Digital Signatures

Objectives

- Features of hand-written signatures in Digital World
- Ensure hardness of forgery

Hand-written Signatures

- Function: bind a statement/message to its authors.
- Verification is public. (against a prior authenticated one)

Hand-written Signatures

- Function: bind a statement/message to its authors.
- Verification is public. (against a prior authenticated one)
- Properties:
  - Correctness: A correct signature should always be verified true.

Hand-written Signatures

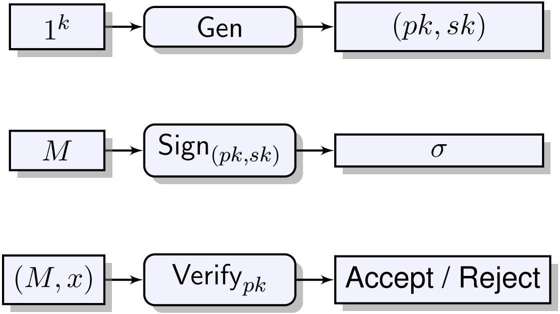
- Function: bind a statement/message to its authors.
- Verification is public. (against a prior authenticated one)
- Properties:
  - Correctness: A correct signature should always be verified true.
  - Security: Hard to forge.

Signature Schemes

Signature Scheme (Gen, Sign, Verify)

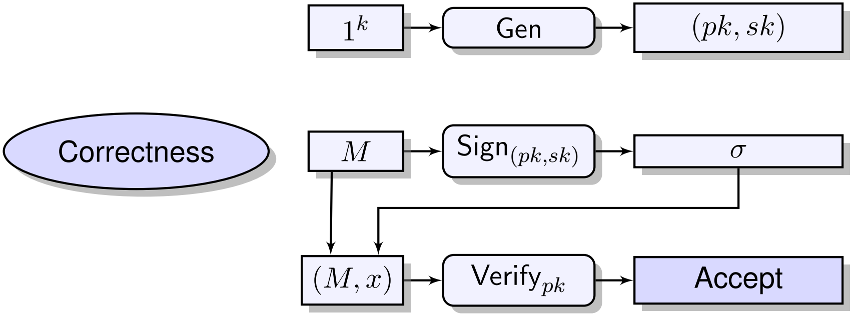
Required Properties:

- Correctness
- Unforgeability

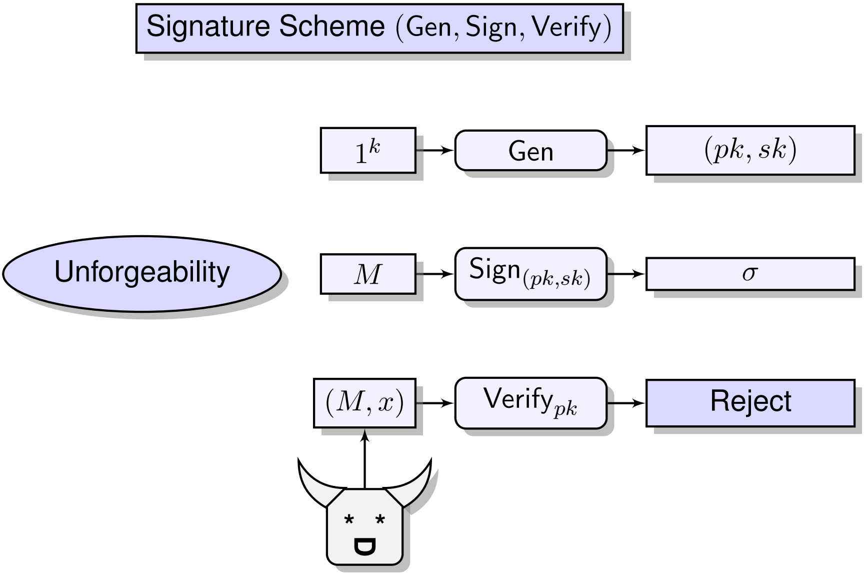


Signature Schemes

Signature Scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$



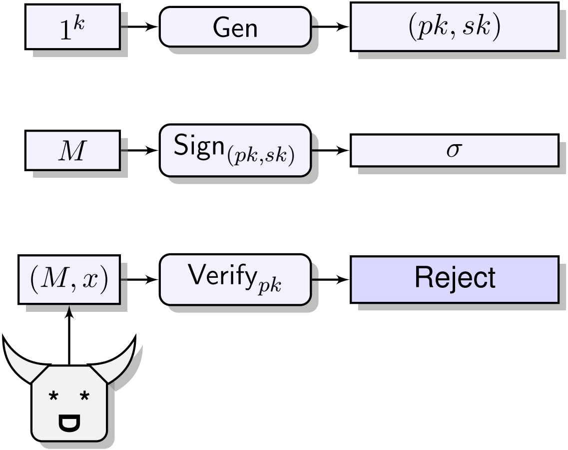
Signature Schemes



Signature Schemes

Signature Scheme (Gen, Sign, Verify)

Unforgeability:
Must output forgery for a message for which the attacker did not request the signature.



Signature Schemes Designs: RSA Full Domain Hash

- **Public Functions** A hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- **Keygen:** Run RSA.Keygen. $pk = (e, N)$, $sk = (d, N)$.
- **Sign**: Input: sk, M. Output
  $\sigma = \mathsf{RSA.Dec}(sk, H(M)) = H(M)^d \mod N$

Signature Schemes Designs: RSA Full Domain Hash

- **Public Functions** A hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- **Keygen:** Run RSA.Keygen. $pk = (e, N)$, $sk = (d, N)$.
- **Sign**: Input: sk, M. Output $\sigma = \mathsf{RSA.Dec}(sk, H(M)) = H(M)^d \mod N$
- **Verify**: Input: pk, M, σ. If $\mathsf{RSA.Enc}(pk, \sigma) = H(M)$ output accept, else reject

Signature Schemes Designs: RSA Full Domain Hash

- **Public Functions** A hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$
- **Keygen:** Run RSA.Keygen. $pk = (e, N)$, $sk = (d, N)$.
- **Sign**: Input: sk, M. Output
  $\sigma =$ RSA.Dec$(sk, H(M)) = H(M)^d \mod N$
- **Verify**: Input: pk, M, σ. If RSA.Enc$(pk, \sigma) = H(M)$ output accept, else reject
- If $\sigma^e \mod N = H(M)$, output accept, else reject.

Signature Schemes Designs: RSA Full Domain Hash

- **Public Functions** A hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$
- **Keygen:** Run RSA.Keygen. $pk = (e, N)$, $sk = (d, N)$.
- **Sign**: Input: sk, M. Output $\sigma =$ RSA.Dec$(sk, H(M)) = H(M)^d \mod N$
- **Verify**: Input: pk, M, σ. If RSA.Enc$(pk, \sigma) = H(M)$ output accept, else reject
- If $\sigma^e \mod N = H(M)$, output accept, else reject.

note

A hash function takes strings of arbitrary length as input and produces a fixed length output. For cryptographic hash functions, given a z, it is very expensive to find x such that $H(x) = z$