CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography
[Symmetric Encryption]

Spring 2017

Franziska (Franzi) Roesner
franzi@cs.washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Reminder

• Checkpoint for lab #1 due TONIGHT
  – Submit md5 hashes to Catalyst dropbox
• I’ll have office hours right after class today (CSE 654)
Last Time: One-Time Pad

• Easy to compute
  – Encryption and decryption are the same operation
  – Bitwise XOR is very cheap to compute

• As secure (for secrecy) as theoretically possible
  – Given a ciphertext, all plaintexts are equally likely, regardless of attacker’s computational resources
  – ... as long as the key sequence is truly random
    • True randomness is expensive to obtain in large quantities
  – ... as long as each key is same length as plaintext
    • But how does sender communicate the key to receiver?
Reducing Key Size

• What to do when it is infeasible to pre-share huge random keys?
  – When one-time pad is unrealistic…

• Use special cryptographic primitives:
  block ciphers, stream ciphers
  – Single key can be re-used (with some restrictions)
  – Not as theoretically secure as one-time pad
Stream Ciphers

- **One-time pad**: $\text{Ciphertext}(\text{Key}, \text{Message}) = \text{Message} \oplus \text{Key}$
  - Key must be a random bit sequence as long as message
- **Idea**: replace “random” with “pseudo-random”
  - Use a pseudo-random number generator (PRNG)
  - PRNG takes a short, truly random secret seed and expands it into a long “random-looking” sequence
    - E.g., 128-bit seed into a $10^6$-bit pseudo-random sequence
- $\text{Ciphertext}(\text{Key}, \text{Msg}) = \text{Msg} \oplus \text{PRNG}(\text{Key})$
  - Message processed bit by bit (unlike block cipher)

No efficient algorithm can tell this sequence from truly random
Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Each key defines a different permutation
  - Same key is reused for each block (can use short keys)
Permutations

- For N-bit input, \(2^N!\) possible permutations
- Idea for how to use a **keyed** permutation: split plaintext into blocks; for each block use secret key to pick a permutation
  - Without the key, permutation should “look random”
Block Cipher Security

• Result should look like a random permutation on the inputs

• Only computational guarantee of secrecy
  – Not impossible to break, just very expensive
    • If there is no efficient algorithm (unproven assumption!), then can only break by brute-force, try-every-possible-key search
  – Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information
Block Cipher Operation (Simplified)

Block of plaintext

Add some secret key bits to provide confusion

Key

Each S-box transforms its input bits in a “random-looking” way to provide diffusion (spread plaintext bits throughout ciphertext)

repeat for several rounds

Block of ciphertext

Procedure must be reversible (for decryption)
Standard Block Ciphers

- **DES: Data Encryption Standard**
  - Feistel structure: builds invertible function using non-invertible ones
  - Invented by IBM, issued as federal standard in 1977
  - 64-bit blocks, 56-bit key + 8 bits for parity
DES and 56 bit keys

• 56 bit keys are quite short

<table>
<thead>
<tr>
<th>Key Size (bits)</th>
<th>Number of Alternative Keys</th>
<th>Time required at 1 encryption/µs</th>
<th>Time required at 10^6 encryptions/µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>2^{32} = 4.3 × 10^9</td>
<td>2^{31} µs = 35.8 minutes</td>
<td>2.15 milliseconds</td>
</tr>
<tr>
<td>56</td>
<td>2^{56} = 7.2 × 10^{16}</td>
<td>2^{55} µs = 1,142 years</td>
<td>10.01 hours</td>
</tr>
<tr>
<td>128</td>
<td>2^{128} = 3.4 × 10^{38}</td>
<td>2^{127} µs = 5.4 × 10^{24} years</td>
<td>5.4 × 10^{18} years</td>
</tr>
<tr>
<td>168</td>
<td>2^{168} = 3.7 × 10^{50}</td>
<td>2^{167} µs = 5.9 × 10^{36} years</td>
<td>5.9 × 10^{30} years</td>
</tr>
<tr>
<td>26 characters (permutation)</td>
<td>26! = 4 × 10^{26}</td>
<td>2 × 10^{26} µs = 6.4 × 10^{12} years</td>
<td>6.4 × 10^{6} years</td>
</tr>
</tbody>
</table>

• 1999: EFF DES Crack + distributed machines
  – < 24 hours to find DES key
• DES ---> 3DES
  – 3DES: DES + inverse DES + DES (with 2 or 3 diff keys)
Standard Block Ciphers

• **DES: Data Encryption Standard**
  – Feistel structure: builds invertible function using non-invertible ones
  – Invented by IBM, issued as federal standard in 1977
  – 64-bit blocks, 56-bit key + 8 bits for parity

