Public Key Cryptography Mathematics

Introduction

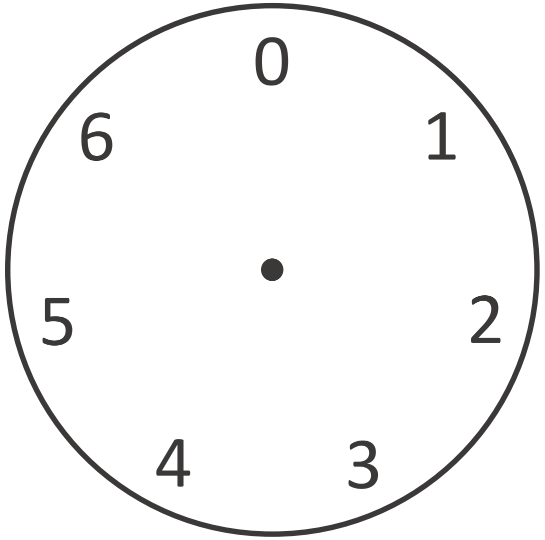
- This primer explains the mathematics behind Diffie-Hellman key exchange, and RSA public-key cryptography
- This is further reading for those who are interested, if you’ve followed the course to the end, you’ll know enough about how to use these techniques
- But, the maths is fascinating, and it doesn’t hurt to know it!

Big Numbers

- Modular arithmetic and integer factorisation drive public-key cryptography
- These examples are small, in reality numbers are hundreds of digits long
- As computer power increases, we can increase the size of these numbers to preserve the integrity of our algorithms

Modular Arithmetic

- A system of arithmetic based around cycles of numbers
  - Numbers modulo n are a finite field
- The field is finite, because whatever you do, addition, multiplication, subtraction etc., you remain within these numbers.
- If you ever go above these numbers, you wrap back around past zero



The set of numbers modulo 7

Notation

- In modular arithmetic, we often avoid the standard notation you might be familiar with from programming. E.g.:

$$9 \bmod 7 = 2$$

- In favour of writing (mod n) after the equation, signalling that everything is taken modulo that number. E.g.:

$$8+3 \equiv 2^5 \pmod{7}$$

Congruence. The two sides are identical when everything is mod 7

Equivalences

$$\big((\mathrm{a\ mod\ n}) + (b\ mod\ n)\big) mod\ n = (a + b)\ mod\ n$$

$$\big((\mathrm{a\ mod\ n}) \cdot (b\ mod\ n)\big) mod\ n = (a \cdot b)\ mod\ n$$

For addition, multiplication etc., the order you perform operations doesn’t matter, and when you calculate the modulus also doesn’t matter

Multiplication Example

Rule:

$$\big((\mathrm{a\ mod\ n}) \cdot (b\ mod\ n)\big)\ mod\ n = (a \cdot b)\ mod\ n$$

Example: $(29013 \cdot 1123)\ mod\ 7$

$$32581599\ \mathrm{mod}\ 7$$

This can be handy. Above is a seemingly difficult sum.

Multiplication Example

Rule:

$$\big((\mathrm{a \bmod n}) \cdot (b \; mod \; n)\big) \; mod \; n = (a \cdot b) \; mod \; n$$

Example: $(29013 \cdot 1123) \; mod \; 7$

$32581599 \mathrm{\ mod\ } 7$

Or: $\big((29013 \; mod \; 7) \cdot (1123 \; mod \; 7)\big) \mathrm{\ mod\ } 7$

$(5 \cdot 3) \; mod \; 7$

$15 \; mod \; 7 = \mathbf{1}$

But in fact, we can take modulus early, and simplify the expression.
Computers do this to make DH and RSA much faster

Exponentiation

Example: $13^{11}\ mod\ 7 = ?$

Suppose we are trying to calculate the above. Let's not bother working out the whole left hand side in one go

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

$$\left((13^2 \; mod \; 7) \cdot (13^9 \; mod \; 7) \right) mod \; 7$$

$$169 \; mod \; 7 = 1$$

13^{11} is actually 13^2 x 13^9

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

$$\left((13^2 \; mod \; 7) \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^2 \; mod \; 7) \cdot (13^7 \; mod \; 7)\right) mod \; 7$$

etc.

