Public Key Cryptography (II)

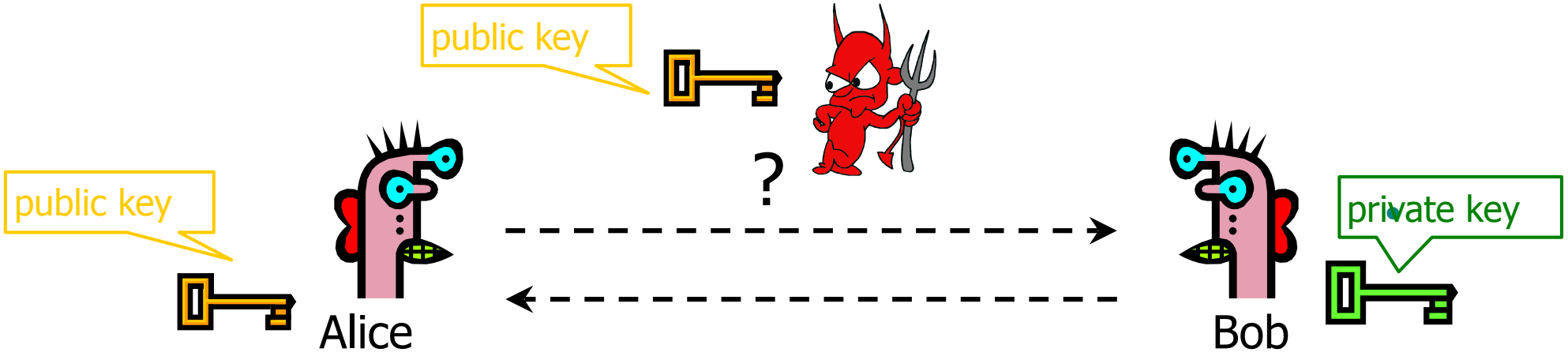
The era of “electronic mail” [Potter1977] may soon be upon us; we must ensure that two important properties of the current “paper mail” system are preserved: (a) messages are private, and (b) messages can be signed.

R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. January 1978.

Yan Huang

Credits: David Evans, Vitaly Shmatikov

Public-Key Cryptography



Given: Everybody knows Bob's public key

- How is this achieved in practice?

Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a message that only Bob can read

2. Bob wants to send a message that only Bob could have written

Some Number Theory Facts

- Euler totient function φ(n) where n≥1, is the number of integers in the interval [1,n] that are relatively prime to n
  - x and y are relatively prime if gcd(x, y) = 1
  - φ(n) is also the *size* of $\mathbb{Z}_n^*$

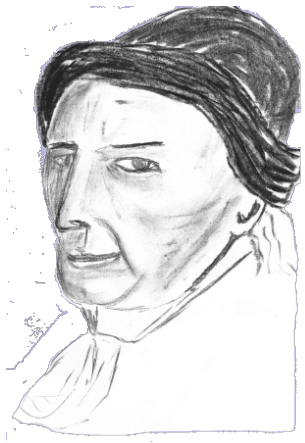
$$\mathbb{Z}_7^* = \{1,2,3,4,5,6\}, \quad \varphi(7) = \| \mathbb{Z}_7^* \| = 6$$

$$\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\}, \quad \varphi(15) = \varphi(3 \cdot 5) = \| \mathbb{Z}_{15}^* \| = (3-1) \cdot (5-1) = 8$$

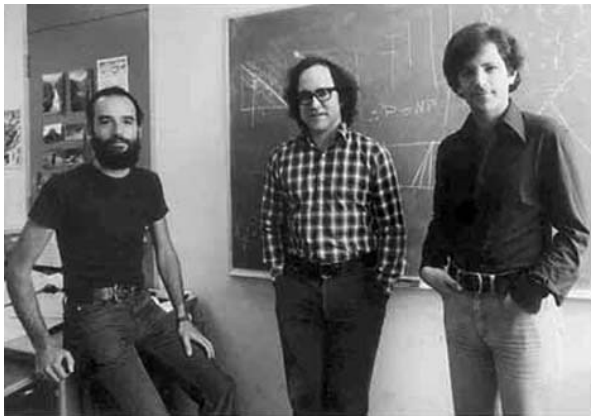
$$\varphi(n) = n \prod_{p|n,\ p:prime} (1 - 1/p)$$

