Computer Security and Cryptography

CS 4840 - Spring 2016

Term Project

INTRODUCTION

Your term project will be to implement a demonstration version of the Rivest/Shamir/Adleman (RSA) public key cryptography system.

For the first pass, to keep things simple, we will be using comparatively "small" (∼15 bits) rather than "large" (∼2048 bits) primes. This will allow developing code for 64-bit machines using native integer operations, without having to write "multiple precision" integer code.

BASIC IDEA of RSA

Remember the basic process:

Step 1 – find two primes

Example: $p = 32003, q = 31511$

Step 2 – set $n = p * q$

$n = 1008446533 = 32003 * 31511$

Step 3 – Note that $\varphi(n) = (p - 1) * (q - 1)$

$\varphi(1008446533) = 1008383020 = (32003 - 1) * (31511 - 1)$

Step 4 – We choose a (relatively small) encryption exponent e (making sure $GCD(e, \varphi(n)) = 1$)

$e = 65537$

Step 5 – Calculate $1 = GCD(e, \varphi(n)) = f * e + x * \varphi(n)$ (using the extended GCD algorithm)

$1 = 120429893 * 65537 + -7828 * 1008383020$

Step 6 – Set the decryption exponent d to be such that $1 < d < \varphi(n)$
and $d \equiv f \bmod \varphi(n)$

$d = 120429893$

Step 7 – The "public key" is the pair (e, n)

$(e, n) = (65537, 1008446533)$

I keep my "private key" d secret.

When someone wants to send me an encrypted message M, they break M up into blocks, and represent it as a sequence of integers M_i, with $1 < M_i < n$ – i.e., as bit strings,

$$M = M_1 M_2 \dots M_i \dots M_k,$$

and they transmit to me the sequence of values

$$\mathit{Encrypt}(M_i) = (M_i^e \bmod n)$$

I can then decrypt, using

$$\begin{aligned} \mathit{Decrypt}(\mathit{Encrypt}(M_i)) &= \mathit{Encrypt}(M_i)^d \bmod n \\ &= (M_i^e)^d \bmod n \\ &= (M_i^{ed}) \bmod n \\ &= M_i \end{aligned}$$

So, for example,

$$123456^{65537} \bmod 1008446533 = 123667454$$

and

$$123667454^{120429893} \bmod 1008446533 = 123456$$

PROJECT (MINIMUM)

Develop an implementation of the RSA system, using native 64-bit calculations, and primes p and q of approximately 15 bits.

You should be able to publish your "public key pair" (e, n) (it's reasonable to use $e = 65537$). Publish n as a hex number.

In the example above, the "public key pair" would be (65537, 3C1BAC45).

Messages will be broken up into blocks of 3 ascii characters (24 bits), and "transmitted" as blocks of 8 hex digits.

In the example above, the message string "abcdef" would be transmitted as
3267F007 0FD08F6B

For testing and validation, exchange public key pairs with other teams, and "transmit" encrypted messages to each other.

For an additional test, send a "signed" message encrypted with your secret key. Another team should be able to decrypt your "signed" message using your public key.

PROJECT (ASPIRATIONAL)

Develop an implementation of the RSA system, using primes p and q of approximately 256 bits. One way to find "large" primes is to use the 'openssl' command (available on the Mac computers). That command has a 'prime' subcommand:

```
OpenSSL> prime 11111222223333344444555556666677777888889999191
 137671C1DB429B6FDBE64A659B6DE4875596B5D7 is prime
```

You should be able to publish your "public key pair" (e, n) (it's reasonable to use $e = 65537$). Publish n as a hex number.

Messages will be broken up into blocks of 60 bytes (480 bits) . . . make sure that your modulus satisfies $n > 2^{500}$.

To make this work, you will have to implement (or link libraries for) "multiple precision" integer arithmetics, including multiplication, raising to a power, and mod . . .

For testing and validation, exchange public key pairs with other teams, and "transmit" encrypted messages to each other.

For an additional test, send a "signed" message encrypted with your secret key. Another team should be able to decrypt your "signed" message using your public key.

A Couple of Examples

Here are a couple of examples of encrypted files.

The first has been encrypted with a public key pair. The first two lines are the public key pair that have been used to encrypt the file. If you know the corresponding private key pair (see above), you should be able to decrypt it:

```
65537
3C1BAC45
2F793D99 24EE975F 0CF251A5 0C9E718E 0B2FCA65 2D74C7A9 2B6C0EB0 1ECFF5A0
0D576EBC 110EC31F 01891CA5 00DCBE4C 2B0A635E 0E23E0B7 1D29C991 1CB99176
2AF7E5F3 2E7D07C2 0A8BC609 163951F3 2C6F9F04 214FB386 38A90BEB 1B45DA85
1ECFF5A0 07597F26 2210F69A 01891CA5 00DCBE4C 25A8062C 2DC11F56 228C5D41
1BE05C03 284CB18E 01B4A61D 2CF180E4 1B0003DA 02E90DE5 21A9FC1A
```

The second has been encrypted with a private key (i.e., “signed”). The corresponding public key pair is near the beginning of the file, and should be able to decrypt the file:

```
This is signed (encrypted with a private key).
You should be able to decrypt with the corresponding public key:
65537
3C1BAC45
18D59952 13893990 1CC547B8 36527CFD 37ADD4FF 3957ED8E 08712795 20A871EA
03EC9F19 11486441 1C7BF650 2D0C2DF1 234B618A 042544B4 04020621 1DE4C13B
30E8F787 0D392A34 340315C6 2176EFCA 387E23E4 0464AEF5 322DD075 1FF83097
32F9EB6C 2CACE584 30593265 374AF861 37551B3B 0F479C81 079865FB 138AFBCF
29B72792 0D392A34 07959522 22446BDA 3239E140 17D48F08 081DCB2E 2D5E523B
248B1B53 3AA3886E 330F9770 1A23EDEE 04EC78AF 140F5905 1385A7E9 20E3D18B
22268F80 18221105 31E8B4C7 06AB6AC2 230E3CEF
```