public_key

March 15, 2020

0.0.1 Public Key Cryptography

- Review of symmetric key
- Key differences (so to speak)
- Hard problems
- Security Levels
- Quantum Computing

0.0.2 Review of Symmetric Key Crypto

In DES and AES (and any symmetric key cryptosystem) there is only one secret: the key.

The same key can be used to encrypt as well as decrypt messages.

The encryption process and decryption process are either similar or identical.

But there are some shortcomings…

0.0.3 Shortcomings of Symmetric Key Cryptography

Key Distribution The biggest problem with symmetric key cryptography is the key distribution problem.

How do Alice and Bob agree on a key?

They could meet and exchange a key, or use a trusted carrier.

But these are not very convenient.

0.0.4 Shortcomings of Symmetric Key Cryptography

Key Distribution It would be better if they could exchange a key online.

But how is this possible if the key is exactly what they need to share information securely?

This seems like a vicious circle.

0.0.5 Shortcomings of Symmetric Key Cryptograpy

Number of keys Another problem is that each pair of users needs to have a shared key.

There are $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs of users.

For $n = 10000$ users there need to be 49995000 keys.

All of these must be protected, regularly refreshed, etc.

0.0.6 Shortcomings of Symmetric Key Cryptography

Non-repudiability If Alice and Bob share exactly they same key, then you can't use that key to uniquely identify Alice or Bob.

There's no way to tell whether it was Alice or Bob that sent a given message.

It the message is a financial payment you can't tell who paid whom, or whether the whole thing was forged.

0.0.7 Public Key Cryptography

There is a simple idea that solves all of these problems:

Have two keys, one for encryption and one for decryption.

There is no reason to keep the encryption key secret (so we call it the *public key*).

However the decryption key must be kept secret (it is the *private key*).

0.0.8 Public Key Cryptography

A metaphor…

The mail slot: Your house may have a mail slot: anyone can send a message to you, but only you can unlock the door and read the messages.

0.0.9 Public Key Cryptography

How does the public key idea solve the problems we mentioned?

1. Key distribution
2. Number of keys
3. Non-repudiability?

0.0.10 Key distribution

Alice and Bob want to exchange a key for AES.

Alice generates the key k using a TRNG.

She encrypts the key using Bob's encryption key : $c = E_{Bob_{pub}}(k)$.

She then sends Bob c, which he decrypts using his private key: $k = D_{Bob_{priv}}(c)$

(This is how HTTPS works)

Why do they even need AES? Because it's much *much* faster than the public key algorithm.

0.0.11 Number of keys

We no longer need a key for each pair of people.

We now only need two keys per person: a public key and a private key.

For $n = 10000$ users there are 20000 keys (not 49995000 keys).

As a bonus, there is no reason to keep the public key secret. You can put it in your public profile.

(Being sure that you have Bob's public key and not Eve's is still a problem though.)

0.0.12 Non-repudiability

Each user has a unique private key. Knowledge of the private key is proof of who you are.

You can even prove you know the key without revealing it:

Just "decrypt" some challenge statement c:

$s = D_{Bob_{priv}}(c)$.

Then anyone can "encrypt" s using the public key and see that the result is c:

$c \stackrel{?}{=} E_{Bob_{pub}}(s)$

Something very similar to this happens when you spend bitcoin.

0.0.13 History of Pub Key

Ralph Merkle had the idea for Pub Key when he was an undergraduate at UC Berkeley in 1974.

story

His idea was rejected by his professor.

He then wrote the idea up as a paper and submitted it to a journal.

It was rejected as a crazy idea.

Another rejection.

0.0.14 History of Pub Key

Eventually in 1975 two established cryptographers had similar ideas: Diffie and Hellman.

They published a famous paper called "New Directions in Cryptography" which became a classic work.

They solved the key exchange problem (we still use their solution).

Here is a deeper history:

http://cr.yp.to/bib/1988/diffie.pdf

Secret branches of the US and British government claimed later to have known about pub key methods.

But it is doubtful that they grasped the implications.

0.0.15 One-way functions:

A function $f(x)$ is a **one-way** function if:

1. $y = f(x)$ is computationally easy, and
2. $x = f^{-1}(y)$ is computationally hard.

We can't prove these exist (it depends on $P \neq NP$).

But they probably do and there are a few known candidates.

0.0.16 Example of candidate one-way functions

Factoring Given large primes p and q, computing $N = pq$ is easy.

But going from N back to the factors p and q is hard.

There is no known polynomial time algorithm for factoring.

(But there is a fast quantum algorithm)

0.0.17 Example of candidate one-way functions

The Rabin Problem Given N as in the previous slide, let $0 < x < N$.

Then computing $y = x^2 \mod N$ is easy.

But computing x from y is hard.

0.0.18 Example of candidate one-way functions

The Discrete Log Problem Given a big prime p, let $0 < x < p$. Assume p and x are public.

Let d be a big number which is kept private.

Then computing $x^d \mod p$ is easy.

But recovering d from $x^d \mod p$ and x is hard.

(We want to take the discrete logarithm of x^d base x.)

0.0.19 Key Lengths and Security Levels

In symmetric key cryptography the key is a random number, secretly generated.

In public key cryptography, the decryption key is the solution to a simple mathematical equation that everyone can see.

Whereas in symmetric key you might need to try every key to break a cipher, in public key you only need to solve the equation.

Solving the equation might be easier than trying every key (and often is).

For that reason public keys tend to be much longer than private keys.

0.0.20 Key Lengths and Security Levels

Table 6.1 Bit lengths of public-key algorithms for different security levels

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

0.0.21 Quantum Computing...

According to the NSA

"A sufficiently large quantum computer, if built, would be capable of undermining all widely-deployed public key algorithms used for key establishment and digital signatures.

... It is generally accepted that quantum computing techniques are much less effective against symmetric algorithms than against current widely used public key algorithms.

While public key cryptography requires changes in the fundamental design to protect against a potential future quantum computer, symmetric key algorithms are believed to be secure provided a sufficiently large key size is used. ...

In the longer term, NSA looks to NIST to identify a broadly accepted, standardized suite of commercial public key algorithms that are not vulnerable to quantum attacks. ”

[0]: