Encryption:
The RSA Public Key Cipher

Michael Brockway

March 5, 2018

Overview

Transport-layer security employs an asymmetric public cryptosystem to allow two parties (usually a client application and a server) to authenticate each other and negogiate parameters for their conversation.

This lecture describes RSA, the cryptosystem devised by Rivest, Shamir and Adleman and commonly used for this putpose.

RSA makes heavy use of *modular arithmetic*; we start start with a reminder about this and an overiview of some key tools and principles needed for RSA.

Modular Arithmetic

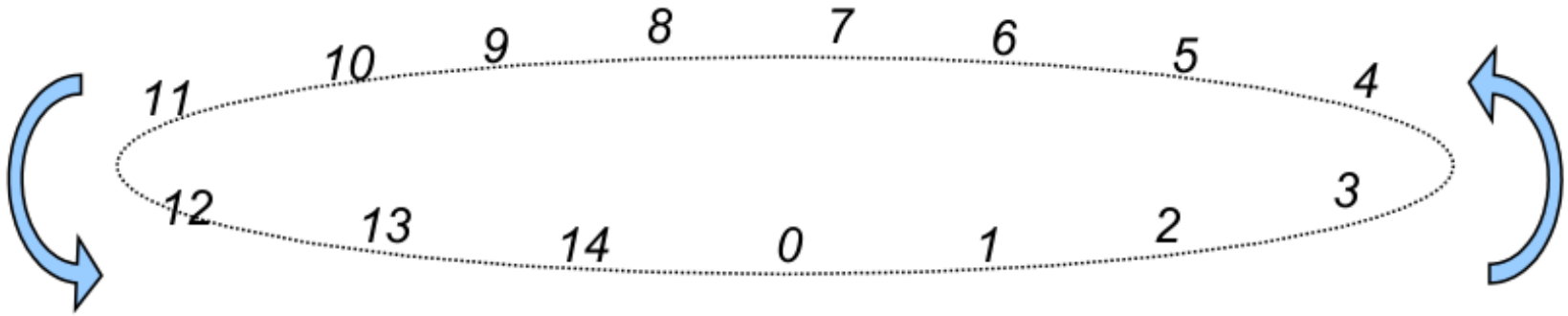
In C/C++/Java, if `a`, `n` are `int` variables, the expression `a % n` denotes the *remainder* when `a` is integer-divided by `n`.

In more mathematical language, we can say *a, b are congruent* (or *equivalent*) *modulo n*, written $a \equiv_n b$ to mean that a, b have the same remainder on integer-division by n: $a\%n = b\%n$.

Modulo n arithmetic features integers 0, 1, 2, 3, ... n-1. When you count up to n you wrap around to 0. Instead of a 'number line'

Modular Arithmetic - plus and minus

... we have a 'number circle': here is one for modulo-15:



$(5 + 12) \equiv_{15} 17 \equiv_{15} 2$;

- ... or count around the circle: start from 5, count on 12, end up on 2.

$3 - 8 \equiv_{15} 10$

- start from 3, count back 8.
- 0r if you prefer, $(3 - 8) \equiv_{15} (15 + 3 - 8) \equiv_{15} 10$.

Modular Arithmetic - multiplication

We can multiply: $8 \times 5 \equiv_{15} 40 \equiv_{15} 10$. Check that starting from 0 and counting 8 lots of 5 around the circle, you end up at 10.

Figure: Modulo 15 multiplication table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	0	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	0	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	0	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	0	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	0	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	0	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	0	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	0	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	0	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	0	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	0	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Modular Arithmetic - reciprocals

It is possible to divide, *sometimes*. Interestingly 5 has no multiplicative inverse, or reciprocal modulo 15 (is there a 1 in row 5 of the multiplication table?) but 8 does: there is a 1 in row 8, in column 2; so 2 is the mod 15 reciprocal of 8 (and *vice versa*).

x has a reciprocal modulo n (a number y such that $x \times y \equiv_n 1$) if and only if x *has no factor* (other than 1) in common with n. (x and n are *relatively prime*).

Since 5 has a factor in common with 15, 5 has no reciprocal modulo 15.

The reciprocal of x modulo n we write $= x^{-1} mod\ n$.

Rows 0, 3, 5, 6, 9, 10, 12 of modulo 15 multiplication tabledo not have a 1 in. These numbers have a factor in common with 15, and so do not have a reciprocal modulo 15.

- 2 has a reciprocal: 8 ($2 \times 8 \equiv_{15} 1$ so $2^{-1} mod\ 15 = 8$)
- 13 has reciprocal, 7
- 4 is its own reciprocal. So is 11.