This doesn't look so challenging now. Every 13^2 mod 7 is just 1.

Exponentiation

Example: $13^{11}\ mod\ 7 = ?$

$$\left((13^2\ mod\ 7)\cdot(13^9\ mod\ 7)\right)\ mod\ 7$$

$$\left(1\cdot(13^9\ mod\ 7)\right)\ mod\ 7$$

$$\left(1\cdot(13^2\ mod\ 7)\cdot(13^7\ mod\ 7)\right)\ mod\ 7$$

etc.

$$\left(1\cdot(13^1\ mod\ 7)\right)\ mod\ 7$$

$$13\ mod\ 7 = \mathbf{6}$$

Logarithms

- A logarithm is the inverse function to exponentiation:

$$a^b = c$$

$$b = \log_a(c)$$

- This is easy to compute even for large numbers

If you're not used to logs, read this as "b is the number you have to raise a to, to get to c."

Discrete Logarithms

- When operating mod n, we call the operation a discrete logarithm:

$$a^b = c \pmod{\mathrm{n}}$$

$$b = \mathrm{dlog}_{a,n}(c)$$

Example:

$$7^2 = 4 \pmod{9}$$

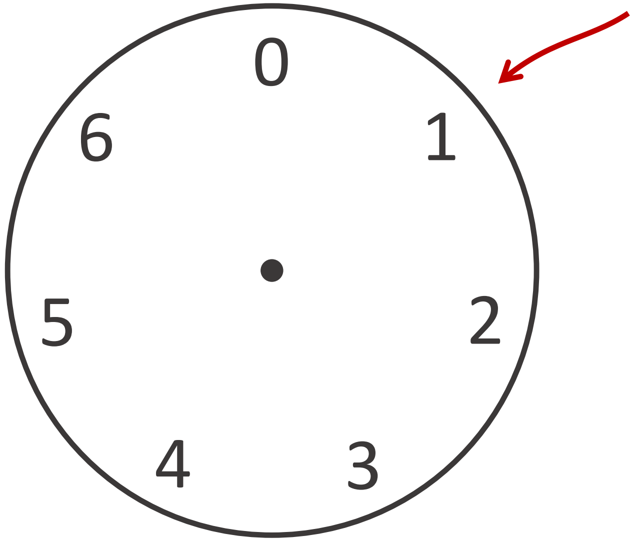
$$2 = \mathrm{dlog}_{7,9}(4)$$

Discrete Logarithms

- Discrete logs are much harder to compute

$$3^{?} = 1 \pmod 7$$

$$? = \mathrm{dlog}_{3,7}(1)$$



The set of numbers modulo 7

Intuitively, this is because the output is somewhere on this finite cycle of numbers, but where we are tells us nothing about how many times we've looped around past zero

Discrete Logarithms

- Discrete logs are much harder to compute

$$3^{?} = 1 \pmod 7$$
$$? = \mathrm{dlog}_{3,7}(1)$$

Brute force:

$$3^{1} = 3 \pmod 7$$
$$3^{2} = 2 \pmod 7$$
$$3^{3} = 6 \pmod 7$$
$$3^{4} = 4 \pmod 7$$
$$3^{5} = 5 \pmod 7$$
$$3^{6} = 1 \pmod 7$$

This leaves us having to brute force the answer. What if mod 7 was mod some 2000 bit number?

This exponentiation, modulo some prime, is the hard to reverse "mixing" that we talked about in the class.

Primitive Roots

- The number that is raised to a certain power, is called the generator *g*

***g* = 9**

$$9^1 = 2 \pmod 7$$
$$9^2 = 4 \pmod 7$$
$$9^3 = 1 \pmod 7$$
$$9^4 = 2 \pmod 7$$
$$9^5 = 4 \pmod 7$$
$$9^6 = 1 \pmod 7$$

***g* = 3**

$$3^1 = 3 \pmod 7$$
$$3^2 = 2 \pmod 7$$
$$3^3 = 6 \pmod 7$$
$$3^4 = 4 \pmod 7$$
$$3^5 = 5 \pmod 7$$
$$3^6 = 1 \pmod 7$$

3 is a primitive root. Primitive roots are better, notice how the power is harder to guess based on the output, because 3 generates all possible outputs 0-6.

