实验五数字签字算法 DSA

1611531-信息安全-刘新慧

摘要

通过对数字签字算法 DSA 的实际操作，理解 DSS 的基本工作原理。

目录

1 实验内容和步骤

1、为了加深对 RSA 算法的了解，根据已知参数：p = 3，q = 11，m = 2，手工计算公钥和私钥，并对明文 m 进行加密，然后对密文进行解密。

2、编写一个程序，用于生成 512 比特的素数。

3．利用 2 中程序生成的素数，构建一个 n 的长度为 1024 比特的 RSA 算法，利用该算法实现对明文的加密和解密。

4、在附件中还给出了一个可以进行 RSA 加密和解密的对话框程序 RSATool，运行这个程序加密一段文字，了解 RSA 算法原理。

2 实验原理

2.1 DSA 原理

以往的文件或书信可以通过手写亲笔签名来证明其真实性，而通过计算机网络传输的信息则通过数字签字技术实现其真实性的验证。

数字签字目前采用较多的是非对称加密技术，其实现原理简单的说，就是由发送方利用杂凑函数对要传送的信息进行计算得到一个固定位数的消息摘要值，用发送者的私钥加密此消息的杂凑值所产生的密文即数字签字。然后将数字签字和消息一同发送给接收方。接收方收到消息和数字签字后，用同样的杂凑函数对消息进行计算得到新的杂凑值，然后用发送者的公开密钥对数字签字解密，将解密后的结果与自己计算得到的杂凑值相比较，如相等则说明消息确实来自发送方。

下面我们以 DSA（Digital Signature Algorithm）为例，介绍数字签字算法。DSA 源于 Elgamal 签名算法，被美国 NIST 采纳为 DSS（Digital Signature Standard）数字签名标准。下面介绍 DSA 的算法描述：

1．全局变量的设置

(1) 素数 p，$2^{511} < p < 2^{512}$;

(2) q 是 p-1 的一个素因子，$2^{159} < q < 2^{160}$;

(3) 其中 h 是整数，1<h<(p-1).

2．私钥：

私钥 x 是随机数或伪随机数，其中 0<x<q.

3. 公钥：

y=g^xmod p，（p，q，g，y）为公钥。

4．用户的随机数选择：

k 为随机数或伪随机数，其中 0<k<q。

基于以上选择的参数，DSA 的签字过程如下：

$$\gamma = (g^k mod p) mod q$$
$$\delta = (k^{-1}(H(m) + xr)) mod q$$

则 (γ, δ) 为对消息 m 的签字，显然，签字长度为 320 比特。

其中，H 是一个安全杂凑函数，在 DSS 标准中，采用 SHA-1 算法作为安全的杂凑函数。

签字完成后，把对消息 m 的数字签字和消息 m 一同发送给接收方。接收方接收到消息 m 和数字签字后，对数字签字的验证过程如下：

计算：

$$w = \delta^{-1} mod q$$
$$1 = H(m)wmodq$$
$$2 = \gamma wmodq$$

如果有：$((g^{u1}y^{u2})modp)modq = \gamma$ ，则说明信息确实来自发送方。否则，签字是无效的。

2.2 杂凑函数 SHA

SHA-1 算法是 SHA 家族成员之一。

SHA (Secure Hash Algorithm，译作安全散列算法) 是美国国家安全局 (NSA) 设计，美国国家标准与技术研究院 (NIST) 发布的一系列密码散列函数。正式名称为 SHA 的家族第一个成员发布于 1993 年。然而人们给它取了一个非正式的名称 SHA-0 以避免与它的后继者混淆。两年之后，SHA-1，第一个 SHA 的后继者发布了。另外还有四种变体，曾经发布以提升输出的范围和变更一些细微设计：SHA-224, SHA-256, SHA-384 和 SHA-512 (这些有时候也被称做 SHA-2)。

最初载明的算法于 1993 年发布，称做安全散列标准 (Secure Hash Standard)，FIPS PUB 180。这个版本常被称为"SHA-0"。它在发布之后很快就被 NSA 撤回，并且以 1995 年发布的修订版本 FIPS PUB 180-1 (通常称为"SHA-1") 取代。根据 NSA 的说法，它修正了一个在原始算法中会降低密码安全性的错误。然而 NSA 并没有提供任何进一步的解释或证明该错误已被修正。1998 年，在一次对 SHA-0 的攻击中发现这次攻击并不能适用于 SHA-1 —我们不知道这是否就是 NSA 所发现的错误，但这或许暗示我们这次修正已经提升了安全性。SHA-1 已经被公众密码社群做了非常严密的检验而还没发现到有不安全的地方，它被认为是安全的。

