6.915 Computer and Network Security	**Fall Term, 1995**
Lecture 6 : 9-27-1995	
Lecturer: Ron Rivest	*Scribe: Hai Po Sun*

1 Outline

Readings: Pfleeger, Chapter 3

Handouts: Sections 13.3, 13.5, 14.1, and 14.5 from Schneier's book, "Applied Cryptography"

Today's topics:

1. Signatures with hashing
   - MD5
2. Specific public-key systems:
   - RSA
   - ElGamal
   - DSS
3. Secret Sharing

2 Digital signatures with hash functions

Recall from last lecture: We can let signature function be plain encryption where $Signature(S_A, M) = E(S_A, M)$. [$S_A$ is the secret key of A, and S_B is the secret key of B.]

The problem with this naive scheme is that the signature is as long as the original message. To solve this problem, we can use a hash function H to shorten the "message" that we have to sign.

Hash functions map arbitrarily long messages M to a short "message digest" or "fingerprint" $H(M)$ such that are $H(M)$ is unique to the message M. Since H is a "Fixed function", we don't need any key to produce the value of $H(M)$, we can make H perform extremely fast in the actual implementation.

Desirable Properties of the function H:

- The hash value has a fixed length. (say 128 or 160 bits).
- It should be computationally infeasible to find two messages M and M' such that $H(M) = H(M')$. (Collision–resistant)
- Now we can sign $H(M)$ rather than signing the M itself. Recipient can re-compute $H(M)$ (The hash function H is publicized, and we know M from transmission) and check if he/she gets $H(M)$ back after decryption.
- Collision–resistance. This can prevent adversary from asking you to sign M, and then claiming you signed M'.
- Some example of signature function with hashing are MD5 and SHS.

2.1 Message Digest 5 (MD5)

MD5 was originally designed and optimized for 32-bit processors. It uses four 32-bit registers (A,B,C,D), each was initialized with some magic constants. The return value of this digest function will be the hash value $H(M)$ that satisfies all the property of a correct hash functions.

To prepare the message M for MD5, we append a single 1 to M and then concatenate enough 0's until message is 64 bits shy of being a multiple of 512 bits long.

Now we append a 64-bit represenation of the message length to the end of the message and show the length of the message. We process each 512-bit (16-word) block to update registers, and for each block we do the following:

- save A, B, C, D (as A_A, B_B, C_C, D_D)
- repeat four "rounds" to update A, B, C, D
- add A_A back to A, B_B back into B, $\cdots$, D_D back into D.

- Each round processes all of the 16 words, and updates each register 4 times. Typical update would be:

  $$a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s)$$

  where

  - $a <<< s$ means a left-rotated by s bits
  - $F(b, c, d) = (b \wedge c) \vee ((\neg b) \wedge d)$
  - $X[k]$ = a message word
  - $T[i]$ = a magic constant

Note: handout from Schneier's book has a missing + and misdefines <<<. See the class web page for the complete paper on MD5 with the correct definitions.

2.2 Secure Hash Signature [SHS or SHA, Secure Hash Algorithm]

SHS/SHA is similar to MD5, but it is more complicated and slower (about 0.2 Mbytes/sec on a SPARCstation).

3 Specific public-key systems

The idea behind a public-key system is that it might be possible to find a cyrptosystem where it is computationally infeasible to determine the decryption key given the encryption key. If this is so, then the encryption key can be made public by publishing it in a directory.

The advantage of a public-key system is that Alice (or anyone else) can send an encrypted message to Bob (without the prior communication of a secret key) by using the public key. Bob will be the only person that can decrypt the cipher text, using his secret key.

The following are some different public crytosystems.

3.1 RSA

The idea here is that it is easy to find two large prime, but extremely hard to factor the product of the two primes. Note that this is a deterministic public-key cryptosystem. For each plain-text and a public key, there is only one cipher-text.

- Key generation:
  - pick large primes p,q and
  - let $n = p \cdot q$
  - $\phi(n) = (p-1) \cdot (q-1)$
  - Where ϕ (phi) is a function that returns the size of the mulitplicative group modulo n.
  - pick e small, relatively prime to $\phi(n)$ (often 3.) This means that we can solve for d in $ed = 1 \pmod{\phi(n)}$
  - public key is then set to $P_K = (n, e)$
  - and the private key is $S_K = d$
- Encryption with public key:
  - Plaintext message M have a length from 0 to $n-1$
  - Ciphertext $C = M^e \pmod{n}$
- Decryption with private key:
  - Ciphertext C have a length from from 0 to $n-1$
  - Plaintext $M = C^d \pmod{n}$
- Proof of correctness:
  - $M^{(p-1)} = 1 \pmod{p}$ [if $M \neq 0 \pmod{p}$; Fermat's theorem]
  - $M^{(\phi(n))} = 1 \pmod{p}$ [since $p-1$ divides $\phi(n)$]
  - $M^{(k \cdot \phi(n))} = 1 \pmod{p}$ [for any k]
  - $M^{(k \cdot \phi(n)+1)} = M \pmod{p}$ [even if $M = 0 \pmod{p}$]
  - $M^{(ed)} = M \pmod{p}$
  - $M^{(ed)} = M \pmod{q}$ [by repeating the above argument modulo q]
  - $M^{(ed)} = M \pmod{n}$

This shows that either key can be used first ...

