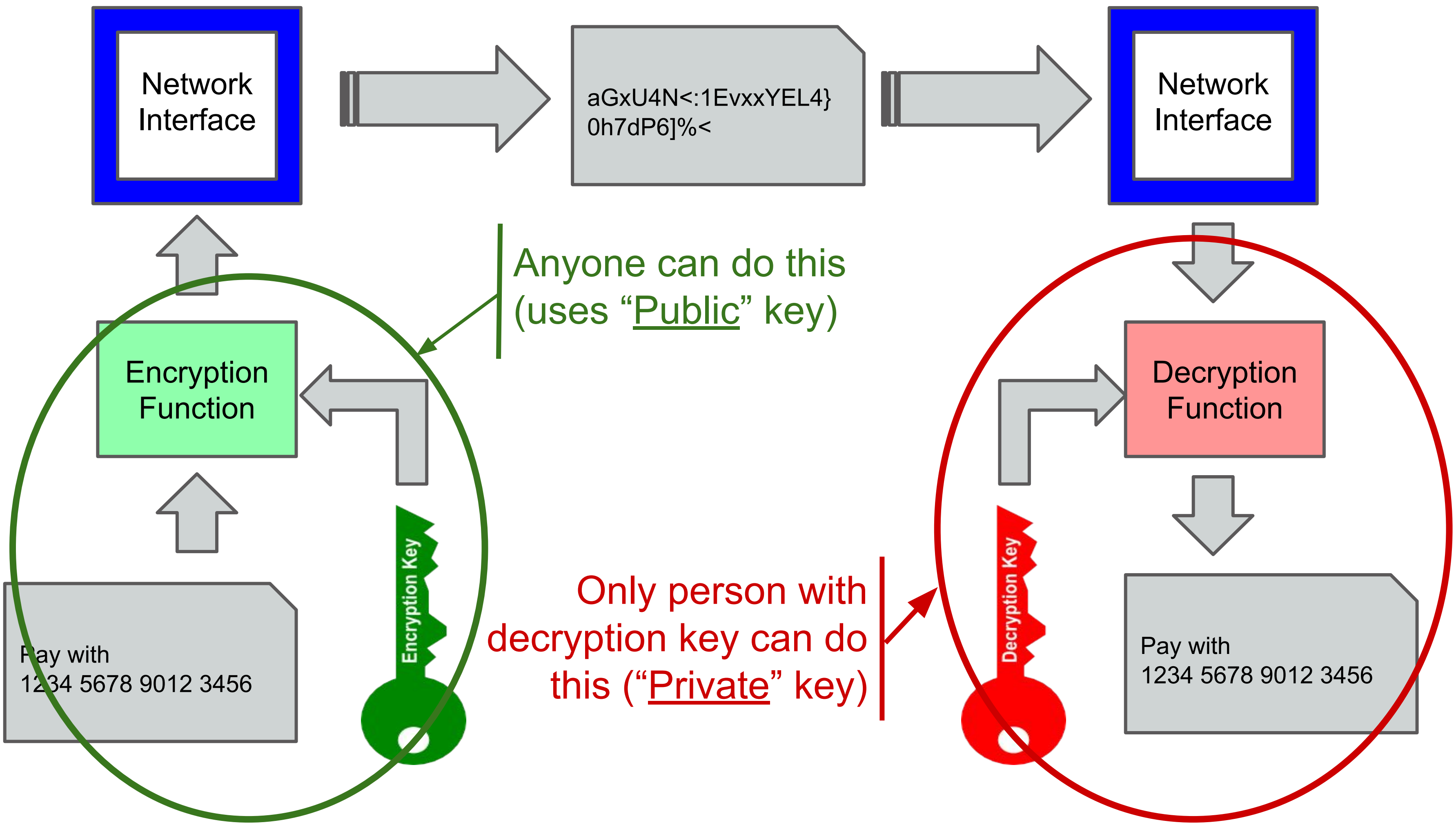
# CSC 580
# Cryptography and Computer Security

