7.3. Cryptographic algorithms

Message M (plaintext, a sequence of bits); key K; published encryption functions E, D; \(\{M\}_K\) is the ciphertext (another sequence of bits)

- **Symmetric (secret key) cryptography**
  \[ E(K, M) = \{M\}_K \quad \text{and} \quad D(K, E(K, M)) = M \]
  
  Same key for E and D
  
  M must be hard (infeasible) to compute if K is not known.
  
  Usual form of attack is brute-force: try all possible key values for a known pair M, \(\{M\}_K\).
  
  Resisted by making K sufficiently large ~ 128 bits

- **Asymmetric (public key) cryptography**
  
  Separate encryption and decryption keys: \(K_e, K_d\)
  
  \[ D(K_d, E(K_e, M)) = M \]
  
  depends on the use of a *trap-door function* (easy to compute in one direction but infeasible to compute its reverse unless a secret is known) to make the keys. E and D have high computational cost. Very large keys > 512 bits

- **Hybrid protocols** - used in SSL (now called TLS)
  
  Uses asymmetric crypto. to transmit the symmetric key, which is then used to encrypt a communication session.
Cipher blocks, chaining cipher blocks

- Most encryption algorithms work on 64-bit blocks.
- Weakness of simple block cipher (blocks are independent)- repeated patterns can be detected.
- Cipher block chaining (CBC): each plaintext block is combined with the preceding ciphertext block using XOR before it is encrypted
- On decryption, the block is decrypted, and then the preceding encrypted block is XOR-ed with it to obtain the new plaintext block
  - It works because (A XOR B) XOR B = A.
- On encryption, $C_{n+1} = E(K, M) = E(K, (p_{n+1} \ XOR \ c_n))$; after decryption, $(p_{n+1} \ XOR \ c_n)$ is obtained, and $(p_{n+1} \ XOR \ c_n) \ XOR \ c_n = p_{n+1}$

Figure 7.6 Cipher block chaining (CBC)
CBC improvement: initialization vector

- CBC introduces dependency between blocks, and is intended to prevent identical portions of plaintext encrypting to identical pieces of ciphertext.
- Possible weakness: if send same messages to two destinations, the encrypted sequences of blocks will be the same, and eavesdropper might gain useful info.
- Solution: to insert a different piece of plaintext in front of each message, called initialization vector (usually timestamp). So, even two identical plaintexts will result in different ciphertexts.
Design of cryptographic algorithms

- All cryptographic alg. rely on (1) information-preserving manipulations of $M$, making use of confusion and diffusion to conceal the content of a ciphertext block $M$; (2) combining it with a key $K$ of sufficient size to render it proof against brute-force attacks.

- Confusion and diffusion
  - Confusion: non-destructive operations such as XOR and circular shifting are used to combine each block of plaintext with the key, producing a new bit pattern that obscures the relationship between the blocks in $M$ and $\{M\}_K$.
  - Diffusion: there is usually repetition and redundancy in the plaintext. Diffusion dissipates the regular patterns that result by transposing portions of each plaintext block.
More on confusion and diffusion

- In cryptography, **confusion** and **diffusion** are two properties of the operation of a secure cipher which were identified by Shannon in his paper, "Communication Theory of Secrecy Systems" published in 1949.

- In Shannon's original definitions, **confusion** refers to making the relationship between the key and the ciphertext as complex and involved as possible; **diffusion** refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext.

- **Diffusion** is associated with dependency of bits of the output on bits of the input. In a cipher with good **diffusion**, flipping an input bit should change each output bit with a probability of one half (this is termed the Strict Avalanche Criterion).

- **Substitution** (a plaintext symbol is replaced by another) has been identified as a mechanism for primarily **confusion**; conversely **transposition** (rearranging the order of symbols) is a technique for **diffusion**.
More on confusion and diffusion

• In cryptography, a substitution cipher is a method of encryption by which units of plaintext are substituted with ciphertext according to a regular system; the "units" may be single letters, pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

• Substitution ciphers can be compared with transposition ciphers. In a transposition cipher, units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.
Symmetric encryption algorithms

These are all programs that perform confu. and diffu. operations on blocks of binary data:

**TEA**: a simple but effective algorithm developed at Cambridge U (1994) for teaching and explanation. 128-bit key, 700 kbytes/sec

   - No longer strong in its original form. In 1997, successfully cracked in a brute-force attack (a competition to demonstrate the lack of security of encryption with keys shorter than 128 bits), the attack took about 12 weeks with tens of thousands PCs involved (coordinated by a single server)
   - **Triple-DES**: applies DES three times with two different keys. 112-bit key, 120 Kbytes/sec


There are many other effective algorithms. See Schneier [1996].

*The above speeds are for a Pentium II processor at 330 MHZ. Today's PC's (2005) should achieve a 10 x speedup.*
## Symmetric encryption algorithms

<table>
<thead>
<tr>
<th>Key size (bits)</th>
<th>Key space size</th>
<th>Mean time required at 1 key test/(\mu)sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>(2^{32} = 4.3 \times 10^9)</td>
<td>35.8 minutes</td>
</tr>
<tr>
<td>56 (DES)</td>
<td>(2^{56} = 7.2 \times 10^{16})</td>
<td>1,142 years</td>
</tr>
<tr>
<td>128</td>
<td>(2^{128} = 3.4 \times 10^{38})</td>
<td>(5.4 \times 10^{24} = 300) billion big bangs</td>
</tr>
<tr>
<td>168</td>
<td>(2^{168} = 3.7 \times 10^{50})</td>
<td>(5.9 \times 10^{36}) big bangs</td>
</tr>
</tbody>
</table>
TEA encryption function

```c
void encrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = 0; int n;
    for (n = 0; n < 32; n++) {
        sum += delta;
        y += ((z << 4) + k[0]) ^ (z+sum) ^ ((z >> 5) + k[1]);
        z += ((y << 4) + k[2]) ^ (y+sum) ^ ((y >> 5) + k[3]);
    }
    text[0] = y; text[1] = z;
}
```

- Lines 5 & 6 perform confusion (XOR of shifted text) and diffusion (shifting and swapping)
TEA decryption function

```c
void decrypt(unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = delta << 5; int n;
    for (n = 0; n < 32; n++) {
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    text[0] = y; text[1] = z;
}
```
void tea(char mode, FILE *infile, FILE *outfile, unsigned long k[]) {
/* mode is 'e' for encrypt, 'd' for decrypt, k[] is the key. */
    char ch, Text[8]; int i;
    while(!feof(infile)) {
        i = fread(Text, 1, 8, infile); /* read 8 bytes from infile into Text */
        if (i <= 0) break;
        while (i < 8) { Text[i++] = ' ';} /* pad last block with spaces */
        switch (mode) {
        case 'e':
            encrypt(k, (unsigned long*) Text); break;
        case 'd':
            decrypt(k, (unsigned long*) Text); break;
        }
        fwrite(Text, 1, 8, outfile); /* write 8 bytes from Text to outfile */
    }
}
Asymmetric encryption algorithms

• Only a few practical public-key schemes have been developed to date. They all depend on the use of *trap-door functions*

  – A trap-door function is a one-way function with a secret exit - e.g. product of two large numbers; easy to multiply, very hard (infeasible) to factorize.

A trapdoor provides a secret way into a room. If you're inside, the way out is obvious, if you're outside, you need to know a secret to get in.
Asymmetric encryption algorithms

RSA: The first practical algorithm (Rivest, Shamir and Adelman 1978) and still the most frequently used. Key is usually in the range of 512-2048 bits. Speed 1-7 kbytes/sec. (350 MHz PII processor)

Elliptic curve: A recently-developed method, shorter keys and faster.

Asymmetric algorithms are ~1000 x slower and are therefore not practical for bulk encryption, but their other properties make them ideal for key distribution and for authentication uses – initial stage of secure communication stages.
RSA Encryption

Encryption: \( C = P^e \mod N \)
Decryption: \( P = C^d \mod N \)

\[ K_e = (e, N), K_d = (d, N) \]

\[ N = 55, \ e = 7, \ d = 23 \]

“RSA” = “18, 19, 1”

\[ C_1 = 18^7 \mod 55 = 17 \]
\[ C_2 = 19^7 \mod 55 = 24 \]
\[ C_3 = 17 \mod 55 = 1 \]

\[ P_1 = 17^{23} \mod 55 = 18 \]
\[ P_2 = 24^{23} \mod 55 = 19 \]
\[ P_3 = 1^{23} \mod 55 = 1 \]

\[ N = P \times Q \ (P = 5, \ Q = 11 \text{ in previous example}) \]

Choose decryption key \( d \) s.t. \( Z = (P - 1) \times (Q - 1) \) are relatively prime \((d=23 \text{ and } Z=40)\)

Compute encryption key \( e \) s.t. \( e \times d = 1 \mod Z \) \((e = 7; \ 7 \times 23 = 161 = 4Z + 1)\)

\[ \bullet \text{That is, } e \times d \text{ is the smallest element divisible by } d \text{ in the series } Z+1, 2Z+1, 3Z+1, ... \]
RSA Encryption - 1