- Euler's theorem:

$$\text{If } a \in \mathbb{Z}_n^*, \text{ then } a^{\phi(n)} \equiv 1 \bmod n$$



RSA Cryptosystem



[Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - At least 1024 bits each… need primality testing!
  - Compute n=pq
    - Note that $\varphi(n)=(p-1)(q-1)$
  - Choose small e, relatively prime to $\varphi(n)$
    - Typically, e=3 (may be vulnerable) or $e=2^{16}+1=65537$ (why?)
  - Compute unique d such that $ed \equiv 1 \bmod \varphi(n)$
  - Public key = (n,e); private key = d
- Encryption of m: $c = m^e \bmod n$
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

Why RSA Decryption Works

Because

$$e \cdot d \equiv 1 \bmod \varphi(n),$$

thus there exists integer k such that

$$e \cdot d = 1 + k \cdot \varphi(n)$$

So $m^{ed} \equiv m^{1+k \cdot \varphi(n)} \equiv m \mod n$. (Euler's theorem)

Why Is RSA Secure?

- RSA Problem: given c, n=pq, and e such that gcd(e, (p-1)(q-1))=1, find an e^{th} root of c modulo n.
- RSA Assumption: there is no *efficient* algorithm to solve RSA problem.
- Factoring problem: given positive integer n=pq where p, q are large primes (thousands of bits), factor n.
- If factoring is easy, then RSA problem is easy, but may be possible to break RSA without factoring n

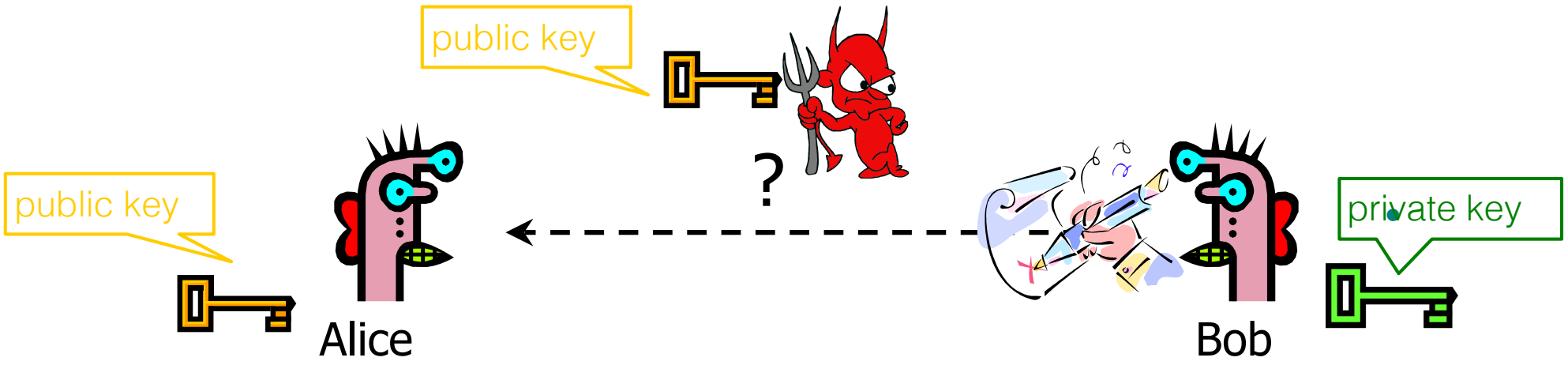
“Textbook” RSA Is Bad Encryption

- Deterministic
  - Attacker can guess plaintext, compute ciphertext, and compare for equality
  - If messages are from a small set (for example, yes/no), can build a table of corresponding ciphertexts
- Can tamper with encrypted messages
  - Take an encrypted auction bid c and submit $c(101/100)^e \bmod n$ instead
- Many other attacks to “Textbook RSA” (see [Katz&Lindell, CRC Press] Page 412-414)

