Today

- Secure algorithms
- Block and Stream ciphers
- DES, RC4
- Encryption modes
- Key length
- Cryptographic hash functions
Encryption

• Some cryptographic methods rely on the secrecy of the algorithms
  – Only historical interest
  – Not adequate for real-world applications

• Generally, no algorithm that depends on its secrecy is secure

• All modern algorithms
  – Use keys to control encryption and decryption
  – Cannot really be executed by humans

• In theory, any cryptographic method with a key can be broken by trying all possible keys in sequence
  – Except One-time Pad
Block and Stream ciphers

- Block ciphers operate on blocks of plain-text and cipher-text.
  - Usually 64 or 128 bits
  - The same plain-text block will always encrypt to the same cipher-text block (same key of course)
  - DES, IDEA, AES (Rijndael), Blowfish, Twofish

- Stream ciphers operate on streams of plain-text and cipher-text.
  - Usually 1 bit or 1 byte
  - The same bit or byte will encrypt to a different bit or byte
  - RC4, LFSRs, A5, SEAL
Encryption modes

- Encryption algorithms are seldom used directly, a cryptographic mode is used instead.
- A cryptographic mode usually combines:
  - the basic cipher
  - some sort of feedback
  - some simple operations
Electronic Codebook Mode

- ECB - Electronic Codebook Mode
- Simplest mode: no feedback - the same block always encrypt to the same cipher-text
- Theoretically possible to create a codebook for each key; but for blocks of 64 bit the codebook needs $2^{64}$ entries...
- Statistical attacks possible
CBC - Cipher Block Chaining

• Chaining adds feedback
  – The result of previous blocks are fed back into the encryption of the current block
• The plaintext is XORed with the previous cipher-text block before it is encrypted
CBC Encryption

\[ E_k \]

\[ E_k \]

\[ E_k \]
CBC Decryption

\[ \begin{align*}
Pi-1 &\rightarrow D_k &\rightarrow Ci-1 \\
Pi &\rightarrow D_k &\rightarrow Ci \\
Pi+1 &\rightarrow D_k &\rightarrow Ci+1
\end{align*} \]
Initialization vector

- CBC will encrypt two identical messages to the same cipher-text
- The beginning of cipher-text of messages with constant headers will be the same
- Fix: use an Initialization Vector (IV)
  - Make the first block random
  - Has no value: only there to make all messages unique
  - Can be transmitted over insecure channels
Stream ciphers

- A keystream generator outputs a stream of bits: \( k_1, k_2, \ldots, k_i \)
- The keystream is XORed with a stream of plaintext bits \( p_1, p_2, \ldots, p_i \)
- Cipher-text bits are produced: \( c_i = k_i + p_i \)
- \( + \) denotes the XOR operation
- \( p_i = c_i + k_i \)
- The security depends entirely on the keystream generator
  - If we get repeating 16 bits values, it is a simple XOR cipher that is very easy to break
  - If we get a true random keystream, we have a One-time Pad and perfect security
- Especially useful to encrypt continuous streams of communication data, e.g. a TV-link.
Stream cipher encryption/decryption

Diagram:

- Keystream Generator
- Keystream
- Plaintext
- Ki
- Encrypt
- Ciphertext
- Decrypt
- Plaintext
- Pi
Keystream generator

Key K

Internal state

Next state Function

Output function

Ki
Cipher-Feedback Mode

- One way of implementing a block cipher as a stream cipher is cipher-feedback mode, CFB
- CBC needs a full block until it data can be transmitted
- Network applications that need data to be processed in byte-sized chunks.
- Uses an IV
- Example: 8-bit CFB
CFB Encryption

Shift register

Key K

Encrypt

Left-most byte

Ki

Pi

Ci
CFB Decryption

Shift register

Key K

Encrypt

Left-most byte

Ki

Ci

Pi
Output-Feedback Mode

• Similar to CFB
• Only use a feedback size that is equal to the block size
• The keystream will repeat eventually, when it does OFB is easily broken
  – For 64-bit feedback size and 64 bit block size, the average cycle length is $2^{64}-1$
  – If the feedback size is smaller, the cycle length decreases to $2^{32}$, which is too small
OFB Encryption

- **Key K**
- **Shift register**
- **Encrypt**
- **Left-most byte**
- **Ki**
- **Pi**
- **Ci**
OFB Decryption
New encryption modes