SHA-0 和 SHA-1 会从一个最大 2^64 位元的讯息中产生一串 160 位元的摘要，然后以设计 MD4 及 MD5 讯息摘要算法的 MIT 教授 Ronald L. Rivest 类似的原理为基础来加密。

SHA-1 的描述：

以下是 SHA-1 算法的伪代码：

```
(Initialize variables:)
a = h0 = 0x67452301
b = h1 = 0xEFCDAB89
c = h2 = 0x98BADCFE
d = h3 = 0x10325476
e = h4 = 0xC3D2E1F0
(Pre-processing:)
paddedmessage = (message) append 1
while length(paddedmessage) mod 512 <> 448:
paddedmessage = paddedmessage append 0
paddedmessage = paddedmessage append (length(message) in 64-bit format)
(Process the message in successive 512-bit chunks:)
while 512-bit chunk(s) remain(s):
break the current chunk into sixteen 32-bit words w(i), 0 <= i <= 15
```

```
(Extend the sixteen 32-bit words into eighty 32-bit words:)
for i from 16 to 79:
w(i) = (w(i-3) xor w(i-8) xor w(i-14) xor w(i-16)) leftrotate 1
(Main loop:)
for i from 0 to 79:
temp = (a leftrotate 5) + f(b,c,d) + e + k + w(i) (note: all addition is mod 2^32)
where:
(0 <= i <= 19): f(b,c,d) = (b and c) or ((not b) and d), k = 0x5A827999
(20 <= i <= 39): f(b,c,d) = (b xor c xor d), k = 0x6ED9EBA1
(40 <= i <= 59): f(b,c,d) = (b and c) or (b and d) or (c and d), k = 0x8F1BBCDC
(60 <= i <= 79): f(b,c,d) = (b xor c xor d), k = 0xCA62C1D6
e = d
d = c
c = b leftrotate 30
b = a
a = temp
h0 = h0 + a
h1 = h1 + b
h2 = h2 + c
h3 = h3 + d
h4 = h4 + e
digest = hash = h0 append h1 append h2 append h3 append h4
```

在 FIPS PUB 180-1 展示的构想，用以下的公式替代可以增进效能：

$$(0 <= \text{i} <= 19): F(B,C,D) = D \oplus (B \wedge (C \oplus D))$$
$$(40 <= \text{i} <= 59): F(B,C,D) = (B \wedge C) \vee (D \wedge (B \vee C))$$

SHA-1 在许多安全协议中广为使用，包括 TLS 和 SSL、PGP、SSH、S/MIME 和 IPsec，曾被视为是 MD5（更早之前被广为使用的散列函数）的后继者。

2.3 SHA 与 MD5 比较

1 对强行攻击的安全性; 最显著和重要的区别是 SHA-1 摘要长 32 位, 使用强行技术. 产生任何一个报文使其摘要等于给报摘要的难度对 MD5 是 ^128 数量级的操作, 而对 SHA-1 则是 2^160 数量级的操作, 这样,SHA-1 对强行攻击有更大的强度,

2 对密码分析的安全性: 由于 MD5 的设计. 易受密码分析的攻击,SHA-1 显得不容易受这样的攻击.

3 速度: 在相同的硬件上,SHA-1 的运行速度比 MD5 慢.