Integrity in RSA Encryption

- "Textbook" RSA does not provide integrity
  - Given encryptions of m_1 and m_2, attacker can create encryption of $m_1 \cdot m_2$
    - $(m_1^e) \cdot (m_2^e) \bmod n \equiv (m_1 \cdot m_2)^e \bmod n$
  - Attacker can convert m into m^k without decrypting
    - $(m^e)^k \bmod n \equiv (m^k)^e \bmod n$
- In practice, OAEP is used: instead of encrypting m, encrypt $m \oplus G(r) \,\|\, r \oplus H(m \oplus G(r))$
  - r is random and fresh, G and H are hash functions
  - Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext
    - … if hash functions are "good" and RSA problem is hard

Digital Signatures: Basic Idea



Given: Everybody knows Bob's public key

Only Bob knows the corresponding private key

Goal: Bob sends a "digitally signed" message

To compute a signature, must know the private key

To verify a signature, only the public key is needed

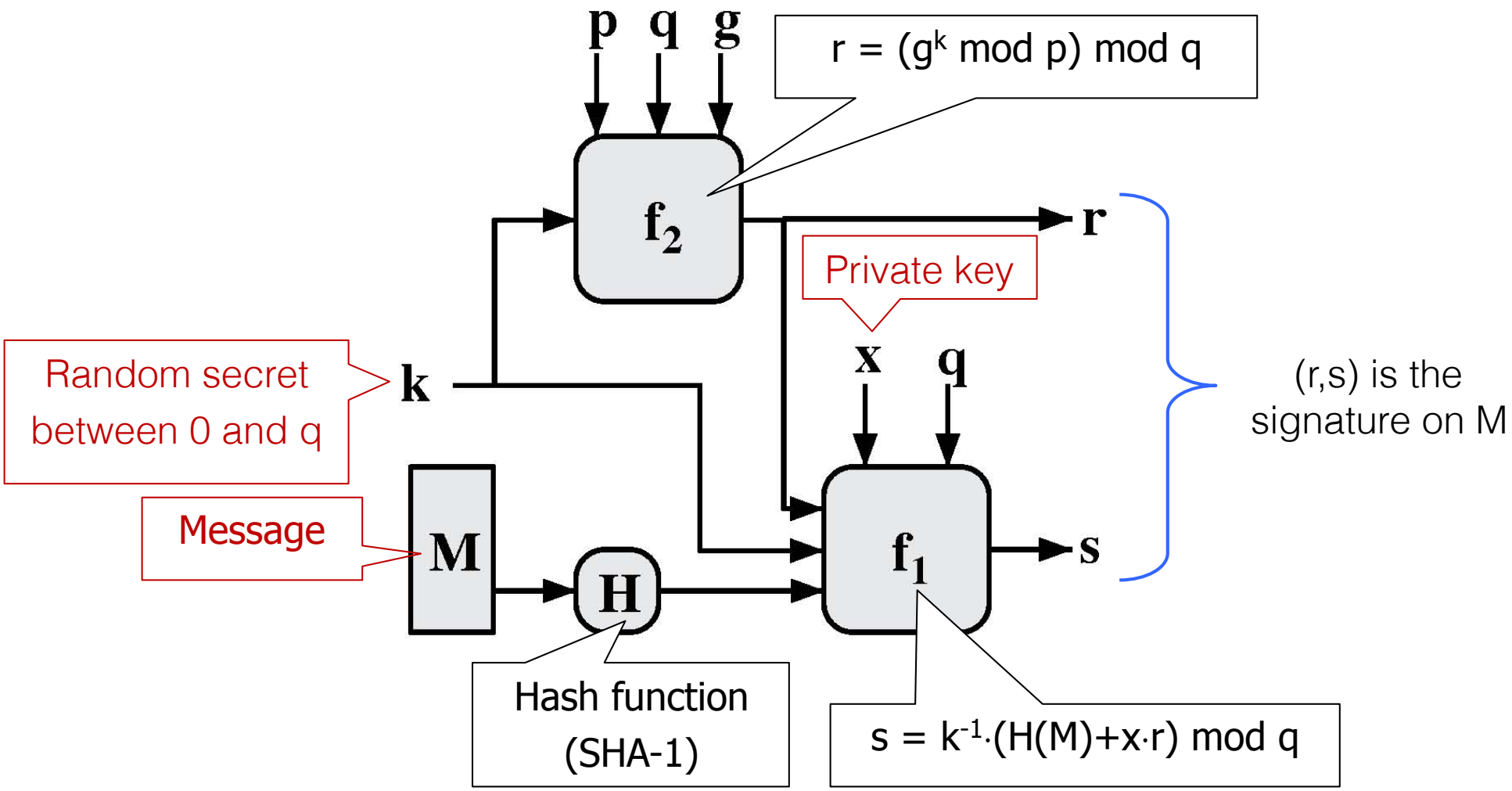
RSA Signatures

- Public key is (n,e), private key is d
- To sign message m: $s = Hash(m)^d \bmod n$
  - Signing and decryption are the same mathematical operation in RSA
  - Hash is a full domain hash: $\{0,1\}^* \rightarrow Z^*_n$
- To verify signature s on message m:
  $s^e \bmod n = (hash(m)^d)^e \bmod n = hash(m)$
  - Verification and encryption are the same mathematical operation in RSA
- Message must be hashed and padded (why?)

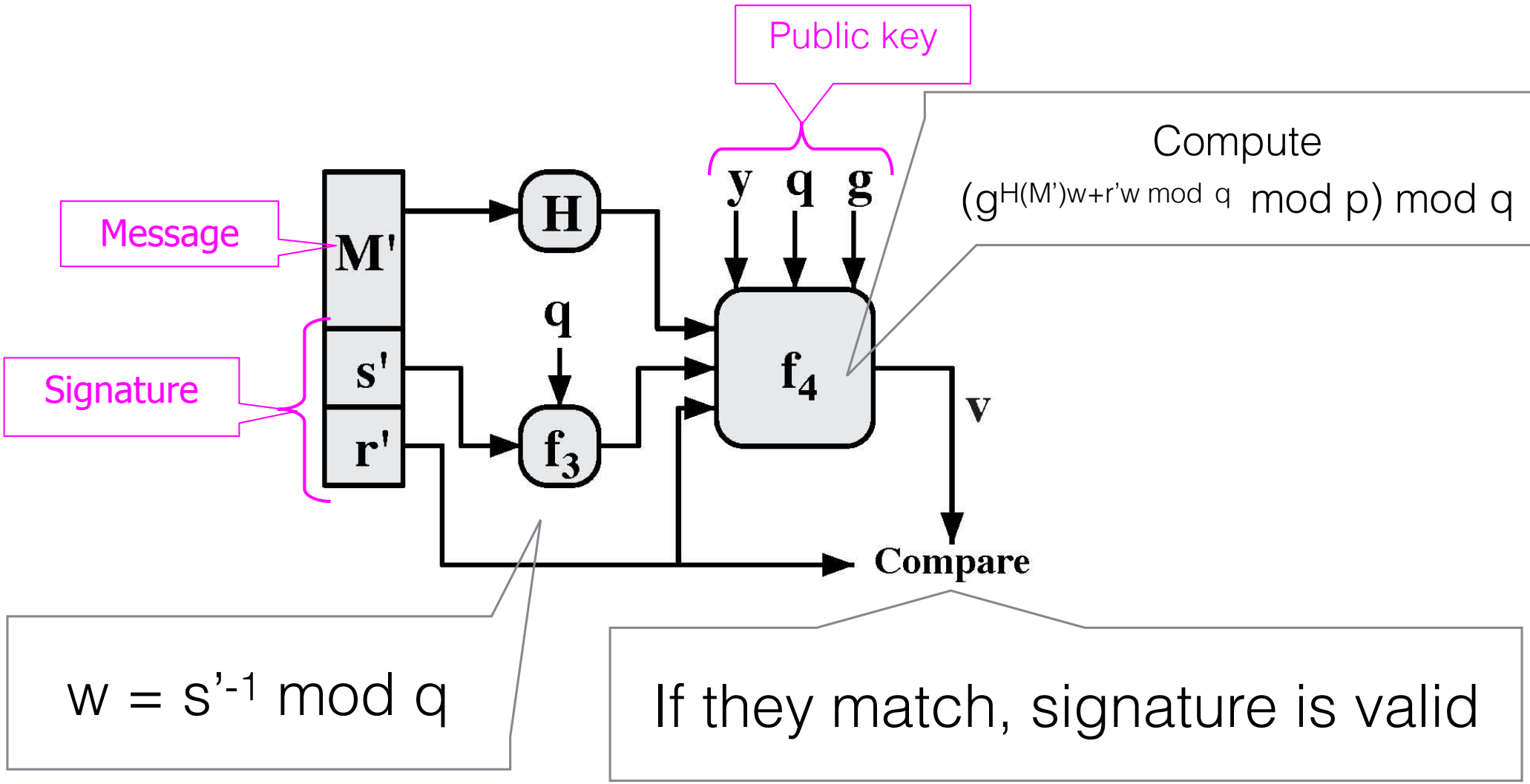
Digital Signature Algorithm (DSA)

