Symmetric ciphers:
Obvious requirements: One-to-one (invertible)
Substitution and permutation are invertible operations
We can use combinations of both.

Desired features: Confusion, Diffusion.
The only secret is the key. Security is based only on the fact that one cannot brute force a long-enough key.
The algorithm is known to every one.

Good cipher: Attack complexity is the same as brute force complexity
Strong cipher: Good cipher with sufficiently large key length to render brute force attacks infeasible
We can always realize a strong cipher from a good cipher by using multiple encryption.
Meet-in-the-middle attack: Double encryption is not as secure as we would wish. More generally n repeated encryptions using a good
cipher with key length k results in a strong cipher with brute force complexity \((n-1)k\)

Symmetric Cipher Types: Block cipher, Stream cipher
Symmetric cipher modes of operation

Hash functions:
n-bit hash function maps any bit sequence (of any length) to a seemingly random n-bit value (hash)
Pre-image resistance: Brute force complexity \(2^n\)
Collision resistance: Brute force complexity \(2^{n/2}\)

HMAC (Hashed message authentication code)
Hash of a message M along with a key K
HMAC provides two assurances to the receiver (who shares the key K with the sender)
1) The message was not tampered with en route
2) The sender knows K

Asymmetric Cryptography:
Every entity has a public and private key pair.
Every entity generates a key pair.
Choose private key (at random). Compute corresponding public key
Private key cannot be computed from the public key.

Uses mathematical operations with special features to ensure this: 1) difficulty of factoring 2) difficulty in finding discrete logarithms
From RSA secrets p and q public key n = pq can be easily computed. But given n we cannot compute p or q
From El Gamal secret a we can easily compute \(\alpha = g^a \mod p\). Cannot compute a given \(\alpha, g, p\)

What is not expensive: multiplication, exponentiation, determining multiplicative inverse, primality checking (O(n) complexity for n-bit numbers).
What is difficult: Factorization, computing discrete logarithms - O(2^n) complexity.

Encryption: Perform operations on the plain text P using the public key of the receiver. The operations can be inverted only by the receiver (using the private key).
Signature: Sender performs operation on some message M using his private key. The operations can be reversed by anyone with access to the public key of the sender.

Uses: Asymmetric cryptography is primarily used only for two purposes:
1) encrypting session keys: For a message from A to B,
   i) A chooses a random session secret K
   ii) encrypts the secret K using B's public key (creates a “digital envelope”) and
   iii) encrypts the message using the session secret and / or append a HMAC for the message
2) “encrypting” the hash of the message M to be signed (for signatures)
   i) Source computes hash \(h\) of the message \(M\)
   ii) encrypts the hash \(h\) using its private key to obtain the signature \(S\)
   iii) Sends \(M\) and \(S\)
**Challenge-Response protocols:**
Principals A and B.  1) A claims “I’m A.”  2) B issues a challenge.  3) A responds.  4) B is satisfied that the entity at the other end is A.  5) Now A challenges B.  6) B responds.
Used under two scenarios: Scenario 1: A and B already share a secret K.  Scenario 2: A and B have asymmetric key pairs
Challenges: 1) Reflection attack; 2) Replay attacks; 3) Man-in-the-middle attacks;
Solutions: 1) Use HMACs with nonces; 2) Time stamps 3) certified key pairs.

Key Distribution: Setting up shared keys
PKI: Certificate authority (CA) signs public keys of all participants. All participants have access to the public key of the CA.
Three step process: 1) generation of key pairs.  2) registration  3) obtaining certificate.
  *Generation:*  1) Choose random private key  2) compute public key
  *Registration:* Submit affiliations to the CA, CA will engage in challenge response protocols to determine that you have the private key corresponding to you public key.
  *Signing:* The X.509 certificate (indicating your affiliations and public keys amongst other things) is signed by the CA

Hierarchical PKI: Hierarchical organization of CAs. Root CA at the top of the tree. CAs who sign public keys of entities are at the leaves of the tree.
Forward and reverse signatures.

PKI issues:  Who will act as the root CA?
  Revocation?
  Large certificate chains implies increased bandwidth and computational overheads.

**PGP**
Public key distribution without using CAs.
Peer-to-peer system. All entities act as CAs.
Every entity assigns a trust to every other entity (OT – owner trust)
Every entity seeks other entities to sign its public key.
The OT that A attributes to B is the extent of signature trust (ST) that will be assigned by A to a public key signed by B
Every person assigns a key legitimacy (KL) to all public keys stored in its public key ring.

Private key ring of A (consisting of all private keys of A) is encrypted using a password chosen by A.

**Kerberos:**
Access control mechanism for access to servers by clients
Components: authentication server (AS), ticket granting server (TGS), Clients (C), Servers (S)
All clients share a secret (password) with the authentication server (AS)
All servers share a secret with a ticket granting server (TGS)

Single sign-on mechanism:
Beginning of the day the clients sign-on with the AS.
Clients get a ticket for approaching the TGS
When ever a client needs the service of a server, client approaches TGS
TGS issues individual tickets for every server.