Digital Signature

Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

Audio/Video recordings of this lecture are available at:

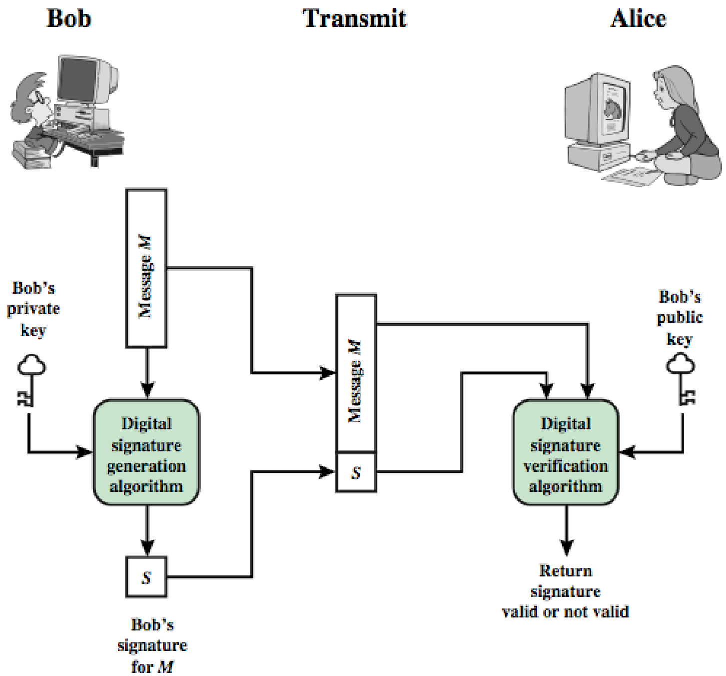
http://www.cse.wustl.edu/~jain/cse571-14/

1. Digital Signatures
2. ElGamal Digital Signature Scheme
3. Schnorr Digital Signature Scheme
4. Digital Signature Standard (DSS)

These slides are based partly on Lawrie Brown's slides supplied with William Stallings's book "Cryptography and Network Security: Principles and Practice," 6th Ed, 2013.

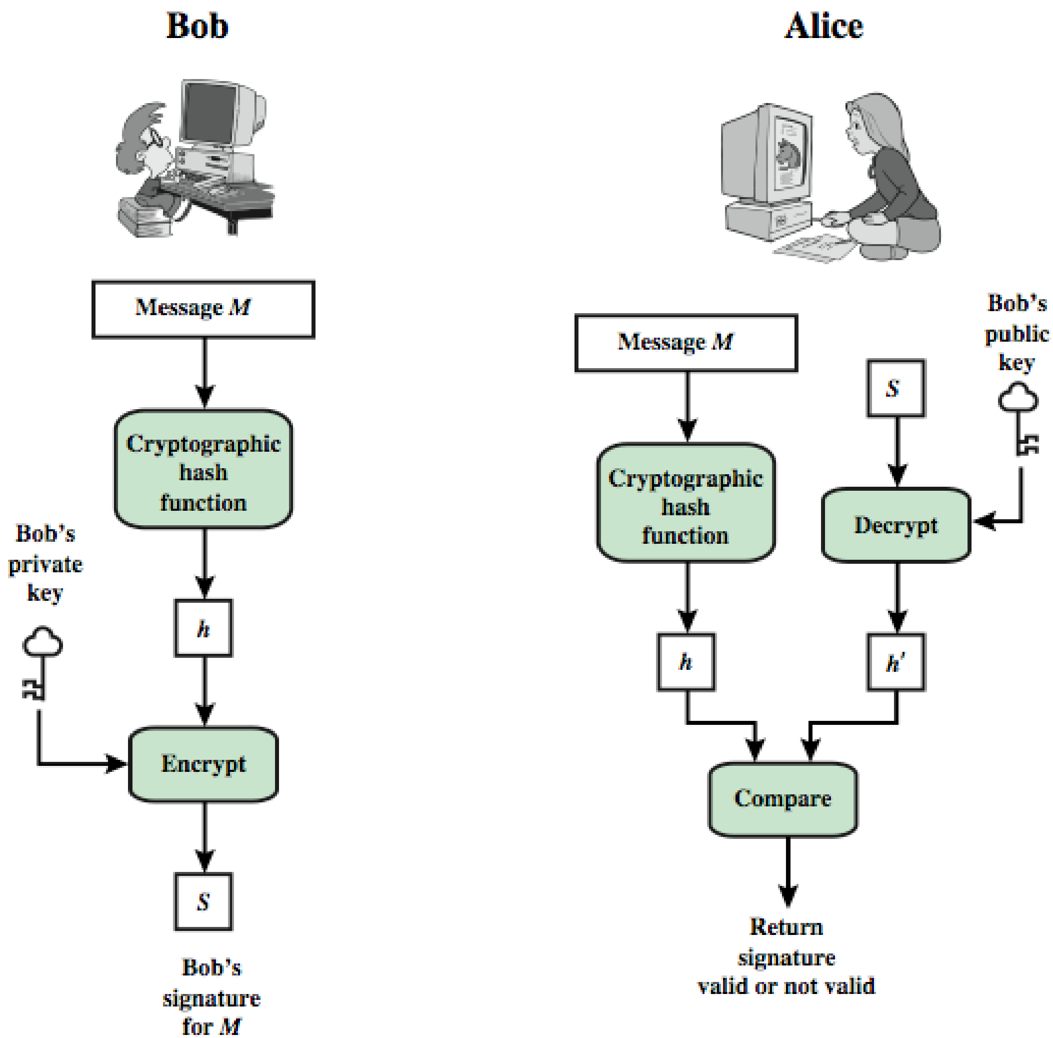
Digital Signatures

- Verify author, date & time of signature
- Authenticate message contents
- Can be verified by third parties to resolve disputes



Ref: http://en.wikipedia.org/wiki/Non-repudiation, http://en.wikipedia.org/wiki/Digital_signature, http://en.wikipedia.org/wiki/Digital_signatures_and_law

Digital Signature Model



Attacks

- In the order of Increasing severity.
- C=Attacker, A=Victim

1. **Key-only attack**: C only knows A's public key
2. **Known message attack**: C has a set of messages, signatures
3. **Generic chosen message attack**: C obtains A's signatures on messages selected without knowledge of A's public key
4. **Directed chosen message attack**: C obtains A's signatures on messages selected after knowing A's public key
5. **Adaptive chosen message attack**: C may request signatures on messages depending upon previous message-signature pairs

Forgeries

1. **Total break**: C knows A's private key
2. **Universal forgery**: C can generate A's signatures on any message
3. **Selective forgery**: C can generate A's signature for a particular message chosen by C
4. **Existential forgery**: C can generate A's signature for a message not chosen by C

Digital Signature Requirements

- Must depend on the message signed
- Must use information unique to sender
  - To prevent both forgery and denial
- Must be relatively easy to produce
- Must be relatively easy to recognize & verify
  Directed $\Rightarrow$ Recipient can verify
  Arbitrated $\Rightarrow$ Anyone can verify
- Be computationally infeasible to forge
  - With new message for existing digital signature
  - With fraudulent digital signature for given message
- Be able to retain a copy of the signature in storage

Ref: http://en.wikipedia.org/wiki/Electronic_signature

ElGamal Digital Signatures

- Signature variant of ElGamal, related to D-H
  - Uses exponentiation in a finite (Galois) field
  - Based on difficulty of computing discrete logarithms, as in D-H
- Each user (e.g., A) generates his/her key
  - Given a large prime *q* and its primitive root *a*
  - A chooses a private key: $1 < x_A < q-1$
  - A computes his **public key**: $y_A = a^{x_A} \bmod q$

Ref: http://en.wikipedia.org/wiki/ElGamal_signature_scheme

ElGamal Digital Signature

- Alice signs a message M to Bob by computing
  - Hash $m = H(M)$, $0 \le m \le (q-1)$
  - Choose random integer K with $1 \le K \le (q-1)$ and $\gcd(K, q-1)=1$ (K is the per message key)
  - Compute $S_1 = a^K \bmod q$
  - Compute K^{-1} the inverse of $K \bmod (q-1)$
  - Compute the value: $S_2 = K^{-1}(m - x_A S_1) \bmod (q-1)$
  - If S_2 is zero, start with a new k
  - Signature is: (S_1, S_2)
- Any user B can verify the signature by computing
  - $V_1 = a^m \bmod q$
  - $V_2 = y_A^{S1}\ S_1^{S2} \bmod q$
  - Signature is valid if $V_2 = V_1$

$$\left(a^{x_A}\right)^{S_1}\left(a^K\right)^{S_2} = a^{x_A S_1 + K S_2} = a^{x_A S_1 + m - x_A S_1} = a^m$$