*Digital Signatures*
*(Sections 13.1, 13.2, 13.4, 13.6)*

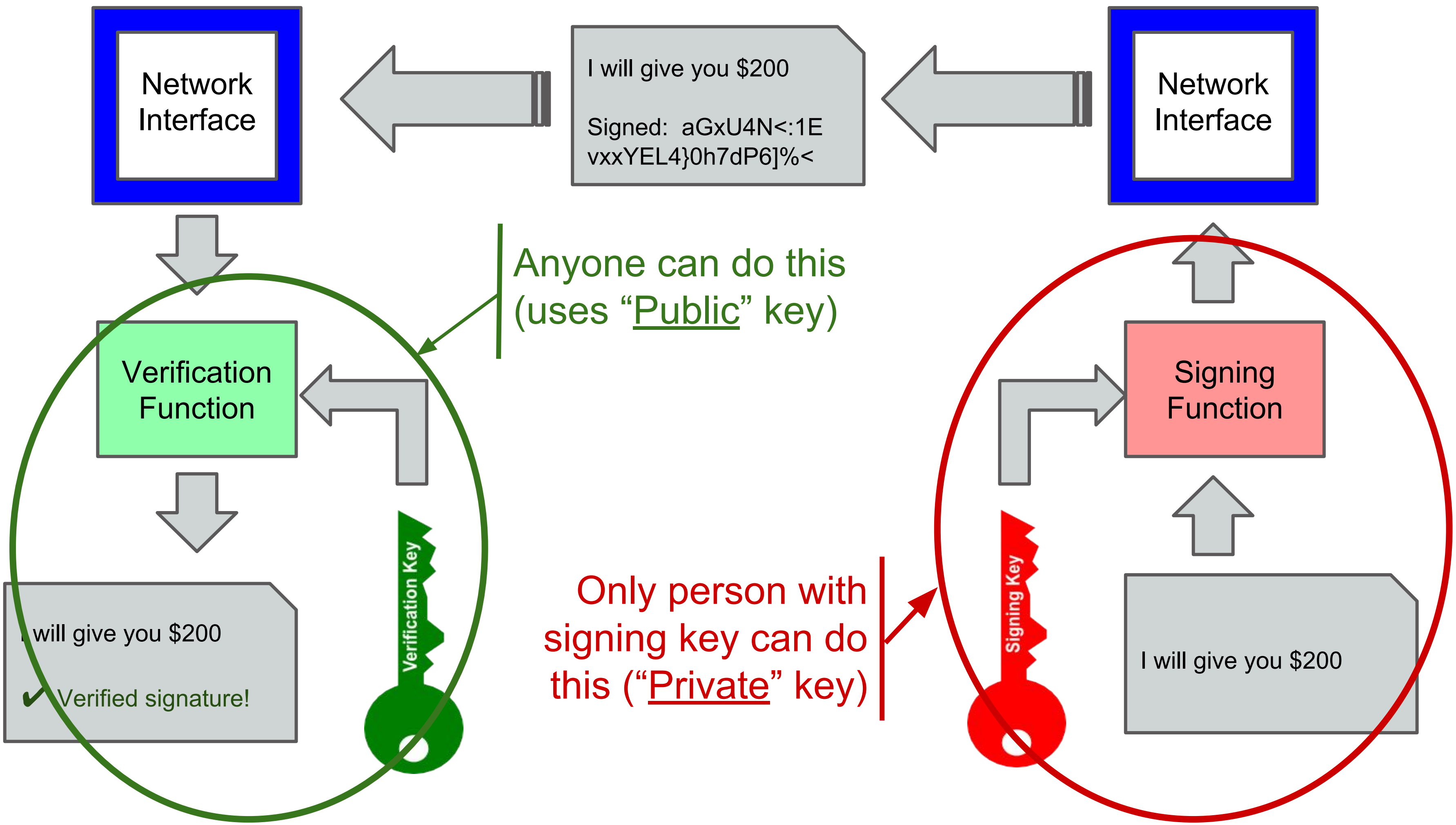
# Digital Signatures - Idea

*Public key encryption idea*



# Digital Signatures - Idea

*Digital signature idea*



# Digital Signatures - How it Works

Signature scheme consists of three algorithms:

- *Generate keypair*: Given keylength (security param) gives (PU,PR)
- *Sign*: Takes message M and PR, and produces signature sig
- *Verify*: Takes M, PU, and sig, and outputs true (verified) or false

Like public key encryption, sign/verify operations are slow!

- So don't run entire (possibly long) message through functions
- First hash, then sign H($M$)

Is this combination secure? Yes! Why: Assume adversary knows valid sigs $(M_1, sig_1), (M_2, sig_2), \ldots, (M_n, sig_n)$ and can find a forgery $(M, sig)$.