- U.S. government standard (1991-94)
  - Modification of the ElGamal signature scheme (1985)
- Key generation:
  - Generate large primes p, q such that q divides p-1
    - $2^{159} < q < 2^{160}$, $2^{511+64t} < p < 2^{512+64t}$ where $0 \le t \le 8$
  - Select $h \in Z_p^*$ and compute $g = h^{(p-1)/q} \bmod p$
  - Select random x such $1 \le x \le q-1$, compute $y = g^x \bmod p$
- Public key: (p, q, g, y), private key: x
- Security of DSA requires hardness of discrete log
  - If one can take discrete logarithms, then can extract x (private key) from y in the public key

DSA: Signing a Message



DSA: Verifying a Signature



Why DSA Verification Works

If (r,s) is a valid signature, then

$r \equiv (g^k \bmod p) \bmod q \;;\; s \equiv k^{-1} \cdot (H(M) + x \cdot r) \bmod q$

Thus $H(M) \equiv -x \cdot r + k \cdot s \bmod q$

Multiply both sides by $w = s^{-1} \bmod q$

$H(M) \cdot w + x \cdot r \cdot w \equiv k \bmod q$

Exponentiate g to both sides

$(g^{H(M) \cdot w + x \cdot r \cdot w} \equiv g^k) \bmod p \bmod q$

In a valid signature, $g^k \bmod p \bmod q = r$, $g^x \bmod p = y$

Verify $g^{H(M) \cdot w} \cdot y^{r \cdot w} \equiv r \bmod p \bmod q$

Security of DSA

- Can't create a valid signature without private key
- Can't change or tamper with signed message
- If the same message is signed twice, signatures are different
  - Each signature is based in part on random secret k
- Secret k must be different for each signature!
  - If k is leaked or if two messages re-use the same k, attacker can recover secret key x and forge any signature from then on

PS3 Epic Fail

- Sony uses ECDSA algorithm to sign authorized software for Playstation 3
  - + Basically, DSA based on elliptic curves
    - … with the same random value in every signature
- Trivial to extract master signing key and sign any homebrew software – perfect “jailbreak” for PS3
- Announced by George “Geohot” Hotz and Fail0verflow team in Dec 2010

Q: Why didn’t Sony just revoke the key?