Notice the homomorphic property:

$$M^e \cdot M'^e = (M \cdot M')^e$$

This is a double-edged sword: It is very useful property in some applications (e.g. Chaum's digital cash), but a security risk in others (signing M and M' separately magically gives recipient signature for $(M \cdot M')$).

3.2 ElGamal [Taher ElGamal, 1985]

The ElGamal Cryptosystem is non-deterministic. This means that there are many valid signatures for any given message. The verification algorithm must be able to accept any of the valid signatures as authentic.

This algorithm is based on the difficulty of the discrete logarithm problem.

- Find a large public prime p,
- and find a generator $g \pmod p$. A generator g is a value such that $g^1, g^2, g^3, \cdots, g^{p-1}$ are all distinct elements in Z_p^*. It is known from number theory that there are many such elements. In order to find one it is necessary to know the factorization of $p-1$.
- Private key x such that $0 \leq x \leq p-1$.
- Public key $y = g^x \pmod p$.
- To encrypt a message M:
  - generate a random $k, 0 \leq k \leq p-1$.
    Note: This is the step that makes ElGamal a non–deterministic cryptosystem.
  - $send(c_1, c_2) = (g^k, M + y^k) \pmod p$
- To decrypt: $M = c_2 - c_1^x \pmod p$.
- To sign a message M:
  - generate a random $k, 0 \leq k \leq p-1, \gcd(k, p-1) = 1$.
  - compute $a = g^k \pmod p$

  - send as $signature(M) = (a, b)$, where $b = (M - ax)/k \pmod{p-1}$

- To verify a signature:
  - $y^a \cdot a^b \stackrel{?}{=} g^M \pmod{p}$
  - $g^{(ax+kb)} \stackrel{?}{=} g^M \pmod{p}$
  - $M \stackrel{\surd}{=} ax + kb \pmod{p-1}$

3.3 Digital Signature Standard [DSS, proposed 1991 by NIST]

When the DSS was proposed in 1991, there were several criticisms from the general public. The main complaint was that the selection process by NIST was not public. The standard was developed by the National Security Agency (NSA) without the input of U.S. industry. This is also a non-deterministic signature scheme and notice that cannot be used for encryption (however it is possible to perform RSA or ElGamal encryptions given an algorithm for DSS as a black–box)

- Pick a large public prime p (say 1024 bits) such that $p-1$ has a prime factor q of 160 bits.

- g = a public element of order q modulo p

$$g^q = 1 \pmod{p} \text{ and } g^i \neq 1 \text{ if } 0 \leq i < q$$

- Private key: x such that $0 \leq x \leq q-1$

- Public key: $y = g^x \pmod{p}$

- To sign a message m:
  - pick a random k such that $1 \leq k \leq q-1$
  - generate $r = (g^k \pmod{p}) \pmod{q}$
  - generate $s = (H(m) + xr)/k \pmod{q}$
  - let $Signature(m) = (r, s)$

- To verify we check that

$$(g^{(H(m)/s)} y^{(r/s)} \pmod{p}) \pmod{q} \stackrel{?}{=} g^M r$$

3.4 Other proposals

There are many other proposals (IDEA ...), some of which have been broken. Read literature (e.g. start from Schneier's book and follow pointers ...)

4 Secret-sharing [Shamir, 1979]

How to manage many secret keys? We need a mechanism to control the secret keys and allow the secret keys to be revealed under certain kind of conditions.

Suppose if we want any two of top four officers to be able to sign for a company check. In this case, they will "share" a secret key S_C, and can recreate the secret key if any two of them get together and present their own "shares".

Shamir proposed the following technique to share a secret s among n trustees in such a way that at least $d+1$ are needed in order to reconstruct the secret, but d or less have no information about it.

- pick a prime p
- and a polynomial $f(x) = a_d x^d + \cdots + a_0$
- where $a_0 = f(0) = s$ and the other coefficients are random chosen in Z_p.
- the share of trustee i is $share(i) = f(i)$ that is the polynomial function evaluated at i.
- Condition for reconstruction:
  any $d+1$ parties can interpolate to obtain f and a_0 while any d or fewer cannot and have no information about s since any free term is possible given d or less points on a polynomial of degree d.

Notice that this technique is not quite enough for the problem of sharing secret keys since there is no guarantee that the secret being shared has a particular property (as being the matching secret key of a known public key.)