ElGamal Signature Example

- GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=16$ & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash $m=14$ as $(3,4)$:
  - Choosing random $K=5$ which has $\gcd(18,5)=1$
  - Computing $S_1 = 10^5 \bmod 19 = 3$
  - Finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - Computing $S_2 = 11(14-16\times3) \bmod 18 = 4$
- Any user B can verify the signature by computing
  - $V_1 = a^m \bmod q = 10^{14} \bmod 19 = 16$
  - $V_2 = y_A^{S1} S_1^{S2} \bmod q = 4^3 \times 3^4 = 5184 \bmod 19 = 16$
  - Since $16 = 16$ signature is valid

Schnorr Digital Signatures

- Also uses exponentiation in a finite (Galois) field
- Minimizes message dependent computation
  - Main work can be done in idle time
- Using a prime modulus p
  - $p-1$ has a prime factor q of appropriate size
  - typically p 1024-bit and q 160-bit (SHA-1 hash size)
- Schnorr Key Setup: Choose suitable primes p, q
  - Choose a such that $a^q = 1 \bmod p$
  - (a,p,q) are global parameters for all
  - Each user (e.g., A) generates a key
  - Chooses a secret key (number): $0 < s < q$
  - Computes his **public key**: $v = a^{-s} \bmod q$

Schnorr Signature

- User signs message by
  - Choosing random r with $0<r<q$ and computing $x = a^r \bmod p$
  - Concatenating message with x and hashing:
    $$e = H(M \,||\, x)$$
  - Computing: $y = (r + se) \bmod q$
  - Signature is pair (e, y)
- Any other user can verify the signature as follows:
  - Computing: $x' = a^y v^e \bmod p$
  - Verifying that: $e = H(M \,||\, x')$
  - $x' = a^y v^e = a^y a^{-se} = a^{y-se} = a^r = x \bmod p$

Ref: http://en.wikipedia.org/wiki/Schnorr_signature

Digital Signature Standard (DSS)

- US Govt approved signature scheme
- Designed by NIST & NSA in early 90's
- Published as FIPS-186 in 1991
- Revised in 1993, 1996 & then 2000
- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
- DSA is digital signature only

Ref: http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

Digital Signature Algorithm (DSA)

- Creates a 320 bit signature
- With 512-1024 bit security
- Smaller and faster than RSA
- A digital signature scheme only
- Security depends on difficulty of computing discrete logarithms
- Variant of ElGamal & Schnorr schemes

DSA Key Generation

- Shared global public key values (p, q, g):
  - Choose 160-bit prime number q
  - Choose a large prime p with $2^{L-1} < p < 2^L$
    - Where L= 512 to 1024 bits and is a multiple of 64
      Now extended to 2048 or 3072 bits
    - Such that q is a 160 bit prime divisor of $(p-1)$
  - Choose $g = h^{(p-1)/q}$
    - Where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
      Commonly h=2 is used
- Users choose private & compute public key:
  - Choose random private key: $x<q$
  - Compute public key: $y = g^x \bmod p$