3 程序演示

打开程序运行，通过移动鼠标获得种子进而生成 DSA 数字签名算法的全局公开钥 p,q,g 以及用户公开钥 y，用户私密钥 x。

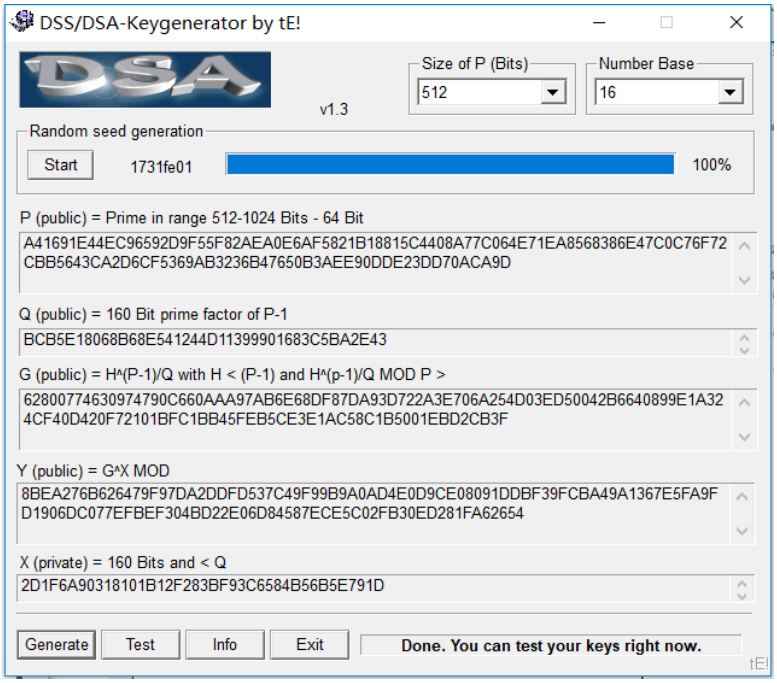

图 1: 获得种子

对消息进行签名：

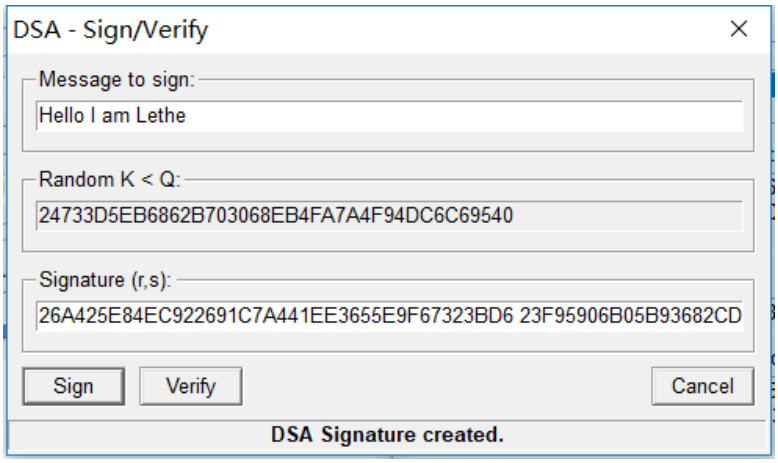

图 2: 进行签名

进行验证，消息正确则验证成功，消息错误则验证失败。

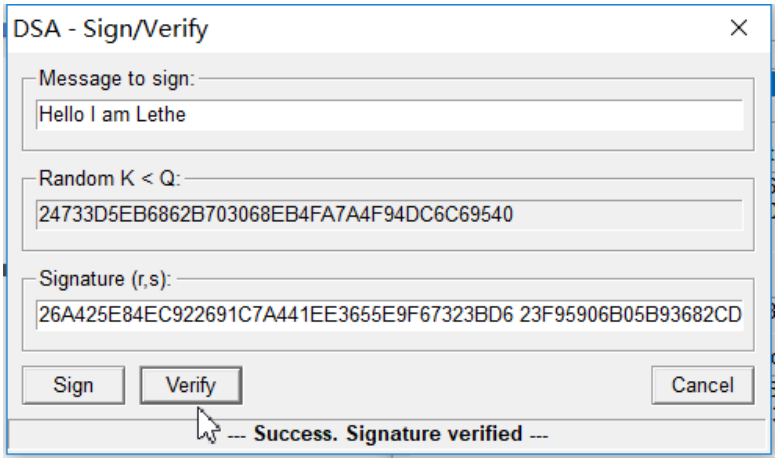

图 3: 验证成功

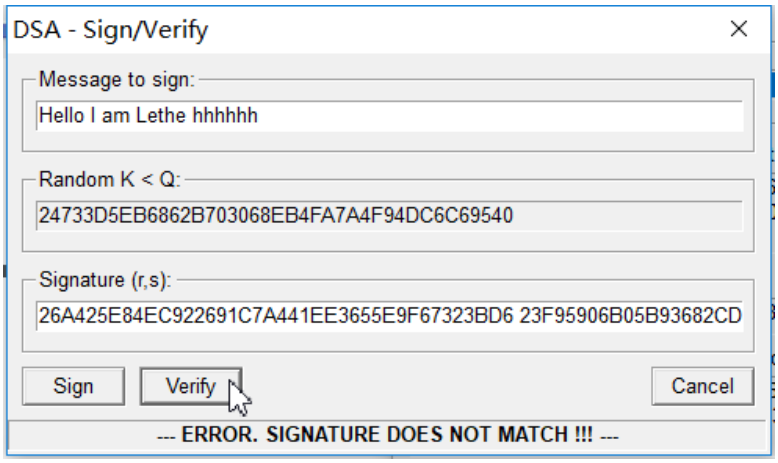

图 4: 验证失败