To find a key pair $e, d$:
1. Choose two large prime numbers, $P$ and $Q$ (each greater than $10^{100}$), and form:
   \[ N = P \times Q \]
   \[ Z = (P-1) \times (Q-1) \]
2. For $d$ choose any number that is relatively prime with $Z$ (that is, such that $d$ has no common factors with $Z$).
   We illustrate the computations involved using small integer values for $P$ and $Q$:
   \[ P = 13, \ Q = 17 \\rightarrow \ N = 221, \ Z = 192 \]
   \[ d = 5 \]
3. To find $e$ solve the equation:
   \[ e \times d = 1 \mod Z \]
   That is, $e \times d$ is the smallest element divisible by $d$ in the series $Z+1, 2Z+1, 3Z+1, \ldots$ .
   \[ e \times d = 1 \mod 192 = 1, 193, 385, \ldots \]
   $385$ is divisible by $d$
   \[ e = 385/5 = 77 \]

Question: $e$ in step 2 and $d$ in step 3?
Hybrid cryptographic protocols

- Public-key cryptography is convenient for E-commerce
  - no need for a secret key distribution mechanism
- But, processing cost too high
  - for safety, 768-bit key or greater
- Common approach in large-scale distributed system: a hybrid scheme
  - public-key cryptography is used to authenticate the parties and to encrypt an exchange of secret keys, which are used for subsequent communication
  - e.g. TLS (transport layer security protocol)
7.4. Digital signatures

- Strong digital signatures are essential for secure systems

- Requirement:
  - To authenticate stored document files as well as messages
  - To protect against forgery
  - To prevent the signer from repudiating a signed document (denying their responsibility)

- Encryption of a document in a secret key constitutes a signature
  - impossible for others to perform without knowledge of the key
  - strong authentication of document
  - strong protection against forgery
  - weak against repudiation (signer could claim key was compromised)

- \([M]_K\): Message \(M\) signed with key \(K\)
Digital signing

- $M, A, [M]_{K_A}$ (Message + identifier + encrypted M)

- If a secret key is used to encrypt the document, only principals that share the secret can verify the signature

- For public key cryptography, signer uses her private key and anyone who has the corresponding public key can verify the signature
  - A better analogue for conventional signatures

- Encrypted text of document makes an impractically long signature
  - So we encrypt a secure digest instead
Digest functions (secure hash functions)

- A digest function computes a fixed-length hash $H(M)$ that characterizes the document $M$. $H(M)$ should be:
  - fast to compute
  - hard to invert - hard to compute $M$ given $H(M)$ (one-way hash function)
  - hard to defeat in any variant of the Birthday Attack

  - Speed 1740 kbytes/sec. one of the most efficient

- **SHA**: (1995) based on Rivest's MD4 but made more secure by producing a 160-bit digest, speed 750 kbytes/second

- Any symmetric encryption algorithm can be used in CBC (cipher block chaining) mode. The last block in the chain is $H(M)$
  - Need not be information preserving since not intended to be reversible
  - Can use any bit-wise logical operations.
Digest functions to ensure message integrity

Received msg: $m$, $\text{MD5}(m)$

$\text{MD5}(m)$ → Compare

Ensures $m$’s integrity

Question: why not just decrypt $\text{MD5}(m)$?

Because digital digest functions are not reversible
Digest functions combine with cryptography

- Make it tamper proof using $K_{Apub}$ and $K_{Apriv}$

Received msg:

\[
\begin{array}{cc}
\text{m} & \{ \text{MD5(m)} \}^{\text{KApriv}} \\
\end{array}
\]

\[
D(K_{Apub}, \{ \text{MD5(m)} \}^{\text{KApriv}})
\]

\[
\text{MD5(m)} \rightarrow \text{Compare}
\]
Digital signatures with public keys

MD5 with RSA signature

• Sender Alice (msg \( m \))
  – Encrypt msg \( \text{MD5}(m) \) by A’s private key, and send
    \( m + K_{\text{Apriv}}\{ \text{MD5}(m) \} \)

• Receiver Bob
  – Compute \( \text{MD5}(m) \) using \( m \).
  – Decrypt signature with A’s public key, i.e., apply \( K_{\text{Apub}}( ) \) to \( K_{\text{Apriv}}\{ \text{MD5}(m) \} \) to extract \( \text{MD5}(m) \)
  – Compare the two
Digital signatures with public keys

A generates $K_{pub}$ and $K_{priv}$, Publish $K_{pub}$ by placing it in a well-known location

**Signing by A**

$$M \xrightarrow{H(M)} h \xrightarrow{E(K_{pri}, h)} \{h\}_{K_{pri}}$$

**Verifying by B**

$$M \xrightarrow{H(doc)} h \xrightarrow{D(K_{pub}, \{h\})} h'$$

$h = h'$? authentic: forged
MAC: Low-cost signatures with a shared secret key

MAC: Message Authentication Code

- Signer must arrange for verifier to receive the secret key
- In case a secure channel is used to transmit unencrypted messages but there is a need to verify authenticity of messages.

