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,

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

and they transmit to me the sequence of values

$$Encrypt(M_i) = (M_i^e \bmod n)$$

I can then decrypt, using

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

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 1111122222333334444455555666667777788888899999191
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.