- If $H(M) = H(M_i)$ for some $M_i$ $\rightarrow$ found a collision in H, should be impossible!
- If $H(M) \neq H(M_i)$ for all $M_i$ $\rightarrow$ then $(H(M), sig)$ is a forger for sig scheme

# Digital Signatures - Security Model

```
A^𝒮(PU)
   // Arbitrary precomputation
   while (not done):
      m = // compute query message
      s = 𝒮(m)
      Known = Known ∪ (m,s)
      // More computing
   (m’, s’) = // compute claimed forgery
   Return (m’,s’)
```

Adversary wins if there is no pair `(m’,x)` in `Known` and Verify(`m’,s’`) = true

Note:

- Adversary picks oracle query messages, and can adapt as it learns
  - That makes this an “adaptive chosen message” attack
- Any valid signature wins - only restriction is that m’ hasn’t been queried
  - That makes this an “existential forgery attack”

Security is Existentially Unforgeable under Adaptive Chosen Message Attack (EUF-CMA)

# ElGamal

As in Diffie-Hellman, let $p$ be a prime and $g$ be a primitive root

Key Generation

*Note similarity to Diffie-Hellman*

1. Pick random $PR \in \{2, \ldots, p\text{-}1\}$
2. Compute $PU = g^{PR} \bmod p$
3. Private (signing) key is $PR$ ; Public (verification) key is $PU$

Signing a message M

1. Pick random $k \in \{2, \ldots, p\text{-}1\}$ that is relative prime to ($p$-1)
2. Compute $r = g^k \bmod p$
3. Compute $k^{-1} \bmod (p\text{-}1)$
4. Compute $s = k^{-1}\,(\mathrm{H}(M) - PR{*}r) \bmod (\mathrm{p}\text{-}1)$
5. Signature is the pair ($r$,$s$)

Verifying a signature ($r$,$s$) on message M:

1. Check if $g^{\mathrm{H}(M)} \equiv PU^{r} * r^{s} \pmod{p}$    [accept if true, reject if false]

# ElGamal

As in Diffie-Hellman, let $p$ be a prime and $g$ be a primitive root

Key Generation

1. Pick random $PR \in \{2, \ldots, p\text{-}1\}$
2. Compute $PU = g^{PR} \bmod p$
3. Private (signing) key is $PR$ ; Public (verification) key is $PU$

*Note similarity to Diffie-Hellman*

Signing a message M

1. Pick random $k \in \{2, \ldots, p\text{-}1\}$ that is relative prime to ($p$-1)
2. Compute $r = g^k \bmod p$
3. Compute $k^{-1} \bmod (p\text{-}1)$
4. Compute $s = k^{-1}\,(\mathrm{H}(M) - PR{*}r) \bmod (\mathrm{p}\text{-}1)$
5. Signature is the pair ($r$,$s$)

Observation: Expensive computations (powering and inverse), but they don't depend on M - precompute!

Verifying a signature ($r$,$s$) on message M:

1. Check if $g^{\mathrm{H}(M)} \equiv PU^{r} * r^{s} \pmod{p}$ [accept if true, reject if false]

# Why does this work for valid sigs?

*Important math fact*: If $x \equiv y \pmod{p-1}$ then $a^x \equiv a^y \pmod{p}$.

*Proof*: If $x \equiv y \pmod{p-1}$ then there exists a $k$ such that $x-y = k*(p-1)$, so $x = k*(p-1)+y$. Then $a^x = a^{k*(p-1)+y} = a^{k*(p-1)}*a^y = (a^{p-1})^k*a^y$. By Fermat's Little Theorem, we know that $a^{p-1} \bmod p = 1$, so $(a^{p-1})^k*a^y \bmod p = a^y$. Therefore $a^x \equiv a^y \pmod{p}$.

What this means: To simplify $a^{formula}$, can simplify *formula* mod ($p$-1).