DSA Signature Creation

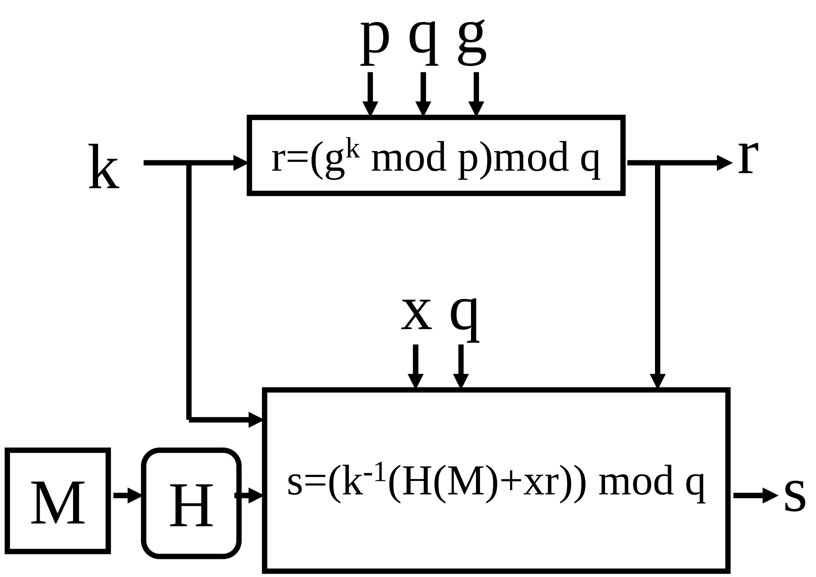
- To **sign** a message `M` the sender:
  - Generates a random signature key `k, k<q`
  - Note: `k` must be random, be destroyed after use, and never be reused
- Then computes signature pair:

  $r = (g^k \bmod p) \bmod q$

  `if r=0 choose another k`

  $s = [k^{-1}(H(M) + xr)] \bmod q$

  `if s=0 choose another k`
- Sends signature `(r,s)` with message `M`



DSA Signature Verification

- To **verify** a signature, recipient computes:

