Aims

- **Digital signatures** allow a principal to cryptographically bind (a representation of) its identity to a piece of information.
- Signatures can help establish security properties such as:
  - authentication
  - accountability/non-repudiation
  - unforgeability
  - integrity
  - verifiability by independent, public or 3rd party
- Digital signatures are the asymmetric analogue of MACs, with a crucial difference. MACs can’t distinguish which of A or B provided integrity to a message (so no non-repudiation or independent verifiability).
- NB: **electronic signature** is a more general notion.

Signature mechanism

A signature mechanism for principal A is given by:

- A message space \( M \) of messages for signing
- A set \( S \) of signatures (e.g. strings \( \{0, 1\}^n \))
- A secret **signing function** \( S_A : M \to S \)
- A public **verification function** \( V_A : M \times S \to \text{Bool} \)

satisfying the correctness and security properties:

1. \( V_A(m, s) = \text{true} \) if and only if \( S_A(m) = s \).
2. For any principal other than \( A \), it is computationally infeasible to find for any \( m \in M \), an \( s \in S \) such that \( V_A(m, s) = \text{true} \).

Usually use a public algorithm yielding key-indexed families \( \{S_s : s \in K\} \) of signing and verification functions \( \{V_v : v \in K\} \). Principal advertises \( v \).

Remark: nobody has proved a signature mechanism satisfying 2 exists, although there are good candidates.

Using a signature scheme

- To sign a message the **signer** \( A \)
  1. Computes \( s = S_A(m) \).
  2. Sends the pair \( (m, s) \).
- To verify that a signature \( s \) on a message \( m \) was created by \( A \), another principal, the **verifier**:
  1. Obtains the verification function \( V_A \) for \( A \).
  2. Computes \( u = V_A(m, s) \).
  3. **Accepts** the signature if \( u = \text{true} \), **Rejects** it if \( u = \text{false} \).
Digital signatures with a TTP
- Given a trusted third party, it is possible to use symmetric cryptography techniques.
- Let secure Sam S be the TTP, who shares a key with each principal.
- For A to send a signed contract M to B, S acts as an intermediary.
  - Message 1. A → S: \( \{ M \}_{K_{as}} \)
  - Message 2. S → B: \( \{ M \}_{K_{bs}} \)
  (like Wide Mouthed Frog key exchange protocol, M should include time-stamps and names).
- If A and B disagree about a signature, a judge Judy can verify the contracts also using S:
  - Message 1. J → S: \( \{ M \}_{K_{as}}, \{ M \}_{K_{bs}} \)
  - Message 2. S → J: \( \{ \text{yes or no} \}_{K_{js}} \)

Digital signatures from PK encryption
- Suppose we have a public-key encryption scheme with \( M = C \), and \((d,e)\) a key-pair. Then because \( E_d \) and \( D_e \) are both permutations on \( M \), we have that:
  \[
  D_e(E_d(m)) = E_d(D_e(m)) = m \quad \text{for all } m \in M
  \]
  A public-key scheme of this type is called reversible.
- RSA is reversible, but not every PK scheme is.
- We can define a digital signature scheme by reversing encryption and decryption:
  - Message space \( M \), signature space \( C \) (\( = M \)).
  - the signing function \( S_A = D_d \)
  - the verification function \( V_A \) is defined by
    \[
    V_A(m,s) = \begin{cases} 
      \text{true} & \text{if } E_d(s) = m, \\
      \text{false} & \text{otherwise}. 
    \end{cases}
    \]

Attacks on signature schemes [HAC]
- An adversary wants to forge signatures. Cases:
  1. Total break. Adversary can compute the private key or find an equivalent signing function.
  2. Selective forgery. Adversary can create a valid signature for some chosen message, without using the signer.
  3. Existential forgery. Adversary can create a valid signature for at least one message, without explicit choice of the message. May involve signer.
- The adversary may have different knowledge levels. For PK schemes:
  1. Key-only attack: adversary only knows PK.
  2. Known-message attack: adversary has signatures for some known (not chosen) messages.
  3. Chosen-message attack: adversary can obtain signatures for messages of his choosing. Messages may be determined in advance or in adaptive way, using signer as oracle.

Existential forgery
- The previous scheme is too simple because signatures are forgable: a principal B can generate a random \( s \in C \) as a signature, apply the public encryption function to get a message \( m = E_d(s) \), and transmit \((m,s)\).
- Obviously this verifies! It is an example of existential forgery.
- The message \( m \) is not likely to be of B’s choosing (and probably garbage).
- But this ability violates property 2 given earlier.