- Integer Counter mode
  - Still not standardized, but will be soon
  - Used in e.g. SRTP
  - Uses a block cipher as a keystream generator
- Combined encryption and authentication modes
Choosing a Cipher Mode

- ECB is the easiest and fastest, but also weakest
- Ok for short random data such as other keys
- For normal plaintext, use CBC, CFB or OFB.
- For files, CBC are almost always the best choice
- CFB, especially 8 bit CFB is best for applications that need streams of characters where each character has to be treated individually
Choosing a Cipher Mode

- **ECB:**
  - **Security**
    - Plaintext patterns are not concealed
    - Input to the block cipher is not randomized; it is the same as the plaintext
    - More than one message can be encrypted with the same key
    - Plaintext is easy to manipulate; blocks can be removed, repeated or interchanged
  - **Fault-tolerance**
    - A ciphertext error affects one full block of plaintext
    - Synchronization error is unrecoverable
Choosing a Cipher Mode

• CBC:
  – Security
    • + Plaintext patterns are concealed by XORing with the previous ciphertext block
    • + Input to the block cipher is randomized
    • + More than one message can be encrypted with the same key
    • +/- Plaintext is somewhat difficult to manipulate; blocks can be removed from the beginning and end of the message, bits of the first block can be changed, and repetition allows some controlled changes
  – Fault tolerance
    • - A ciphertext error affects one full block of plaintext and the corresponding bit in the next block
    • - Synchronization error is unrecoverable
Choosing a Cipher Mode

- **CFB**
  - **Security**
    - + Plaintext patterns are concealed
    - + Input to the block cipher is randomized
    - + More than one message can be encrypted with the same key if a different IV is used
    - +/- Plaintext is somewhat difficult to manipulate; blocks can be removed from the beginning and end of the message, bits of the first block can be changed, and repetition allows some controlled changes
  - **Fault tolerance**
    - - A ciphertext error affects the corresponding bit of plaintext and the next full block
    - + Synchronization errors of full block sizes are recoverable
Choosing a Cipher Mode

- OFB
  - Security
    - + Plaintext patterns are concealed
    - + Input to the block cipher is randomized
    - + More than one message can be encrypted with the same key if a different IV is used
    - - Plaintext is very easy to manipulate; any change in ciphertext directly affects the plaintext

  - Fault tolerance
    - + A ciphertext error only affects the corresponding
Block Cipher Design

- Diffusion and confusion
- Substitution and transposition
- Production ciphers
- Feistel networks
- Don't do it yourself
Confusion and diffusion

• Confusion servers to hide any relationship between the plaintext, ciphertext and the key
  – Output bits must have a complex transformation of the key- and plaintext bits
• Diffusion spreads the influence of individual plain-text or key bits over as much of the ciphertext as possible
• Confusion alone is enough for security: an algorithm with a single key-dependent lookup table of 64 bits of plain-text to 64 bits of cipher is quite secure
  – Requires $10^{20}$ bytes of memory...
• Repeatedly mix confusion and diffusion: a **product cipher**
Feistel networks

- Used in many block ciphers
- Divide each block in two halves, L and R (left and right)
  - \( L_i = R_{i-1} \)
  - \( R_i = L_{i-1} + f(R_{i-1}, K_i) \)
- + denotes XOR
- Used because a cipher that uses this is guaranteed to be invertible
- Doesn't matter what the function \( f \) is
DES

- Data Encryption Standard (DES)
- Also known as Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO
- National Bureau of Standards (NBS, now National Institute of Standards of Technology, NIST) wanted a single standard cryptographic algorithm
- Public request in 1973
Public request

• Design criteria:
  – Must provide a high level of security
  – Must be completely specified and easy to understand
  – The security must reside in the key, not in the secrecy of the algorithm
  – Must be available to all users
  – Must be adaptable for use in diverse applications
  – Must be economically implementable in electronic devices
  – Must be efficient to use
  – Must be able to be validated
  – Must be exportable

• No candidate until IBM's Lucifer in 74-75
DES

- Evaluated by NSA (National Security Agency)
- NSA reduced the key length from 128 to 56 bits
- They changed the inner workings
- Many feared that backdoors were inserted
- Mistake by NSA:
  - They thought that it was for hardware only
  - The publication of DES started a lot of research in cryptography
    - For the first time, an algorithm approved by NSA was available
    - NSA was the world leading authority on cryptography at that time (still is)
DES

