**Encryption**

Turns readable text into unreadable text (ciphertext)

Decryption is turning cypher text back to readable text

**Symmetric Encryption (AES)**

Algorithm

* Public

Key

* Private

These together will create a ciphertext

**Symmetric means it uses 1 key**

**256-bit vs 128-bit**

* The higher the bit rate, the more secure it is but the longer it’ll take to encrypt and decrypt
* Like a door with many locks, secure but takes ages to open

**Brute force**

* This is when you try to guess the key by going through all the possible combinations

**Dictionary attack**

* Try all the words in the dictionary against the key
* Much faster
* But if the key isn’t in the dictionary, then the attack will fail

**Human psychology**

* Monkey is in the top 10 used words in passwords
* Numbers are usually added to the end of passwords
* Monkey + every number combination could be used to try and crack the key

**Symmetric Encryption Algorithms**

* Data encryption standard (DES)
* Triple-DES (3DES)
* Blowfish
* RC4/5/6
* Advanced Encryption Standard (AES)

Symmetric algorithms are used in most encryption tools, fast and currently “unbreakable” – common standard for HTTPS, Tor, SSH etc.

* Always use 256-bit where possible
* Avoid DES and RC where possible as well

**Asymmetric Encryption**

These use 2 keys, a public and private key

**Common asymmetric key algorithms**

* Rivest-Shamir-Adleman (RSA)
* Elliptical curve cryptosystem (ECC)
* Diffie-Hellman (DH)
  + Forward secrecy
* El Gamal

If you encrypt with a private, then you need to decrypt with the public key and vice versa

* Can’t ever encrypt and decrypt with the same key

**Encrypting with senders private key 🡪 Authentication**

* The receiver can confirm that it is you sending the message
* Can be decrypted with a copy of the **sender’s public key**
* Anyone can read the message as key is public

**Encrypting with receiver’s public key 🡪 Confidentiality**

* This ensures that only the holder of the **receiver’s private key** can decrypt the message I.e., the receiver
* **No authentication of sender**, could be from malicious source

When various crypto techniques are used in combination, this is called a **crypto system**

* These can provide a number of security services
  + Confidentiality
  + Authentication
  + Non-repudiation
  + Integrity

**Advantages and Disadvantages of each**

**Asymmetric**

* Better key distribution
  + Bob can place his pub key on a site and anyone can send him encrypted data that only he can read with his private key
  + With a symmetric system, sending an encrypted file to 10 people would require the password to also be sent to those 10 people – NOT SCALEABLE
* Authentication and non-repudiation
* V slow compared to symmetric
* Mathematically intensive (high bit rate)

**Symmetric**

* Fast
* Strong

Hybrid systems can be used to get the best of both worlds

**Hash Functions**

In order to agree keys with the sender and send them to him, **we need to authenticate him first to securely send those keys**

* A MITM could send us a public key pretending to be bob’s
* Can’t just take a public key and assume it’s the real key, must be authenticated

A hash function takes data of any size and concerts it, via a ‘hash function’, into a fixed size string of characters called a **Digest**

* Unconvertable from digest back to input
* No keys required, irreversible
* Provides integrity and unintentional modifications
  + No confidentiality
  + No authentication
  + No intentional modifications

**Hash algorithm 🡪 Hash value 🡪 Encrypt w/ Sender private key = Signed message**

**Types of Hash Methods** 🡪 MD2/4/5, HAVAL, SHA-1/256/384/512, TIGER

This is a way of verifying that what you have downloaded has integrity

But how do you check that the website itself that you’re downloading from isn’t compromised?

* We can’t use hash here as it doesn’t detect **intentional modification**
* This is where certificates and digital signatures are used

**Digital Signature**

A **Digital Signature** is aHashvalue **(Digest)** that is encrypted with the **Sender’s private key**

* Authentication
* Non-repudiation
* Integrity

Manual verification of a certificate must be done by reversing the same process

* You will take the signed message
* Decrypt with sender’s public key to get the hash
* The you take the file and hash it yourself
* Compare hashes and confirm that the file has maintained its integrity

**Device Guard**

Windows 10 technology

Will only allow certain types of signed files to be run

In theory, malware or RATs or Trojans cannot be run because they’re not signed

**Adding a digital signature to a hash value will ensure that the file being sent now has**

* Integrity
* Confidentiality
* Non-repudiation
* Authentication