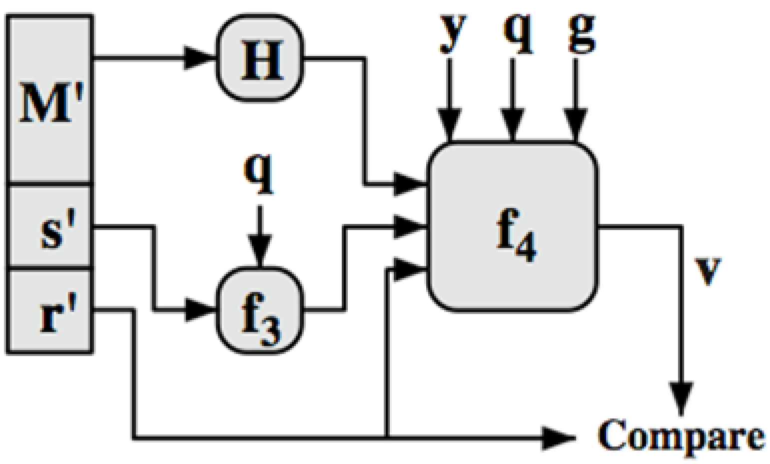
  w = s^{-1} mod q

  u1= [H(M)w]mod q

  u2= (rw)mod q

  v = [(g^{u1} y^{u2})mod p]mod q

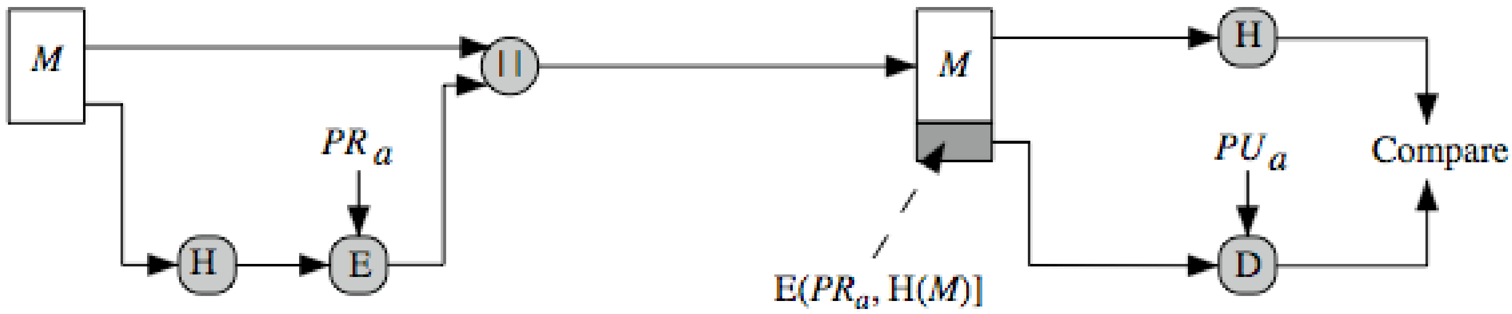
- If v=r then signature is verified



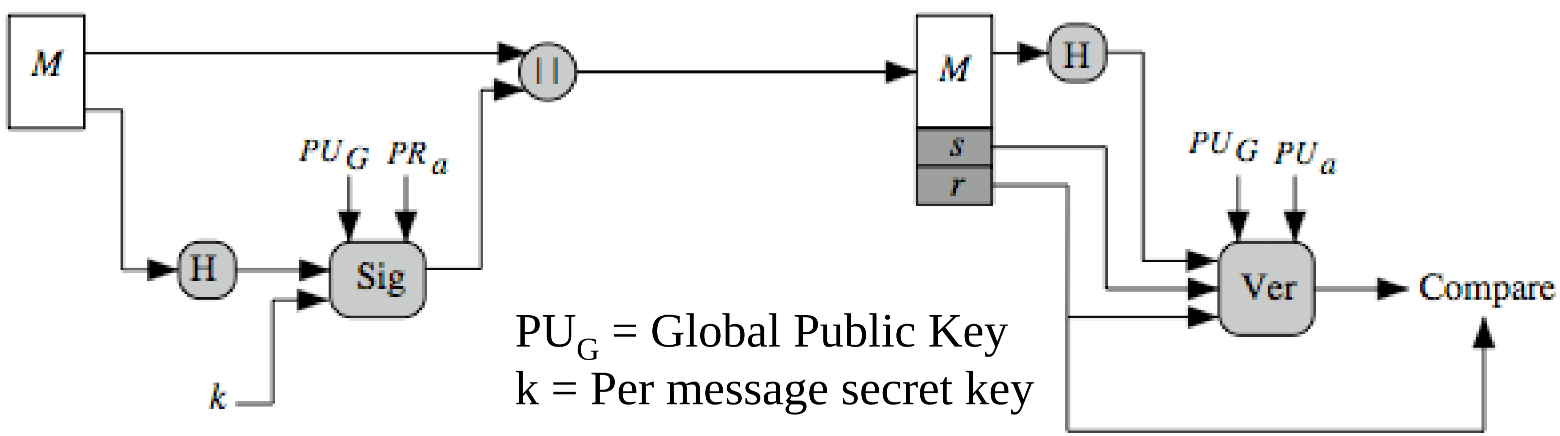
$$w = f_3(s', q) = (s')^{-1} \bmod q$$

$$v = f_4(y, q, g, H(M'), w, r')$$
$$= ((g^{(H(M')w) \bmod q}\, y^{r'w \bmod q}) \bmod p) \bmod q$$

DSS vs. RSA Signatures



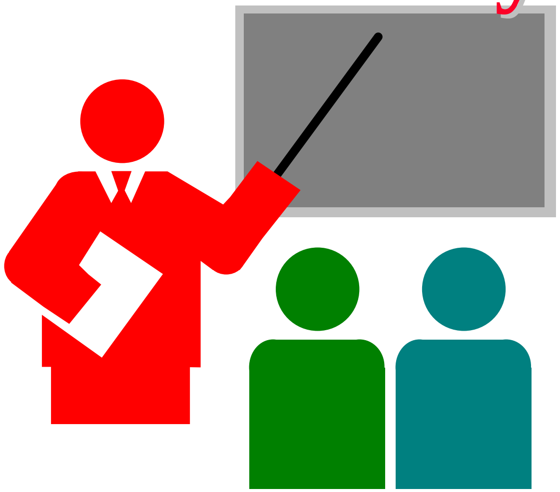
(a) RSA Approach



PU_G = Global Public Key
k = Per message secret key

(b) DSS Approach

Summary



1. Digital signature depends upon the message and some information unique to the signer to prevent forgery and denial. Anyone should be able to verify.
2. ElGamal/Schnorr/DSA signatures use a per-message secret key and are based on exponentiation
3. DSA produces a 320 bit signature

Homework 13

- DSA specifies that if signature generation process results in a value of s=0, a new value of k should be generated and the signature should be recalculated. Why?
- Suppose Alice signed a message M using DSA with a specific k value and then the k value was compromised. Can Alice still use her private key for future digital signatures?
- Hint: Show that the private key of the signer can be easily computed in both of the above cases.