Signatures with redundancy
- A fix to reduce likelihood of existential forgery is to take \( M' \subset M \) to be messages with a special redundant structure, which is publicly known e.g., messages padded to an even length, surrounded with a fixed bit pattern.
- This format is easily recognized by the verifier:
  \[
  V_A(s) = \begin{cases} 
    \text{true} & \text{if } E_d(s) \in M', \\
    \text{false} & \text{otherwise}. 
  \end{cases}
  \]
- Now A only transmits the signature \( s \), since the message \( m = E_d(s) \) can be recovered by the verification function.
- This property is message recovery. The scheme is called a signature scheme with recovery.
- Existential forgery is now less likely.

Signatures and hash functions
- In practice, usually the signing function is constructed by first making a hash of the input document, and signing that. Reasons:
  1. efficiency: signature is on smaller text
  2. avoid attacks on cipher system
- Signer: computes and transmits \((m,s)\) where \( s = S_A(h(m)) \).
- Verifier: computes \( h(m) \) and verifies \( V_A(h(m),s) \).
- The hash function must satisfy appropriate properties (see Hash Functions lecture).
- This is called a signature scheme with appendix.
**RSA Signatures**

- Setup: \( n = pq \) computed as product of two primes. \( ed \equiv 1 \mod \phi(n) \). \((e, n)\) is the public key.
- To sign a message \( m \), compute the signature \( s = h(m)^d \mod n \). Only the owner of the private key \( d \) is able to compute the signature.
- To verify the signature, upon receipt of \((m, s)\), compute \( se \mod n \) and verify whether it equals \( h(m) \).

**Distributed RSA Signatures**

- Signatures can optionally be distributed so that each of \( t \) users contributes to the signature. A trusted party \( T \) computes \( t \) shares such that

\[
d = \sum_{i=1}^{t} d_i \mod \phi(n)
\]

and securely distributes \( d_i \) to each user \( i \).
- To compute a signature on a message \( m \), each user \( i \) computes \( o_i = h(m)^{d_i} \mod n \).
- A signer can compute the resultant signature as

\[
s = \prod_{i=1}^{t} o_i \mod n
\]

- Secret sharing can also be used so that \( l < t \) users could be used to construct a signature.

**ElGamal Signatures**

- Setup as encryption: \( p \) an appropriate prime, \( g \) a generator of \( \mathbb{Z}_p^\ast \), and the private signing key, \( d \) a random integer with \( 1 \leq d \leq p - 2 \).
- The public verification key is \((p, g, g^d \mod p)\).
- To sign a message \( m \), \( 0 \leq m \leq p \), the signer picks a random secret number \( r \) with \( 1 \leq r \leq p - 2 \) and \( \gcd(r, p - 1) = 1 \), and computes:

\[
\mathbf{S}_d(m) = (e, s) \quad \text{where} \quad e = g^m \mod p \quad \text{and} \quad s = r^{-1} (m - dr) \mod (p - 1).
\]

- The verification function checks that \( 1 \leq e \leq p - 1 \), and an equation:

\[
V_{(p, g, d^e)}(m, (e, s)) = \begin{cases} 
\text{true} & \text{if } (g^e)^s \equiv g^m \mod p, \\
\text{false} & \text{otherwise}.
\end{cases}
\]

- Verification works because for a correct signature,

\[
(g^e)^s \equiv g^{d^e} \equiv g^{m+rs} \equiv g^m \mod p.
\]

**From ElGamal to DSA**

- The Digital Signature Algorithm is part of the NIST Digital Signature Standard (FIPS-186).
- Based on ElGamal, but with improved efficiency.
- The first digital signature scheme to be recognized by any government.
- Based on two primes: \( p \) which is 512–1024 bits long, and \( q \) which is a 160-bit prime factor of \( p - 1 \).
- A signature signs a SHA-1 hash value of a message. (In fact, ElGamal signing should be used with a hash function to prevent existential forgery)
- **Security** of both ElGamal and DSA schemes relies on the intractability of the DLP.
- Comparison with RSA signature scheme: key generation is faster; signature generation is about the same; DSA verification is slower. Verification is the most common operation in general.

**Summary: Digital Signature Schemes**

- RSA, ElGamal, DSA already described. There are several variants of ElGamal, including schemes with message recovery.
- Notice difference between randomized and deterministic schemes.
- Schemes for one-time signatures (e.g., Rabin, Merkle), require a fresh public key for each use.
- Typically more efficient than RSA/ElGamal methods.
- But tedious for multiple documents
- E-cash protocols use blind signature schemes that prevent the signer (e.g., a bank) linking a signed message (e.g., the cash) with the user.
- For real world security guarantees:
  - obtaining correct public key is vital;
  - non-repudiation supposes that private key has not been stolen;
  - we may require secure time stamps.

**References**

- [Handbook of Applied Cryptography](https://www.cacr.math.uwaterloo.ca/hac)
- [Cryptography: An Introduction](http://www.cs.bris.ac.uk/~nigel/Crypto.Book/)

**Recommended Reading**

Chapter 14 (14.2–14.4, 14.7) of Smart (3rd Ed).