Cryptography

Cryptography spans a significant range of topics but the core ideas are often represented by the CIA triad:

- Confidentiality
- Integrity
- Availability

The **Non-repudiation** and **Authenticity** are often added as extensions to the triad.

We won't cover most of this but it is helpful to have a base level of understanding of some concepts from crypto. To start we'll cover **Encryption/Decryption** and **Sign/Verify**.

Encryption

There are 2 general types of encryption **Symmetric encryption** and **Asymmetric (Public Key) encryption**.

Symmetric Encryption

Symmetric encryption uses a shared key. That is both parties can transmit a message securely if they both have the same key. Symmetric encryption satisfies the **confidentiality** component of the CIA triad.

So Alice can send Bob an secure message by encrypting it with their shared Key and Bob can decrypt the message by using that same Key. Even if someone intercepts the encrypted message, it will be useless to them unless they have the Key.

Notation

$K := \texttt{The shared key}$
$P := \texttt{Plaintext (unencrypted message)}$
$E() := \texttt{The encryption function}$
$D() := \texttt{The decryption function}$
$C := \texttt{Ciphertext (encrypted message)}$

Although we use the words `plaintext` and `ciphertext` the data isn't actually text. It's all binary.

Basic Usage

So Alice can send a message `P` to Bob like so:
$E(K, P) \to C$
She then transmits the encrypted message `C` to Bob.

Bob then decrypts it using his copy of the key.
$D(K, C) \rightarrow P$
And now he can read the message.

Limitations

However, there is a problem. How did Alice and Bob share the key in the first place? They don't have a secure communication channel, *if they did they wouldn't bother with encrypting their messages*, so they have no way of securely sharing the key.

Historically encryption was only used by militaries and keys were distributed in person.

However, every-time we connect to a server (using HTTPS) we need to establish a secure line of communication. This happens millions of times a day so its impractical to pre-distribute keys to everyone who might possibly connect.

Key exchange

Key exchange is how we can securely transmit a key to another person. There are several methods for securely exchanging keys. For now we'll only cover **public key cryptography**.

Asymmetric (public key) encryption

Public key crypto use **2 keys** a:

- public key
- private key

Everyone has their own `public` and `private` keys. They are generated as a pair.

Asymmetric encryption satisfies **confidentially**, **authenticity**, **integrity** and **non-repudiation**.

Private key

This is kept private. Only you should ever know your private key. If it ever gets disclosed for even a moment, it's burned and you should generate a new key pair.

Public key

This is public. Even if you don't share it with anyone, you should assume every body on earth knows it. That's how the algorithm is meant to work; this key is meant to be publicly shared.

Notation

$K_{pu}^{A} := \texttt{Alice's public key}$
$K_{pr}^{A} := \texttt{Alice's private key}$

$K_{pu}^{B} := \texttt{Bob's public key}$
$K_{pr}^{B} := \texttt{Bob's private key}$

$P := \texttt{Plaintext (unencrypted message)}$
$C := \texttt{Ciphertext (encrypted message)}$

$E() := \texttt{The encryption function}$
$D() := \texttt{The decryption function}$

$S() := \texttt{The sign function}$
$V() := \texttt{The verify function}$

Basic Usage

Public key encryption not only allows us to transmit messages securely, it also allows use to digitally sign messages. This ensures that someone cannot masquerade as another person.

Sign & Verify

Alice encrypts a message using her private key.
$E(K_{pr}^{A}, P) \rightarrow C$

The only way to decrypt it is to use her public key.
$D(K_{pu}^{A}, C) \rightarrow P$

If you can successfully decrypt a message using Alice's public key, you can be sure that Alice was actually the person who sent it since she is the only one that knows her private key. This is why you must key your private key secure. If anyone gets it, for all intents and purposes **they are you**.

Encrypting a message with your private key is called **signing** and decrypting the cipher text with your public key is called **verifying**.

The notation used above can be re-written as:
$S(K_{pr}^{A}, P) \rightarrow C$

and

$V(K_{pu}^{A}, C) \rightarrow P$

This just makes it clear you're using public key encryption for digital signatures rather than for more traditional encryption.

This scheme allows anyone to verify messages actually come from the person they claim to be coming from. However, it does not secure the message since anyone can decrypt it using Alice's public key.

Encryption & Decryption

Say Alice wants to send Bob a secure message using public key crypto.

Alice can encrypt the message using Bob's public key.
$E(K^{B}_{pu}, P) \rightarrow C$

The only person who can decrypt the message since only his private key can be used.

$D(K^{B}_{pr}, C) \rightarrow P$

Remember:

> Whichever key you use to encrypt, you must use the opposite to decrypt.

Now Alice can safely send Bob messages. However Bob can't be sure the message actually came from Alice since **everybody** has his public key.

Exchanging the symmetric key

So Alice can chain encrypting a message and signing it to send Bob a symmetric key (or any message really).

Sending the key

1. First she generates a random Key (this will be the shared key for symmetric encryption)
    - producing K
2. She encrypts that key using Bob's public key (ensuring only he can read the key)
    - producing K' (encrypted key)
3. She signs the encrypted key using her private key
    - producing $K^{'}_{signed}$

Receiving the key

Now Bob gets the message and he:

1. Verifies it came from Alice by using her public key
    - producing K'
2. Decrypts the encrypted key using his private key

- producing K

Now Alice and Bob can switch to using symmetric encryption by using the shared key K.

Limitations

Why bother going through all the trouble of sharing the symmetric key? Why not just keep using asymmetric encryption instead of only using it to share the key?

There are 2 reasons:

First, asymmetric encryption is significantly slower and less efficient than symmetric encryption. So you just use it to share a key so you can switch to the faster and more efficient symmetric encryption scheme.

Second, for SSH we use RSA 2048 for our keys. This produces 2048 bit numbers (massive 2KB numbers). RSA can only encrypt messages as large as its key size which places an upper limit of 2KB of data. Whereas AES' (the most widely used symmetric encryption algorithm) is practically unlimited (technically somewhere on the order of hundreds of millions of terabytes.)