- 64 bit blocks
- 56 bit key (64 bits with parity bits)
- 16 rounds
- Feistel network
- Confusion and Diffusion
DES Overview
DES Expansion permutation

- Expanding Ri-1 from 32 bits to 48 bits
- Using some bits more than once
- Hence, the dependency of the output bits on the input bits spread faster
DES Key transformation

• A different 48 bit sub-key is generated for each of the 16 rounds
• First, the 56-bit key is divided into two 28 bit halves
• Then the halves are circularly shifted left by either 1 or 2 bits depending on the round
• 48 bits are then selected from the 56 bits
  – Compression permutation
DES S-Box substitution

- Eight substitution boxes, S-Boxes
- Each S-Box has a 6 bit input and a 4 bit output
- The 48 input bits are divided into 6 bit blocks
- The S-box values are selected very carefully, do not change them!
DES P-Box permutation

- The 32 bit output is permuted according to a P-Box
- Maps all input bits to output bits; no bits are used twice and no bits are ignored
DES Security

- Weak keys:
  - Do not use:
  - 0101010101010101, 1F1F1F1F0E0E0E0E, E0E0E0E0F1F1F1F1, FEFEFEFEFEFEFEFE
- 12 Semi-weak keys
- 16 rounds just enough not to be sensitive to differential cryptoanalysis
- DES Cracker (EFF)
  - Specialized hardware
  - $250,000
  - Brute force attack: try all possible keys (256)
  - Cracked 56 bit DES in 56 hours (July 1998)
- Can use Triple DES (112 bit keys) or other DES variants
- The rumor is that NSA can crack DES in 3-15 minutes with hardware for $50,000
Triple DES

- Also known as
  - EDE-DES, 3DES
- Two keys K1 and K2 are used
- C = EK1(DK2(EK1(M)))
- M = DK1(EK2(DK1(M)))
- If K1 = K2, then it is the same as DES
- Brute force attack needs $2^{112}$ tries
Triple DES
Blowfish

• By Bruce Schneier
• Designed to
  – Be fast
  – Compact
  – Simple
  – Variable secure: keys can be up to 448 bits long
• Feistel network
• Widely used (e.g. standard in OpenBSD)
• 64 bit blocks
• No known attacks
• Unpatented, free
Blowfish
Blowfish
Other algorithms

- Probably secure: 5 finalist of the Advanced Encryption Standard (AES) Development Effort
  - AES: competition run by NIST to develop a new algorithm to replace DES as a standard
  - MARS, RC6, Rijndael, Serpent and Twofish
  - Rijndael selected as the new standard
  - Three key sizes: 128, 192 and 256 bits

- IDEA: slow, might soon be too unsecure, not free
**Key length**

- The security of a symmetric cryptosystem is a function of two things: the strength of the algorithm and the length of the key, i.e. the size of the key-space
- Assume that the strength of the algorithm is perfect
  - That is, there is no better way to crack the system than trying every possible key
- If possible, use a key length of 112 bits or more
- Average time estimates for a hardware brute-force attack in 1995:
Key length

- Average time estimates for a hardware brute-force attack in 1995 (Bruce Scheier):

<table>
<thead>
<tr>
<th>Cost</th>
<th>40</th>
<th>56</th>
<th>64</th>
<th>80</th>
<th>112</th>
<th>128 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100 K</td>
<td>2 s</td>
<td>35h</td>
<td>1y</td>
<td>70000y</td>
<td>10^14y</td>
<td>10^19y</td>
</tr>
<tr>
<td>$1 M</td>
<td>.2 s</td>
<td>3.5h</td>
<td>37 days</td>
<td>7000y</td>
<td>10^13y</td>
<td>10^18y</td>
</tr>
<tr>
<td>$10 M</td>
<td>0.02s</td>
<td>21min</td>
<td>4 days</td>
<td>700y</td>
<td>10^12y</td>
<td>10^17y</td>
</tr>
<tr>
<td>$100 M</td>
<td>2ms</td>
<td>2min</td>
<td>9h</td>
<td>70y</td>
<td>10^11y</td>
<td>10^16y</td>
</tr>
<tr>
<td>$1 G</td>
<td>0.2ms</td>
<td>13s</td>
<td>1h</td>
<td>7y</td>
<td>10^10y</td>
<td>10^15y</td>
</tr>
</tbody>
</table>
Stream ciphers and random number generators