Diffie-Hellman

1. Alice and Bob agree on a large prime p, and a generator g that is a primitive root of p
2. Alice chooses a private value a at random, then sends Bob a public g^a mod p
3. Bob chooses a private value b at random, then sends Alice a public g^b mod p
4. Alice computes $(g^b)^a$ mod p, which is actually $g^{ab} \bmod p$
5. Bob computes $(g^a)^b$ mod p, which is actually $g^{ab} \bmod p$

The order of exponentiation doesn't matter.

Public, Private

Example

Alice and Bob agree on $g = 3$ and $p = 29$

Alice chooses $a = 23$,
then $g^a = 3^{23} \bmod 29 = 8$

1. Bob chooses $b = 12$,
then $g^b = 3^{12} \bmod 29 = 16$

g^a g^b

Alice calculates:
$(g^b)^a \bmod 29 =$
$16^{23} \bmod 29 =$ **24**

Bob calculates:
$(g^a)^b \bmod 29 =$
$8^{12} \bmod 29 =$ **24**

Public, Private

Why is DH KEX Secure?

- The secret shared key is g^{ab}
- Yet, only g, p, g^a and g^b have been transmitted and are public
- The only way to calculate g^{ab} is either $(g^a)^b$ or $(g^b)^a$
- The only way to find *a* or *b* is solve:

$$a = \log_{g,p}(g^b)$$

$$b = \log_{g,p}(g^a)$$

RSA

- RSA also uses modulo arithmetic, but in some sense it is secondary to it’s primary feature, which is integer factorisation

- Remember that the point of RSA is to produce two keys, in this case numbers, that will reverse each other when used in encryption or decryption

Integer Factorisation

- Any integer can be expressed as the multiplication of a list of prime numbers:

Example: 103284720

= 2 x 2 x 2 x 2 x 3 x 3 x 5
x 7 x 9 x 9 x 11 x 23

- The longer the value, the harder (and slower) this gets

Integer Factorisation

- Semi-primes are the hardest numbers to factor:
  - Product of two primes, $n = pq$

$$\text{Example: } n = \begin{array}{l} 1522605027922533360535618378 \\ 1326374297180681149613806886 \\ 5790849458012296325895289765 \\ 400035069200613 9 \end{array}$$

What are p and q?

You can't whittle this down, you either guess p or q, or you don't factorise n. This is pure brute force.

Integer Factorisation

- Semi-primes are the hardest numbers to factor:
  - Product of two primes, $n = pq$

Example: $n =$ 1522605027922533360535618378
1326374297180681149613806886
5790849458012296325895289765
4000350692006139

What are p and q?

p = 37975227936943673922808872755445627854565536638199
q = 40094690950920881030683735292761468389214899724061

Since you asked!

Euler Totient Function

- Integers a and b are relatively prime if they do not share a divisor (except 1)
- The Euler totient Φ is the integers from 1 to n-1 that are relatively prime with n

  1. What is $\Phi(9)$?
  2. What about $\Phi(11)$?

Why am I telling you this? Well the Euler totient will be used in the key generation of RSA to ensure a mathematical link between the public and private keys

Answer: 6 and 10. Note that the totient of a prime is always the prime - 1.

Euler Totient Function

- The totient value of a prime p is simply p-1
- For two primes multiplied together it's

  (p-1)(q-1)

This will come in useful!

Multiplicative Inverses

- Think of multiplicative inverses as the opposites of one another, when used in multiplication. The common example would be:

$$x \qquad\qquad \frac{1}{x}$$

If you multiply by either of these, you can reverse the process by multiplying by the other. The same is true of division.

RSA – Key Generation

1. Choose two large primes, *p and q*, then calculate *n = pq*
2. Select a value *e* that is relatively prime with the totient of *n*.
   - (Remember, we know $\Phi(n)$ as (p-1)(q-1))

Example:

$$p = 17, q = 11$$
$$n = p \cdot q = 187$$
$$\Phi(n) \ = 160$$
$$e = one\ of\ 3, 6, 7, 11 \ldots\ = 7$$

I've chosen 7 here at random, this is often 3 or 65537, which are relatively prime with most things!