• **AES: Advanced Encryption Standard**
  – New federal standard as of 2001
    • NIST: National Institute of Standards & Technology
  – Based on the Rijndael algorithm
    • Selected via an open process
  – 128-bit blocks, keys can be 128, 192 or 256 bits
Encrypting a Large Message

- So, we’ve got a good block cipher, but our plaintext is larger than 128-bit block size

128-bit plaintext
(arranged as 4x4 array of 8-bit bytes)

128-bit ciphertext

- What should we do?
Electronic Code Book (ECB) Mode

- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks
Information Leakage in ECB Mode

Encrypt in ECB mode

[Wikipedia]
Cipher Block Chaining (CBC) Mode: Encryption

- Identical blocks of plaintext encrypted differently
- Last cipherblock depends on entire plaintext
  - Still does not guarantee integrity

Initialization vector (random)

Sent with ciphertext (preferably encrypted)
CBC Mode: Decryption

 Initialization vector

 plaintext

 decrypt

 ciphertext

 decrypt

 decrypt

 decrypt

 decrypt

 key

 key

 key

 key
**ECB vs. CBC**

AES in ECB mode

Similar plaintext blocks produce similar ciphertext blocks (**not good!**)

AES in CBC mode

[@Picture due to Bart Preneel]
CBC and Electronic Voting

Initialization vector (supposed to be random)

Found in the source code for Diebold voting machines:

```c
DesCBCEncrypt((des_c_block*)tmp, (des_c_block*)record.m_Data,
              totalSize, DESKEY, NULL, DES_ENCRYPT)
```
Counter Mode (CTR): Encryption

- Identical blocks of plaintext encrypted differently
- Still does not guarantee integrity; Fragile if ctr repeats
Counter Mode (CTR): Decryption

Initial ctr

 ctr  \rightarrow  \text{Key}  \rightarrow  block cipher  \rightarrow  ct  \rightarrow  \text{Key}  \rightarrow  block cipher  \rightarrow  ct  \rightarrow  \text{Key}  \rightarrow  block cipher  \rightarrow  ct  \rightarrow  \text{Key}  \rightarrow  block cipher  \rightarrow  ct  \rightarrow  \text{Key}  \rightarrow  block cipher  \rightarrow  ct

\oplus

\oplus

\oplus

\oplus

pt  \leftrightarrow  pt  \leftrightarrow  pt  \leftrightarrow  pt
When is an Encryption Scheme “Secure”?

- Hard to recover the key?
  - What if attacker can learn plaintext without learning the key?

- Hard to recover plaintext from ciphertext?
  - What if attacker learns some bits or some function of bits?

- Fixed mapping from plaintexts to ciphertexts?
  - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
  - Implication: encryption must be randomized or stateful
How Can a Cipher Be Attacked?

• Attackers knows ciphertext and encryption algthm
  – What else does the attacker know? Depends on the application in which the cipher is used!

• Ciphertext-only attack

• KPA: Known-plaintext attack (stronger)
  – Knows some plaintext-ciphertext pairs

• CPA: Chosen-plaintext attack (even stronger)
  – Can obtain ciphertext for any plaintext of his choice

• CCA: Chosen-ciphertext attack (very strong)
  – Can decrypt any ciphertext except the target
Chosen Plaintext Attack (CPA)

Crook #1 changes his PIN to a number of his choice.

PIN is encrypted and transmitted to bank.

cipher(key, PIN)

Crook #2 eavesdrops on the wire and learns ciphertext corresponding to chosen plaintext PIN.

... repeat for any PIN value.
Chosen Plaintext Security Game

• Attacker does not know the key
• She chooses as many plaintexts as she wants, and receives the corresponding ciphertexts
• When ready, she picks two plaintexts $M_0$ and $M_1$
  – He is even allowed to pick plaintexts for which he previously learned ciphertexts!
• She receives either a ciphertext of $M_0$, or a ciphertext of $M_1$
• She wins if she guesses correctly which one it is

→ Any deterministic, stateless symmetric encryption scheme (such as ECB mode) is insecure against chosen plaintext attacks.
**Very Informal Intuition**

- **Security against chosen-plaintext attack (CPA)**
  - Ciphertext leaks no information about the plaintext
  - Even if the attacker correctly guesses the plaintext, he cannot verify his guess
  - Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts

- **Security against chosen-ciphertext attack (CCA)**
  - Integrity protection – it is not possible to change the plaintext by modifying the ciphertext

Minimum security requirement for a modern encryption scheme
Why Hide Everything?

• Leaking even a little bit of information about the plaintext can be disastrous

• Electronic voting
  – 2 candidates on the ballot (1 bit to encode the vote)
  – If ciphertext leaks the parity bit of the encrypted plaintext, eavesdropper learns the entire vote

• Also, want a strong definition, that implies other definitions (like not being able to obtain key)