人认为不是这篇论文重点, 但是也罗列一下)

恢复秘密密钥 x 的攻击

- 突破口 1: 给定消息序列 $\{m_i : i = 1, 2, \dots, l\}$ 以及对应的签名序列 $\{(r_i, s_i) : i = 1, 2, \dots, l\}$, 攻击者希望从 l 个线性同余方程 $m_i \equiv xr_i + k_i s_i \pmod{p-1}$ 中解出 x, k_1, k_2, \dots, k_l
 - 注意: 加密每个块时 k 都会更换, 但是 x 是不换的。因为 x 是由公钥 (y, g, p) 唯一确定的。
- 求解难度: 在模 $p-1$ 的意义下, x 的每一个取值可以确定 (至少) 一组 $\{k_1, k_2, \dots, k_l\}$ 的取值。由于 $p-1$ 至少有一个大的素因子 q , 所以希望求解 $x \pmod{q}$ 需要指数量级的消息-签名对。
 - 注意: 只有有一个 k_i 被使用了两次, x 便可以唯一确定下来。所以为了保证系统的安全性, 任何 k 值都不应该被使用两次。
- 突破口 2: 直接从 $\alpha^m \equiv y^r r^s \pmod{p}$ 求解 m 。
- 求解难度: 显然与求解离散对数问题难度相同。
- 突破口 3: 出发点与突破口 1 类似, 尝试在 k_1, k_2, \dots, k_l 之间建立一些线性依赖关系。如设 $k_i \equiv ck_j \pmod{p-1}$, 以达到消掉消元效果。
- 求解难度: 如设 $k_i \equiv ck_j \pmod{p-1}$, 就有 $r_i \equiv r_j^c \pmod{p}$, c 的求解是离散对数求解问题。

伪造签名的攻击

- 思路 1: 给定消息 m , 攻击者尝试不恢复私钥 x , 直接找满足 $\alpha^m \equiv y^r r^s \pmod{p}$ 的 r, s 。

- 求解难度：如果先固定 s ，那么使用上式求解 r **还没有证明至少和离散对数的求解问题一样困难**；论文假设多项式时间无法求解。可能存在同时求解 r, s 的算法，论文暂未发现。
- 思路 2：攻击者知道一条消息的及其合法签名，就可以生成其他合法签名-消息对。**这种攻击不允许入侵者对任意消息进行签名**，因此不会破坏系统。

这个属性存在于所有现有的数字签名方案中，可以通过要求 m 必须具有某种结构或在签名之前对消息 m 应用单向函数来避免。

- 攻击方法：已知消息 m 的签名 (r, s) ，故有 $\alpha^m \equiv y^r r^s \pmod{p}$ 。选取随机整数 A, B, C ，使得 $(Ar - Cs, p - 1) = 1$ 。令 $r' = r^A \alpha^B y^C \pmod{p}$ ， $s' = sr'(Ar - Cs)^{-1} \pmod{p - 1}$ ， $m' = r'(Am + Bs)(Ar - Cs)^{-1} \pmod{p - 1}$ ，则 (r', s') 是消息 m' 的签名。

特别地，取 $A = 0$ 不需要已知任何消息-签名对就可以生成合法的消息-签名对。

Critical Reviews

Pros

- 论文整体：整体的思路架构比较清晰，新的数字签名方法具有开创性。
- 具体细节：ElGamal 公钥加密系统和 RSA 公钥系统相比，概率性的加密方案增加了攻破系统的难度，也一定程度上抵御了重放攻击和选择明文攻击。此外，ElGamal 公钥加密系统使用非对称加密算法，能够实现安全的密钥交换，避免了密钥传输过程中的安全问题。

Cons

- 论文整体：感觉一些技术细节描述不是特别清晰（存疑的地方已在上面标出），也没有明确给出证明过程或者出处；算法相对比较粗糙，没有考虑一些极端情形；缺少实验验证的环节，没有数据支撑。
- 具体细节：ElGamal 公钥加密系统和 RSA 公钥系统相比，由于使用了较多指数运算，导致加密速度较慢，计算开销大；由于 ElGamal 公钥加密算法需要产生两个密文分量，密文长度较大，是对应 RSA 密文大小的两倍，会增加网络传输和存储的负担。ElGamal 签名方案同样需要进行指数运算，生成签名以及验证签名速度较慢，计算开销大；此外，ElGamal 签名方案的签名需要包含两个数值，长度较长。

Reference

[1] An Improved Algorithm for Computing Logarithms over $GF(p)$ and its cryptographic significance.
<https://ee.stanford.edu/~hellman/publications/28.pdf>