

## Takeaways:

- Don't invent/implement your own cryptosystem.
  - Encryption in motion: SSL/TLS, SSH, IPSec.
  - Encryption at rest: PGP/GnuPG.
  - Do not use AES, RSA, etc. algorithms directly.
  - Avoid and delegate using crypto if possible.
  - Exceptions do exist, but hire a crypto expert before you pass Go.
  - Popular programs/libraries often get it wrong. Spot check their crypto before use.
- 

## Types of Encryption:

	<u>Symmetric (secret key)</u>	<u>Asymmetric (public key)</u>
Typical key sizes:	Same key encrypts and decrypts 128, 192, 256 bits	Different keys encrypt and decrypt 2048, 3072, 4096 bits (256-512 for elliptic curve)
Speed:	Fast! Fast! Fast!	Sooooooooooooow
Algorithms:	block: AES, 3-DES, Blowfish, Serpent stream: RC4, Salsa20	RSA, DSA, ElGamal, Elliptic Curve Algorithms
Typical uses:	data encryption, checksums, pseudo-random number generators, components of cryptographic hash functions	meta data encryption, authentication, verification, digital signatures, symmetric key encryption key exchange

---

## Basic Modes of Symmetric Block Encryption:

### ECB: Electronic Code Book

encrypt(plaintext1, key) → ciphertext1  
 encrypt(plaintext2, key) → ciphertext2

decrypt(ciphertext1, key) → plaintext1  
 decrypt(ciphertext2, key) → plaintext2

CBC: Cipher Block Chaining    Why? — Each ciphertext block is unique, even if plaintext blocks are the same.  
 encrypt(plaintext1 XOR init\_vector, key) → ciphertext1  
 encrypt(plaintext2 XOR ciphertext1, key) → ciphertext2  
 encrypt(plaintext3 XOR ciphertext2, key) → ciphertext3

decrypt(ciphertext1, key) XOR init\_vector → plaintext1  
 decrypt(ciphertext2, key) XOR ciphertext1 → plaintext2  
 decrypt(ciphertext3, key) XOR ciphertext2 → plaintext3

### CFB/OFB: Cipher/Output Feedback

keystream1 = encrypt(init\_vector, key)  
 keystream2 = encrypt(ciphertext1/keystream1, key)  
 keystream3 = encrypt(ciphertext2/keystream2, key)

Why? — Turns a block cipher into a stream cipher.

keystream1 XOR plaintext1 → ciphertext1  
 keystream2 XOR plaintext2 → ciphertext2  
 keystream3 XOR plaintext3 → ciphertext3

keystream1 XOR ciphertext1 → plaintext1  
 keystream2 XOR ciphertext2 → plaintext2  
 keystream3 XOR ciphertext3 → plaintext3

### CTR: Counter

keystream1 = encrypt(nonce, key)  
 keystream2 = encrypt(nonce + 1, key)  
 keystream3 = encrypt(nonce + 2, key)

Why? — Allows fast random access to any part of the key stream.

keystream1 XOR plaintext1 → ciphertext1  
 keystream2 XOR plaintext2 → ciphertext2  
 keystream3 XOR plaintext3 → ciphertext3

keystream1 XOR ciphertext1 → plaintext1  
 keystream2 XOR ciphertext2 → plaintext2  
 keystream3 XOR ciphertext3 → plaintext3

## Authenticated Encryption Modes of Symmetric Block Ciphers (AEAD):

**GCM:** Galois Counter Mode    fastest free AEAD mode, becoming part of standards, complicated implementation  
**OFB:** Offset Feedback Mode    fastest AEAD mode, simple implementation, patented but royalty-free for GPL and  
 non-commercial/non-government use  
**EAX:** not an acronym    two-pass (slow) AEAD mode, simple implementation  
**CCM:** Counter with CBC-MAC    two-pass (slow) AEAD mode, simple implementation, cannot encrypt without entire plaintext

## Types of Hash Functions:

Fast Collision Resistant Hashes: MD and SHA family, use SHA-256, SHA-384, SHA-512, or SHA-3

Password Hashes: bcrypt, PBKDF2, scrypt

MAC Hashes (Message Authentication Code): CBC-MAC, HMAC