# RSA Signature

- KeyGen():
  - Randomly pick two large primes, $p$ and $q$
  - Compute $n = pq$
    - $n$ is usually between 2048 bits and 4096 bits long
  - Choose $e$
    - Requirement: $e$ is relatively prime to $(p - 1)(q - 1)$
    - Requirement: $2 < e < (p - 1)(q - 1)$
  - Compute $d = e^{-1} \bmod (p - 1)(q - 1)$
  - **Public key**: $n$ and $e$
  - **Private key:** $d$

# RSA Digital Signature Algo

Step1: Generate a hash value, or message digest, mHash from the message *M* to be signed

Step2: Pad mHash with a constant value padding1 and pseudorandom value salt to form *M'*

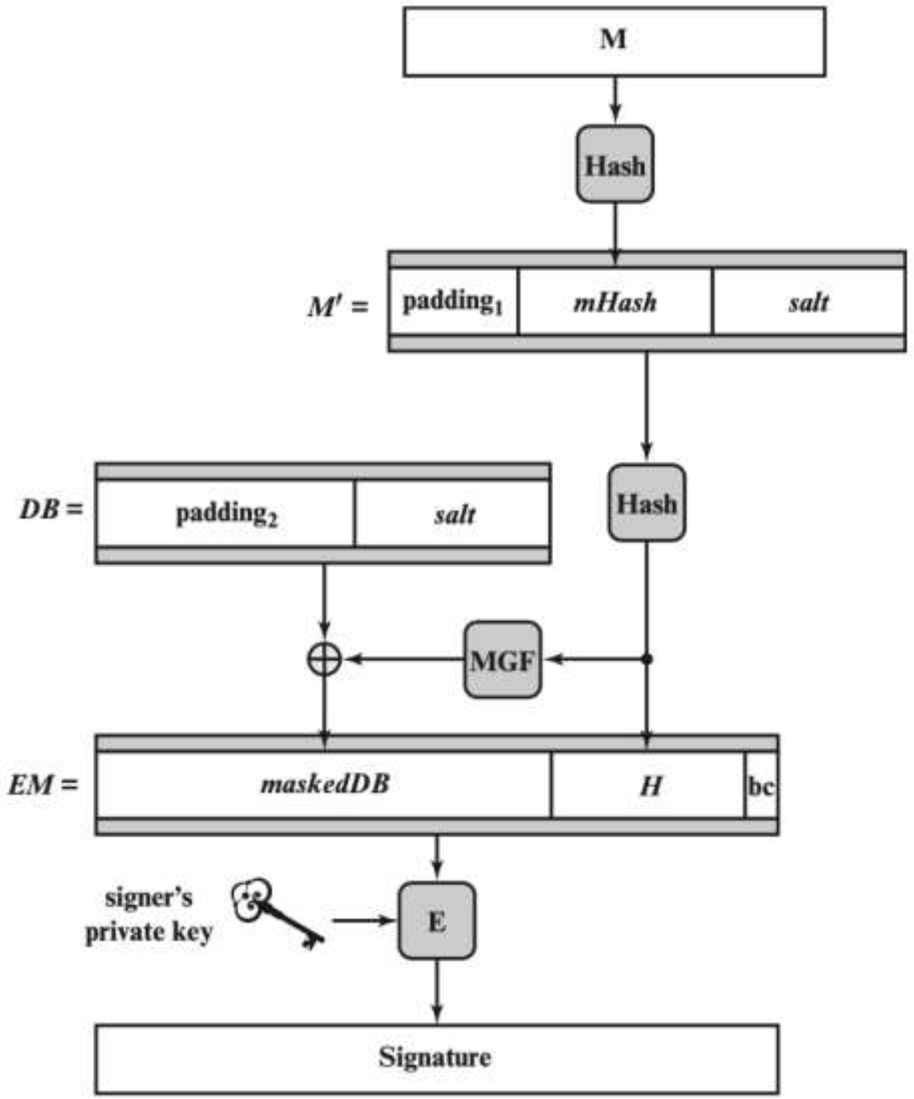
*Step3:* Generate hash value *H* from *M'*

*Step4:* Generate a block DB consisting of a constant value padding 2 and salt

Step5: Use the mask generating function MGF, which produces a randomized out-put from input *H* of the same length as DB

