Asymmetic Keys

Two keys one public and one private.

- Kerckhoffs's Principle
- □□: Public Key, □□ Private Key
- □□
- Encrypt: □□□□
- Decrypt: □□ □□

Kerckhoffs's Principle

Kerckhoffs's principle (also called **Kerckhoffs's desideratum**, **assumption**, **axiom**, **doctrine** or **law**) of cryptography was stated by the Dutch cryptographer Auguste Kerckhoffs in the 19th century. The principle holds that a cryptosystem should be secure, even if everything about the system, except the key, is public knowledge. This concept is widely embraced by cryptographers, in contrast to security through obscurity, which is not.

Kerckhoffs's principle was phrased by the American mathematician Claude Shannon as "the enemy knows the system",[1] i.e., "one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them". In that form, it is called **Shannon's maxim**.

Security Through Obscurity Considered Dangerous

Hiding security vulnerabilities in algorithms, software, and/or hardware decreases the likelihood they will be repaired and increases the likelihood that they can and will be exploited. Discouraging or outlawing discussion of weaknesses and vulnerabilities is extremely dangerous and deleterious to the security of computer systems, the network, and its citizens. [2]

Open Discussion Encourages Better Security

The long history of cryptography and cryptoanalysis has shown time and time again that open discussion and analysis of algorithms exposes weaknesses not thought of by the original authors, and thereby leads to better and more secure algorithms. As Kerckhoffs noted about cipher systems in 1883, "Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi." (Roughly, "the system must not require secrecy and must be able to be stolen by the enemy without causing trouble.") [2]

[1] https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

[2] https://www.cs.columbia.edu/~smb/papers/draft-ymbk-obscurity-00.txt

□□: Public Key, □□ Private Key

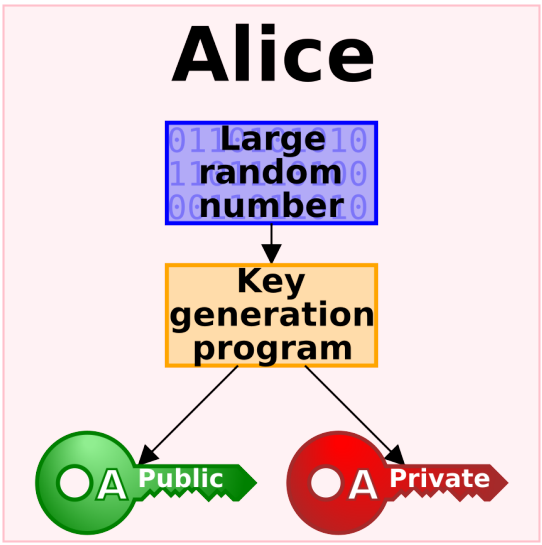
Public-key cryptography, or asymmetric cryptography,

An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

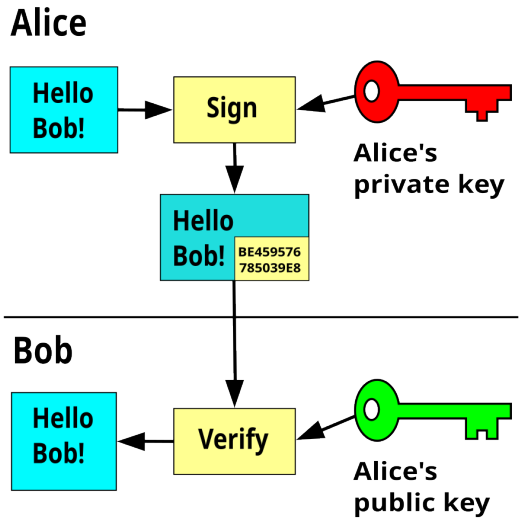
Free Form Programmable Keys

□□: Public Key

□□ Private Key



In this example the message is digitally signed with Alice's private key, but the message itself is not encrypted. 1) Alice signs a message with her private key. 2) Using Alice's public key, Bob can verify that Alice sent the message and that the message has not been modified.



[1] https://en.wikipedia.org/wiki/Public-key_cryptography

□□

□□ is a mighty operator, function and a cloud data source all at once!

In the ff script below you can see on the first line □□ acts as singleton operator or function and , and on the 3rd line as a cloud data source:

```
□□;

tmp = "□□/curve";

show tmp;

save as 2keys;
```

Output:

"tmp" → "ethereum"

Cloud Object Browser

https://www.wolframcloud.com/obj/ccn2/AI/freeform/blockchain/publishedtest/certs_free_2keys

Encrypt: □□□□

- Symbol □□behaves like a binary operator
- □□ encrypts what is on its rhs e.g. □□m
- □□□□uses the key on its lhs to encrypt
- crypt = □□□□ m just like e.g. multiplication copies the resultant into the lhs of =

> □□□□ m behaves like a product by the operator □□incorporating is rhs and lhs into a product.

```
□□["rsa"] ;

m = "hi";

crypt = □□□□m;

show crypt;

save as rsa;
```

Output:

"crypted" ⟶ "<|Cipher -> RSA, Data -> ByteArray[ETlKQciE9Qxm3/+Cm1y8/RdCwREyqc84Od8IrJYGcjlm4RGXxs4oOXl2xEqbsdo1bcUk2C4HkhxeGWInbVPU1ipokp1P7zg3ffUWERID4/NA1waVzbeeFCq4rs6BjXB2/D/QGFYkNEPxkY9NA8OijjeoTWbfjs+DY+nZft585kzDiK8yPF6ZchkcZeLIAJwLi2xiZzGt4Ko1oOypFSyEYoOKljWnm7Iudr4cC/ec/9ap1psUkdDybcMHsFHIUO3OCU0PGIHS9FKR2nCPVTEZZunTLHzXCrHwWf+OZCoFt2qurFpLzD52s3C30hozalgVz5hgn28c9l9DiGmpuc7T0A==], OriginalForm -> String, Padding -> PKCS1|>"

Cloud Object Browser

https://www.wolframcloud.com/obj/ccn2/AI/freeform/blockchain/publishedtest/certs_free_rsa

Decrypt: □□ □□

```
□□["rsa"] ;

bob = "hi";

crypt  = □□□□ob;

alice = crypt □□□□;

show bob also alice;

save as rsa;
```

Output:

"bob" ⟶ "(\"hi\")"

"alice" ⟶ "(\"hi\")"