Applying this to ElGamal formulas:

$PU = g^{PR} \bmod p$

$s = k^{-1}\,(H(M) - PR*r) \bmod (p-1)$

Consider $PU^r * r^s \equiv g^{PR*r}\, g^{k*s} \equiv g^{PR*r+k*s}$ (mod p), and simplify exponent mod (p-1):

$PR*r + k*s \equiv PR*r + k*k^{-1}\,(H(M) - PR*r) \equiv PR*r + H(M) - PR*r \equiv H(M) \bmod (p-1)$

Therefore, $PU^r * r^s \equiv g^{H(M)} \pmod{p}$

# DSA - Digital Signature Algorithm

## Compared to ElGamal

### *ElGamal*

Let $q = p$-1

**Key Generation**

1. Pick random $PR \in \{2, \ldots, q\}$
2. Compute $PU = g^{PR} \bmod p$
3. Private key is $PR$ ; Public key is $PU$

**Signing a message M**

1. Pick rand $k \in \{2, \ldots, q\}$ with gcd(k,q)=1
2. Compute $r = g^k \bmod p$
3. Compute $k^{-1} \bmod q$
4. Compute $s = k^{-1}\,(\mathrm{H}(M) - PR^*r) \bmod q$
5. Signature is the pair ($r$,$s$)

Verifying signature ($r$,$s$) on message M:

1. Check if $g^{\mathrm{H}(M)} \equiv PU^{r} * r^{s}$ (mod $p$)

### *DSA*

$q$ is prime such that $q|p$-1, and let $g$ be a value with order $q$ [ $g^q \equiv 1$ (mod $q$) ]

**Key Generation**

1. Pick random $PR \in \{2, \ldots, q\}$
2. Compute $PU = g^{PR} \bmod p$
3. Private key is $PR$ ; Public key is $PU$

**Signing a message M**

1. Pick rand $k \in \{2, \ldots, q\text{-}1\}$
2. Compute $r = (g^k \bmod p) \bmod q$
3. Compute $k^{-1} \bmod q$
4. Compute $s = k^{-1}\,(\mathrm{H}(M) + PR^*r) \bmod q$
5. Signature is the pair ($r$,$s$)

Verifying signature ($r$,$s$) on message M:

1. Compute $w = s^{-1} \bmod q$
2. Check if $r \equiv (PU^{r*w} * g^{H(M)*w} \bmod p) \bmod q$

# DSA - The Digital Signature Algorithm
## History, Parameters, etc.

One component of NIST's Digital Signature Standard (DSS)

- DSS was adopted in 1993
- DSA dates back to 1991
- One goal: Only support integrity - not confidentiality
  - Why? Export restrictions!
  - Alternative signature scheme: RSA - also an encryption algorithm

Key and Parameter Sizes:

- ElGamal is similar to Diffie-Hellman modulus size ($N$ = number of bits)
  - 1024-bit $p$ was OK in 1990s - now suggest 2048-bit or 3072-bit
  - Signature two $N$-bit values (e.g., two 1024-bit values)
- DSA uses a computationally-hard subgroup
  - In 1990's $q$ was 160 bits (matching SHA1!)
  - Signature was then two 160-bit values (more compact than ElGamal)
  - Now suggest $q$ being 256 bits

# Reminder - RSA Algorithm

## From Public Key Encryption chapter

Key Generation:

Pick two large primes $p$ and $q$
Calculate $n=p*q$ and $\phi(n)=(p-1)*(q-1)$
Pick a random $e$ such that $\gcd(e, \phi(n))$
Compute $d = e^{-1} \pmod{\phi(n)}$ [*Use extended GCD algorithm!*]
Public key is $PU=(n,e)$ ; Private key is $PR=(n,d)$

Encryption of message $M \in \{0,..,n-1\}$:

$E(PU,M) = M^e \bmod n$

Decryption of ciphertext $C \in \{0,..,n-1\}$:

$D(PR,C) = C^d \bmod n$

Correctness - easy when $\gcd(M,n)=1$:

$$
\begin{aligned}
D(PR,E(PU,M)) &= (M^e)^d \bmod n \\
&= M^{ed} \bmod n \\
&= M^{k\phi(n)+1} \bmod n \\
&= (M^{\phi(n)})^k M \bmod n \\
&= M
\end{aligned}
$$

Also works when $\gcd(M,n)\neq 1$, but slightly harder to show...