Modular Arithmetic - reciprocals

When the modulus n is a *prime* number all numbers other than 0 have a reciprocal modulo n. Here is the modulo-13 multiplication table:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	4	6	8	10	12	1	3	5	7	9	11
3	0	3	6	9	12	2	5	8	11	1	4	7	10
4	0	4	8	12	3	7	11	2	6	10	1	5	9
5	0	5	10	2	7	12	4	9	1	6	11	3	8
6	0	6	12	5	11	4	10	3	9	2	8	1	7
7	0	7	1	8	2	9	3	10	4	11	5	12	6
8	0	8	3	11	6	1	9	4	12	7	2	10	5
9	0	9	5	1	10	6	2	11	7	3	12	8	4
10	0	10	7	4	1	11	8	5	2	12	9	6	3
11	0	11	9	7	5	3	1	12	10	8	6	4	2
12	0	12	11	10	9	8	7	6	5	4	3	2	1

Modular Arithmetic - exponentials

We also need to do exponentiation:

- $3^4 = 3 \times 3 \times 3 \times 3 = 81 \equiv_{15} 6$
- $3^4 = 81 \equiv_9 0$
- $5^6 = 5 \times 5 \times 5 \times 5 \times 5 \times 5 \equiv_9 25 \times 25 \times 25 \equiv_9 7 \times 7 \times 7$ [because $25 \equiv_9 7$] $\equiv_9 4 \times 7 \equiv_9 28 \equiv_9 1$

Here is a table of exponentials in modulo 9, for nonzero exponent numbers

	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	2	4	8	7	5	1	2	4
3	3	0	0	0	0	0	0	0
4	4	7	1	4	7	1	4	7
5	5	7	8	4	2	1	5	7
6	6	0	0	0	0	0	0	0
7	7	4	1	7	4	1	7	4
8	8	1	8	1	8	1	8	1

Modular Arithmetic - exponentiation

The algorithm to calculate b to the power of e, modulo n is quite neat:

```
set result initially = 1;
while (e != 0) {
  if (e modulo 2 != 0)
    set result = (result * b) mod n;
  set e = e / 2;
  set b = (b*b) mod n;
  }
return result;
```

Modular Arithmetic - tools

You will find a number of resources for exploring modular arithmetic at
http://computing.northumbria.ac.uk/staff/cgmb3/teaching/cryptography/index_crypto.html

In particular,

- Spreadsheets[click] from which the the multiplication and exponentiation tables above were made. Add your own sheets!
- A modular calculator[click] - run at a command-line, eg
  `java -jar ModularCalc.jar 15`
- A java application to calculate modular exponentials[click] - download and run with java.
- A java implmentation of the extended Euclidean algorithm[click] - download and run with java.

This web page has links to the java source code of these tools, for your interest.

Modular Arithmetic - finding reciprocals

Computationally, modular reciprocals are found using the Euclidean algorithm

- See the last link above.
- Given two numbers x, y this algorithm will will find their *greatest common divisor* the largest number that exactly divides into x and y.
- It also displays the GCD as a linear combination of x, y:
  - It will find r, s such that $GCD(x, y) = rx + sy$
- Eg $GCD(24, 54) = 6$; $6 = 24(-2) + 54 \times 1$
- For $x^{-1} mod\ n$ to exist, x and n have to be relatively prime: $GCD(x, n) = 1$.

Modular Arithmetic - finding reciprocals

To calculate x^{-1} mod n,

- Feed x, n into the Euclidean algorithm: get out numbers r, s such that $1 = GCD(x, n) = rx + sn$
- This tells us that $rx \equiv_n 1$: in other words, $x^{-1} mod\ n$ is r.

For example $200003^{-}1 mod\ 263160$:

- feed 200003 and 263160 into the Euclidean algorithm and we get a message that their GCD $= 1$ and
  $1 = -68573 \times 263160 + 90227 \times 200003$
- This tells us 200003^{-1} mod $263160 = 90227$

Euclidean algorithm in brief

The "classic" Euclidean algorithm:

```
To find the GCD of x and y {
repeat {
  Subtract the smaller of x, y from the bigger
  Replace the bigger of x, y by this number
}
...until 0 is obtained.
The last nonzero number is the GCD
}
```

Example: GCD(36, 102):

- $36, 102 \to 66; 36, 66 \to 30; 36, 30 \to 6; 30, 6 \to 24;$ $24, 6 \to 18; 18, 6 \to 12; 12, 6 \to 6; 6, 6 \to 0.$
- GCD $= 6$

Euclidean algorithm in brief

To recover r and s such that $GCD(x, y) = rx + sy$

- It is necessary to keep track of all the repeated subtractions
- "Unwind" them to recover the original x, y; build up the values of r, s as you go.

The Euclidean algorithm application uses integer division rather than subtraction to speed things up a bit.

x, y $\rightarrow$ quotient q, remainder r

- subtract x from y repeatedly until a number smaller than x is left
- q counts the subtractions, r is the number left
- $q = y/x$, $r = y \ \% \ x$

RSA - Rivest, Shamir, Adleman Public Key Cryptosystem

Based on a clever theorem -

- Pierre de Fermat, in the 1600s, showed that if m is any integer whatever, and p is a prime number then $m^{(p-1)} \equiv_p 1$.
- Eg $4^6 = 8^3 \equiv_7 1^3 = 1$
- Leonard Euler, in the 1700s, extended this result: If p and q are prime numbers, and $n = pq$, and m is any number relatively prime to n, then $m^{(p-1)(q-1)} \equiv_n 1$.
- It follows that for any k, $m^{k(p-1)(q-1)+1} \equiv_n m$.