- A generates random K and distribute it via secure channel
- A computes the digest h of \( M + K \) and sends \([M]_K = M, h\)
  - \( M + K \) is the concatenation
  - h is a MAC
- B concatenates K with M and computes \( H(M+K) \), compares with \( h \)

\[ h = h'? \]

authentic: forged
### 7.5. Cryptography pragmatics

#### Performance of encryption and secure digest algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Key size/hash size (bits)</th>
<th>Extrapolated speed (kbytes/sec.)</th>
<th>PRB optimized speed (kbytes/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secret key</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEA</td>
<td>128</td>
<td>700</td>
<td>-</td>
</tr>
<tr>
<td>DES</td>
<td>56</td>
<td>350</td>
<td>7746</td>
</tr>
<tr>
<td>Triple-DES</td>
<td>112</td>
<td>120</td>
<td>2842</td>
</tr>
<tr>
<td>IDEA</td>
<td>128</td>
<td>700</td>
<td>4469</td>
</tr>
<tr>
<td><strong>Public key</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSA</td>
<td>512</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>RSA</td>
<td>2048</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td><strong>Digest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>1740</td>
<td>62425</td>
</tr>
<tr>
<td>SHA</td>
<td>160</td>
<td>750</td>
<td>25162</td>
</tr>
</tbody>
</table>
PGP (Pretty Good Privacy) digital signature

- Created by Philip Zimmermann
- Is the de facto standard program for secure e-mail and file encryption on the Internet, uses IDEA, RSA, and MD5
- Message encryption and decryption
  - E-mail sent over the Internet is more like paper mail on a postcard than mail in a sealed envelope. It can easily be read, or even altered, by anyone with privileged access to any of the computers along the route followed by the mail. Hackers can read and/or forge e-mail. Government agencies eavesdrop on private communications.
- File encryption and decryption
- Digital Signature
How secure is a message against brute force decryption attempts by someone who does not have your private key.

If you choose a sufficiently long key (and complex enough passphrase), then it would take today’s most powerful supercomputers centuries to break the “lock”, except by luck.

That is why the U.S. Government fought so long to prevent the export of versions of PGP that could handle long keys, and harrassed PGP’s creator Phil Zimmerman, one of the people who has truly made a difference on behalf of human freedom.

- Cryptographic software was classified as a munition in the US.
PGP (Pretty Good Privacy) digital signature

• analogous to following situation: I (Bob) am expecting a secured package from you (Alice) so I send you an open *padlock* (public key) to which I have the key (private key). You lock the contents box with the *padlock*, and nobody can unlock it except me.

• Say, in our class, each one generates a priv/pub key pair and publish the public key to others, then the class can exchange secured messages

• Pick random key \( k \), encrypt whole message,

\[
m + K_{\text{Apriv}}\{\text{MD5}(m)\}
\]

and append encrypted \( k \):

\[
k\{ m + K_{\text{Apriv}}\{\text{MD5}(m)\} \} + K_{\text{Bpub}}\{k\}
\]

• Receiver B decrypts \( K_{\text{Bpub}}\{k\} \) first, retrieving \( k \), with which \( m + K_{\text{Apriv}}\{\text{MD5}(m)\} \) can be recovered

• Then B verifies the integrity of \( m \). How?
More on PGP

- man pgp (Unix man page)
- mkdir $HOME/.pgp /*to create keyring
- pgp –kg /*create public/private keys
  - Will prompt for passphrase and random number
- pgp –e text her_userid /*recipient public key used, text.pgp generated
- What’s keyring? What’s passphrase? Find out by yourselves.

- For windows, download pgp 6.5.8 from http://www.pgpi.org/products/pgp/versions/freeware/win32/6.5.8/
Summary

- It is essential to protect the resources, communication channels and interfaces of distributed systems and applications against attacks
- This is achieved by the use of access control mechanisms and secure channels
- Public-key and secret-key cryptography provide the basis for authentication and for secure communication
- Kerberos and SSL are widely-used system components that support secure and authenticated communication
  - Self-reading for interested students