# RSA Algorithm for Signatures

## "Textbook algorithm" - not how it's really done

Key Generation:

Pick two large primes $p$ and $q$

Calculate $n=p*q$ and $\phi(n)=(p-1)*(q-1)$

Pick a random $v$ such that gcd($v$, $\phi(n)$)

Compute $s = v^{-1} \pmod{\phi(n)}$ [*Use extended GCD algorithm!*]

Public key is $PU=(n,v)$ ; Private key is $PR=(n,s)$

Signing message $M \in \{0,..,n-1\}$:

Sign($PR$,$M$) = $M^s \bmod n$

Verification of signature $\sigma \in \{0,..,n-1\}$:

Verify($PU$,$M$,$\sigma$): Check if $M = \sigma^v \bmod n$

# RSA-PSS (Probabilistic Signature Scheme)

## How it's really done - with padding (similar to OAEP for encryption)

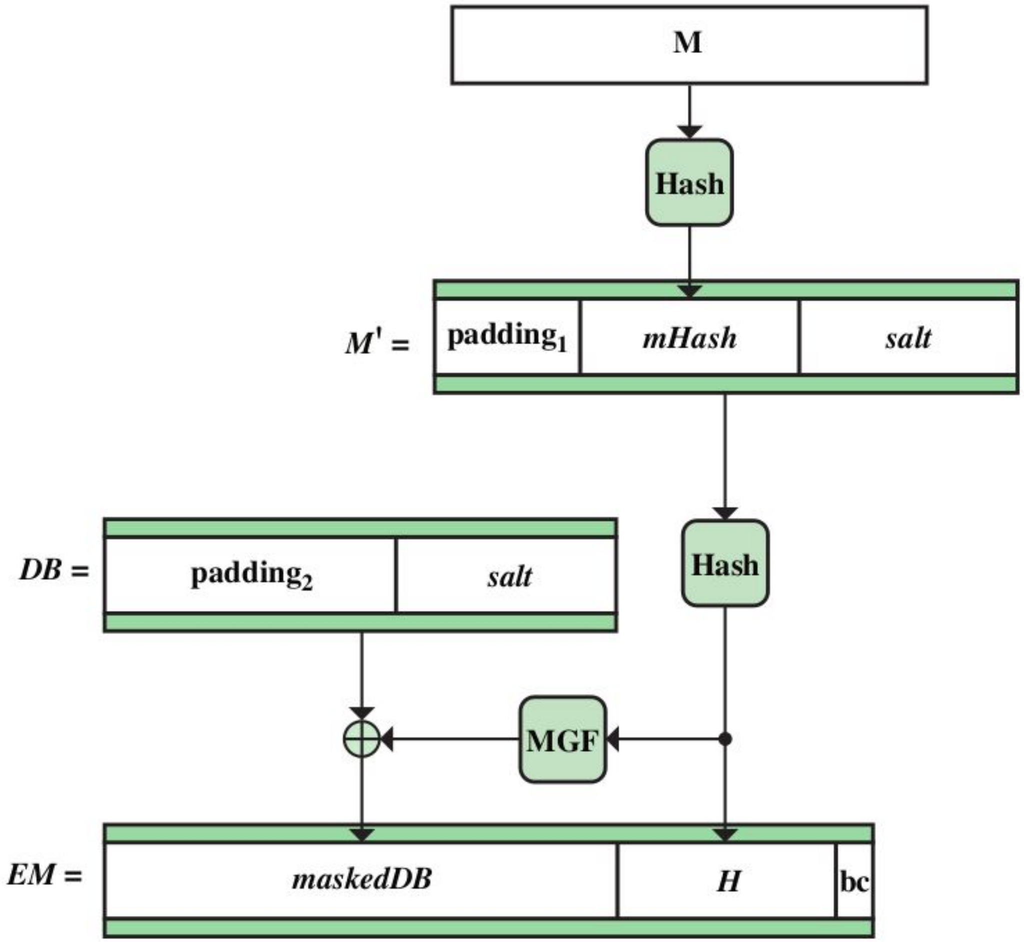



Figure 13.6 RSA-PSS Encoding

Invented (and proved secure) by Bellare and Rogaway

- Also inventors of OAEP and HMAC

Forging sigs w/ "textbook RSA"

- Pick random sig $R$
- Let message $M=R^v \bmod N$
- (M,R) is valid sig pair!

Modifying sigs ("blinding")

- Given $\sigma = M^s \bmod N$
- Compute $X = R^v \bmod N$
- Let M' = X*M mod N
- Let σ' = R*σ mod N
- Note $(\sigma')^v = R^v\sigma'^v = X*M$ = M' (mod $N$)