- Random generators are used in cryptography mainly for two things: key stream generators for stream ciphers and to generate encryption keys.
- Pseudo random generators: generate a sequence that looks random.
- Cryptographically secure pseudo random sequences: it is unpredictable.
  - Given complete knowledge of the algorithm and previous bits in the stream, it must be computationally infeasible to predict the next bit.
Simple Random Number Generator

- $X_n = (a \cdot X_{n-1} + b) \mod m$
- $X_n = (a \cdot X_{n-13} + b \cdot X_{n-12} + c \cdot X_{n-1} + d) \mod m$
- Easy to predict
Linear Feedback Shift Registers

- Shift register and a feedback function
LFSR ciphers

- Stream ciphers based on LFSR exist
- Many have been broken
- Example: the cipher used for DVDs, broken by Frank Stevenson
RAND Tables

- Book published by the Rand Corporation in 1955
- Contains a million random digits
- Many arbitrary constants in cryptographic algorithms are chosen from this book
Using Random Noise

- Get randomness from nature
- Might require specialized hardware, but we can play tricks with computers
  - Measure keyboard latency
  - Mouse movements
  - IO operations
  - Number of the current scanline of monitor
  - Contents of the filesystem, kernel tables, etc.
  - CPU load
  - Arrival times of network packets
  - Input from a microphone
RC4

- Developed 1987 by Ron Rivest for RSA Data Security, Inc.
- RC = Ron's Code, or Rivest Cipher
- One S box with 256 entries and one key with 256 entries (bytes)
- Initially: $S[i] = i$
RC4

- First, prepare the S-box:
  for $i = 0$ to $255$:
    $j = (j + S[i] + K) \mod 256$
    swap $S[i]$ and $S[j]$

- To get the next byte:
  1. $i = (i + 1) \mod 256$
  2. $j = (j + S[i]) \mod 256$
  3. Swap $S[i]$ and $S[j]$
  4. $t = (S[i] + S[j]) \mod 256$
  5. $K = S[t]$
RC4

- K is XORed with either the cipher-text or plain-text
- Fast and have no known attacks, so why isn't it used more often?
  - Used to be a trade secret, but someone found out the algorithm
  - People were afraid of law-suits from RSA...
- Stream cipher – so we need to be careful how we use it
  - Implementation in WEP (802.11b, Wavelan, etc...) broken...
Cryptographic hash functions

- One-way hash functions, message digest, cryptographic checksum
- Hash function: takes a variable length input (pre-image) and converts it to a fixed-length (usually smaller) output (hash value)
- One-way hash function: easy to compute a hash value, hard to generate a pre-image that hashes to a specific hash value
- Hard to generate two messages M1 and M2 that generates the same hash: \( H(M1) == H(M2) \)
- MAC, Message Authentication Codes:
  - A One-way hash function with the addition of a secret key
  - Only the owner of the key can verify the hash value
- Usually hash-values are at least 128 or 160 bits
Cryptographic hash functions

• Uses:
  – Checking if files has changed
  – Storing passwords
  – Authenticating messages
MD4

- By Ron Rivest
- Not good enough, replaced by MD5
MD5

- 128 bit
- Some weaknesses have been found, but they have little practical impact
SHA

- Secure Hash Algorithm - SHA
- Designed by NIST and NSA
- Based on MD4
- 160 bit
- No known cryptographic attacks
- Preferred over MD5
MAC

• Block cipher as a MAC:
  – CBC or CFB modes: the hash is the last encrypted block encrypted once more

• Hash function MAC:
  – H(K + M) is not secure
  – H(M + K) also has problems
  – The following schemes seems secure:
    • H(K1 + H(K2 + M))
    • H(K + H(K + M))
    • H(K + p + M + K) (p - padding to a full block)
HMAC

- IETF RFC2104 – Keyed-Hashing for Message Authentication
- $\text{HMAC}(\text{text}, K) = H((K \text{ XOR opad}) + H((K \text{ XOR ipad}) + \text{text}))$
Problem with symmetric algorithms

- Distribution of keys becomes a problem.
  - Keys must be distributed with utmost security
  - Can be a complex task
  - Often keys has to be delivered by hand
- The number of keys increases with the square of the number of people exchanging secrets
  - 10 people = 45 keys, 1000 people = 499500 keys
Public key cryptography

- One key can be made public, the other one is private and secret
- No need to distribute secret keys
- Anyone who have Bob's public key can encrypt messages that only Bob can decrypt
Next time

- Public Key Cryptography