Public, Private

RSA – Key Generation

3. Calculate a multiplicative inverse to *e*, *d*, where:

$$e \equiv d^{-1}\ (mod\ \Phi(n))$$

Or: $(e \cdot d)\ mod\ \Phi(n) = 1$

4. This is easily achieved if we know $\Phi(n)$, but not otherwise, using the extended Euclidean algorithm (not shown here)

Example: $e = 7, d = 23$

$$(7 \cdot 23)\ mod\ 160 = 1$$

Public, Private

RSA – Encryption

- Now we have a public key *e,n* and a private key *d*
- Encryption is performed by:

$$M^e = C \ (mod\ n)$$
$$C^d = M \ (mod\ n)$$

Example:

$$M = 74$$
$$C = 74^7\ mod\ 187 = 167$$
$$M' = 167^{23}\ mod\ 187 = ?$$

Public, Private

RSA – Encryption

- Now we have a public key *e*,*n* and a private key *d*
- Encryption is performed by:

$$M^e = C \ (mod\ n)$$
$$C^d = M \ (mod\ n)$$

Example:

$$M = 74$$
$$C = 74^7\ mod\ 187 = 167$$
$$M' = 167^{23}\ mod\ 187 = 74$$

Public, Private

Why is RSA Secure

- We'd like the message M based on some ciphertext C, given the public key e:

$$C = ?^e \ (mod\ n)$$

Equivalent to: $M = C^? \ (mod\ n)$

- Calculating d can only be achieved by knowing the totient Φ of n. Finding this is extremely hard, for example we could factor n into p and q

Why does RSA work?

- The RSA proof is complex, I will put it in the following slides, but understanding of this is not necessary to use RSA safely. Consider this a warning!
- Put simply, because they are multiplicative inverses modulo the totient of n, e and d become inverses when used in exponentiation modulo n
- The key to it all, euler's theorem:

$$a^{\Phi(n)} = 1 \ (mod\ n)$$

Raising any number to the totient of n, gives 1 when taken modulo n

RSA Proof

$$M^{ed} = ?\ (mod\ n)$$

We want to show that $M^{ed} = M$ (mod n), in other words, that if you "encrypt" with e, and then again with d, the process is reversed

RSA Proof

$$M^{ed} = ?\ (mod\ n)$$

Recall that: $(e \cdot d)\ mod\ \Phi(n) = 1$

$$\therefore\ (e \cdot d) = \mathrm{k} \cdot \Phi(n) + 1$$

Since we know that ed (mod $\Phi(n)$) is 1, we know that ed is some multiple (k) of $\Phi(n)$, + another 1.

RSA Proof

$$M^{ed} = ?\ (mod\ n)$$

Recall that: $(e \cdot d)\ mod\ \Phi(n) = 1$

$$\therefore\ (e \cdot d) = \mathrm{k} \cdot \Phi(n) + 1$$

So: $M^{ed} = M^{\mathrm{k}\cdot\Phi(n)+1} = M \cdot M^{\mathrm{k}\cdot\Phi(n)}$

Putting what we know into the formula, we get one M, multiplied by $M^{\mathrm{k}\cdot\Phi(n)}$

RSA Proof

$$M^{ed} = ?\ (mod\ n)$$

Recall that: $(e \cdot d)\ mod\ \Phi(n) = 1$

$$\therefore\ (e \cdot d) = \mathrm{k} \cdot \Phi(n) + 1$$

So: $M^{ed} = M^{\mathrm{k}\cdot\Phi(n)+1} = M \cdot M^{\mathrm{k}\cdot\Phi(n)}$

$$= M \cdot \left(M^{\Phi(n)}\right)^{k}$$

We can extract out k. Remember a^{bc} is just $(a^b)^c$

RSA Proof

$$M^{ed} = ?\ (mod\ n)$$

Recall that: $(e \cdot d)\ mod\ \Phi(n) = 1$

$$\therefore\ (e \cdot d) = \mathrm{k} \cdot \Phi(n) + 1$$

So: $M^{ed} = M^{\mathrm{k}\cdot\Phi(n)+1} = M \cdot M^{\mathrm{k}\cdot\Phi(n)}$

$$= M \cdot \left(M^{\Phi(n)}\right)^{k} = M \cdot 1^{k}$$

$$= M$$

Euler's Theorem:
$a^{\Phi(n)} = 1\ (mod\ n)$

Finally, we can use euler's theorem to reduce that term to 1, and we're done!