To use this for cryptography,

- Our m will be a block of bits – part of a message to be encrypted
- The operation of raising m to the power of $(p-1)(q-1)+1$ (mod n) we will split into 2 parts: part 1 will be the encryption process, part 2 will be the decryption process; applying one part after the other will give back the message block we started with.

Definition of an RSA Cryptosystem

- Think of two big prime numbers – nowadays, of the order of 2^{1024}; call them p, q; Compute $n = pq$
- Pick a random number d bigger than p and q but smaller than n, such that d is relatively prime to $(p-1)(q-1)$
- Compute e, the reciprocal of d mod $(p-1)(q-1)$.
  - ed % $(p-1)(q-1) = 1$ (ie $ed \equiv_{(p-1)(q-1)} 1$)
- Publish the pair n, e – these are your public key. Keep p, q, d secret
- Encryption: from each block of bits m, compute $c = m^e$ % n
- Decryption: from each encrypted block c, compute $m = c^d$ % n

Why this works

From Eulers theorem, $m^{k(p-1)(q-1)+1} \equiv_n m$

But we made e and d so that $ed \equiv_{(p-1)(q-1)} 1$. That is, e.d = 1 + a multiple of (p-1)(q-1).

So Eulers result tells us that $m^{ed} \equiv_n m$

In other words, $(m^e)^d \equiv_n m$

So if $c \equiv_n m^e$ then $c^d \equiv_n m$

In other words, if we encrypt by computing $c = m^e \ \% \ n$, then computing $c^d \ \% \ n$ recovers the original m

Thus encryption is $c = m^e \ \% \ n$; decryption is $m = c^d \ \% \ n$.

RSA example

$2^{18} = 262144$; With an n just above this we can encrypt blocks of 18 bits. Two prime numbers p = 613, q = 431 give n = 264203.

(p-1)(q-1) = 263160

Random number in range 614 – 263159 inclusive, relatively prime to 263160: d = 200003. This is prime and not a factor of 263160: it will do.

$e = d^{-1} \bmod 263160$

- We have a program to compute this: the Euclidean algorithm computes the greatest common divisor or factor of two numbers, and expresses it as a linear combination of the two numbers.
- Feed 200003 and 263160 into the Euclidean Algorithm application and it says:
  - $1 = -68573 \times 263160 + 90227 \times 200003$;
  - GCD(263160,200003) = 1
- The second equation tells us something we knew already; the first tells us -

The reciprocal or inverse (mod 263160) of d = 200003 is e = 90227

RSA example

Check with a calculator that $200003 \times 90227 \,\%\, 263160 = 1$

We now have a completely defined RSA crytposystem which we can use to encrypt and decrypt data in blocks of 18-bits.

- The public key is (n = 264203, e = 90227)
- Each block we interpret as a number $< 2^{18} = 262144$; we can do arithmetic modulo n with such blocks.

Example: to encrypt the block 000100 000101 000100:

- 000100 000101 000100 (binary) = 16708 (decimal)
- $16708^{90227} \,\%\, 264203 = 111885$ using modular exponentiation app
- = 011011 010100 001101

Decrypting:

- $111885^{200003} \,\%\, 264203 = 16708$ = the original bits

Security of RSA

Alice sends Bob a message m using Bob's n, e which are public. The computation $c = m^e \ \% \ n$ is easy and c is sent to Bob.

Bob must use his (private) d to recover m: $m = c^d \ \% \ n$, also easy given knowledge of d.

An eavesdropper must be able to figure out d. She knows n and e, and knows that d is the reciprocal of e modulo $(p-1)(q-1)$. But she does not know p, q, although she does know $n = pq$.

Eve can figure out p, q from pq by *factorisation* if p, q, are small; but interestingly, inferring primes p, q from their product pq is HARD when p, q, are sufficiently large - around 1000 bits, for instance.

As of 2010, the largest factorized pq was 768 bits long. This took, on a grid of parallel CPUs around 1500 CPU years (2 actual years on hundreds of computers). No larger RSA key is known publicly to have been factorized.

In practice, 1024 to 4096 a considered to suffice.

Further reading

You will find a number of resources for exploring RSA at
http://computing.northumbria.ac.uk/staff/cgmb3/teaching/cryptography/index_crypto.html

In particular,

- A short paper[click] on the Euler-Fermat theorem and RSA;
- An overview[click] of the mathematical basis of RSA;
- A survey of attacks[click] on RSA;
- Another survey of attacks[click] on RSA;
- The Wikipedia articles on Integer factorization and on RSA are a good read.