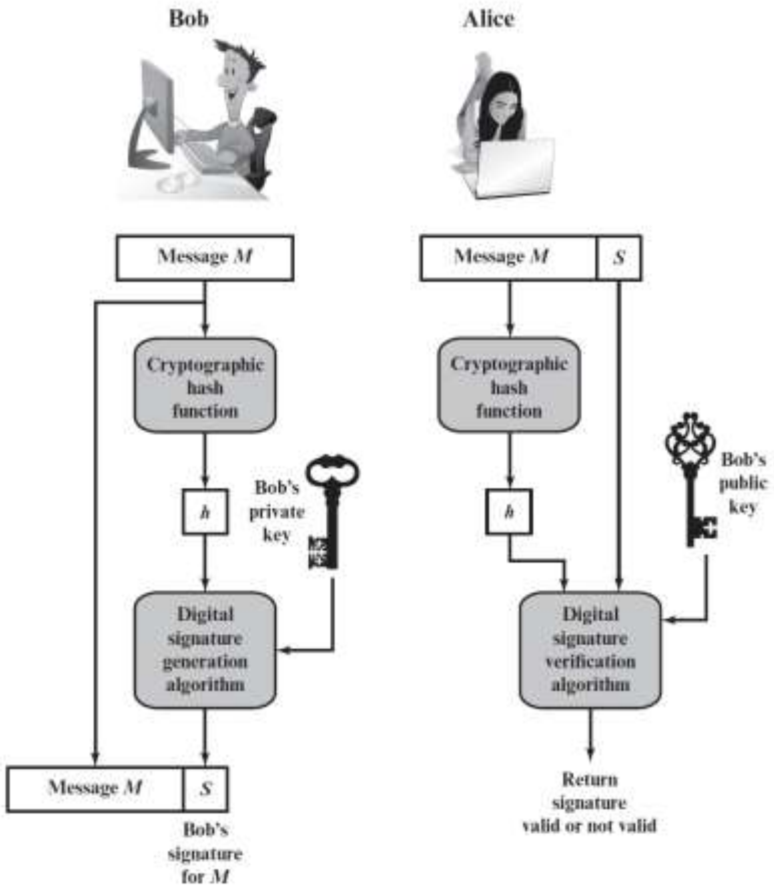
Step 6: Create the encoded message (EM) block by padding *H* with the hexadecimal constant bc and the XOR of DB and output of MGF

Step 7: Encrypt EM with RSA using the signer's private key



# RSA Signatures

- Sign($d$, $M$):
  - Compute $H(M)^d \bmod n$
- Verify($e$, $n$, $M$, $sig$)
  - Verify that $H(M) \equiv sig^e \bmod n$




(a) Bob signs a message

(b) Alice verifies the signature

# RSA Signatures: Correctness

Theorem: $sig^e \equiv H(M) \bmod N$

Proof:

$$sig^e = [H(M)^d\ ]^e\ mod\ N \quad = H(M)^{ed} \text{ mod N}$$

$$= H(M)^{k\phi(n)+1} \text{ mod N}$$

$$= [H(M)^{\phi(n)}]^k \cdot H(M) \quad \text{mod N}$$

$$= \text{H(M)} \quad \text{mod N}$$

# RSA Digital Signature: Security

- **Necessary hardness assumptions:**
  - **Factoring hardness assumption:** Given *n* large, it is hard to find primes pq = n
  - **Discrete logarithm hardness assumption:** Given *n* large, *hash*, and $hash^d$ *mod n*, it is hard to find *d*
- Salt also adds security
  - Even the same message and private key will get different signatures

# Hybrid Encryption

- Issues with public-key encryption
  - Notice: We can only encrypt small messages because of the modulo operator
  - Notice: There is a lot of math, and computers are slow at math
  - Result: We don't use asymmetric for large messages
- **Hybrid encryption**: Encrypt data under a randomly generated key *K* using symmetric encryption, and encrypt *K* using asymmetric encryption
  - $\text{Enc}_{\text{Asym}}(\text{PK}, \text{K}); \text{Enc}_{\text{Sym}}(\text{K}, \text{large message})$
  - Benefit: Now we can encrypt large amounts of data quickly using symmetric encryption, and we still have the security of asymmetric encryption