Contents

1 Public key cryptography

- Two keys, a public key and a private key
- Public-key (asymmetric) cryptography hinges upon the premise that: **It is computationally infeasible to calculate a private from a public key**
- Public-key cryptography gains us a few important abilities:
  - We can exchange a private symmetric key in the open
  - We can verify the sender of a message
  - Non-repudiation you cant deny you did something

Symmetric	Asymmetric
One Key	Two Keys
Keys are usually 128 or 256-bits	Keys are much longer, 2048 or 4096-bits
Usually extremely fast	Computationally slower
Longer term communication	Key exchange, verification and authentication only
Based on circuits of permutation and substitution	Based entirely on mathematical principles

2 Big numbers

- Modular **arithmetic** and **integer factorisation** drive public-key cryptography
- As computer power increases, we can increase the size of these numbers to preserve the integrity of our algorithms

3 Modular Arithmetic

A system of arithmetic based around **cycles of numbers**. Numbers modulo n are a *finite field*. Can think of it as a circle, because the set of numbers is limited and goes in a loop

3.1 The Congruence Relation

For a positive integer n, two numbers a and b are said to be *congruent modulo* n, if their difference $a - b$ is an integer **multiple** of n (that is, if there is an integer k such that $ab = k * n$). This congruence relation is typically considered when a and b are integers, and is denoted

$$a \equiv b \pmod{n} \tag{1}$$

$$a \pmod{n} = b \pmod{n} \tag{2}$$

When you apply modulo makes no difference

$$((a \bmod n) + (b \bmod n)) \bmod n = (a + b) \bmod n \tag{3}$$

$$((a \bmod n) * (b \bmod n)) \bmod n = (a * b) \bmod n \tag{4}$$

3.2 Logarithms

A logarithm is the inverse function to exponentiation

$$a^b = c \tag{5}$$
$$b = log_a c \tag{6}$$

When operating $\mod n$, we call the operation a discrete logarithm. Discrete logs are much harder to compute. The number that is raised to a certain power, is called the **generator g**

$$a^b = c \text{ (mod n)}$$
$$b = \text{dlog}_{a,n}(c)$$

$$7^2 = 4 \text{ (mod 9)}$$
$$2 = \text{dlog}_{7,9}(4)$$

4 Diffie-Hellman

- Two parties can jointly agree a *shared secret* over an insecure channel
- Mathematically, what we are doing is both calculating the same value, mod a prime p
- Look at notes:

4.1 Why is DH KEX Secure?

- The secret shared key is gab
- 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:

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

4.2 Vulnerabilities

Man-in-the-middle: A third party could intercept the initial communication from Alice, then create two separate key exchanges with both Alice and Bob. **An asymmetric protocol is required to prevent this**

4.3 Perfect Forward Secrecy

- Theres always a chance a DHKEX key might be broken
- If we establish a symmetric key, how long should we use it for?
- Perfect forward secrecy means we **generate new keys for each session**, rather than persistent keys

4.4 Ephemeral Mode

- In protocols like TLS, running Diffie-Hellman in ephemeral mode forces a new key exchange every time
- The recommended settings for TLS are now 2048-bit DH keys, in ephemeral mode

5 Elliptic Curve Cryptography

Elliptic curves, of the form $y^2 = x^3 + ax + b$ can be used in place of mod arithmetic in DHKEX Elliptic curves are much stronger than traditional public-key schemes for the **same key length**

Reference section

finite field

A finite field is a set of numbers in which we can add, subtract, multiply and divide, and stay within that set