Symmetric encryption

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu
Orome1 writes

"Two U.S. satellites have been tampered with by hackers — possibly Chinese ones — in 2007 and 2008, claims a soon-to-be released report by the the U.S.-China Economic and Security Review Commission. The two satellites, Landsat-7 and Terra AM-1, had been interfered with on four separate occasions, allowing the attackers to be in command of the satellites for two to over twelve minutes each time. Luckily, both of the satellites are used only for observing the Earth's climate and terrain, and the hackers never actually misused their control over them in any way."
Symmetric encryption

- Block ciphers
- Modes of operation
- Hash functions
- HMAC
- Authenticated encryption
Cryptography as computational science

Use computational intractability as basis for confidence in systems

1. Design a cryptographic scheme
2. Provide proof that no attacker with limited computational resources can break it

Goldwasser, Micali, and Blum circa 1980’s

Security proofs (reductions)

- Breaking scheme
- Breaking assumptions

Provable security yields

1) well-defined assumptions and security goals
2) cryptanalysts can focus on assumptions and models

Example:

- Attacker cannot recover credit card
- Can factor large composite numbers

But no one knows how to do this. It’s been studied for a very long time!
Typical assumptions

• Basic atomic primitives are hard to break:
  – Factoring of large composites intractable
  – RSA permutation hard-to-invert
  – Block ciphers (AES, DES) are good pseudorandom permutations (PRPs)
  – Hash functions are collision resistant

Confidence in atomic primitives is gained by cryptanalysis, public design competitions
Symmetric encryption

Correctness: \( D(K, E(K, M, R)) = M \) with probability 1 over randomness used
In TLS symmetric encryption underlies the Record Layer

What security properties do we need from symmetric encryption?

1) **Confidentiality**: should not learn any information about M
2) **Authenticity**: should not be able to forge messages

Often referred to as Authenticated Encryption security
Block ciphers

Key generation

R → Kg

Key is a uniformly selected bit string of length k

K

M → E → C

E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n

C → D → M

Implements a family of permutations on n bit strings, one permutation for each K

Key is a uniformly selected bit string of length k
Data encryption standard (DES)

Originally called Lucifer
- team at IBM
- input from NSA
- standardized by NIST in 1976

\[ n = 64 \quad \text{Number of keys:} \quad 72,057,594,037,927,936 \]
\[ k = 56 \]

Split 64-bit input into \( L_0, R_0 \) of 32 bits each
Repeat Feistel round 16 times
Each round applies function \( F \) using separate round key

\[ F_{K_1} \]
\[ + \]

\[ L_0 \]
\[ R_0 \]

\[ L_1 \]
\[ R_1 \]

\[ F_{K_2} \]
\[ + \]

\[ L_2 \]
\[ R_2 \]

\[ \vdots \]
# Best attacks against DES

<table>
<thead>
<tr>
<th>Attack</th>
<th>Attack type</th>
<th>Complexity</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biham, Shamir</td>
<td>Chosen plaintexts, recovers key</td>
<td>$2^{47}$ plaintext, ciphertext pairs</td>
<td>1992</td>
</tr>
<tr>
<td>DESCHALL</td>
<td>Unknown plaintext, recovers key</td>
<td>$2^{56/4}$ DES computations 41 days</td>
<td>1997</td>
</tr>
<tr>
<td>EFF Deepcrack</td>
<td>Unknown plaintext, recovers key</td>
<td>$\sim$4.5 days</td>
<td>1998</td>
</tr>
<tr>
<td>Deepcrack + DESCHALL</td>
<td>Unknown plaintext, recovers key</td>
<td>22 hours</td>
<td>1999</td>
</tr>
</tbody>
</table>

- DES is still used in some places
- 3DES (use DES 3 times in a row with more keys) expands keyspace and still used widely in practice
Advanced Encryption Standard (AES)

Response to 1999 attacks:
- NIST has design competition for new block cipher standard
- 5 year design competition
- 15 designs, Rijndael design chosen
Advanced Encryption Standard (AES)

Rijndael (Rijmen and Daemen)

\[ n = 128 \]
\[ k = 128, 192, 256 \]

Number of keys for \( k = 128 \):
340,282,366,920,938,463,463,374,607,431,768,211,456

Substitution-permutation design.
\( k = 128 \) has 10 rounds of:

1) Permute:
   - SubBytes (non-linear S-boxes)
   - ShiftRows + MixCols (invertible linear transform)

2) XOR in a round key derived from \( K \)
   (Actually last round skips MixCols)
## Best attacks against AES

<table>
<thead>
<tr>
<th>Attack</th>
<th>Attack type</th>
<th>Complexity</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bogdanov, Khovratovich, Rechberger</td>
<td>chosen ciphertext, recovers key</td>
<td>2^{126.1} time + some data overheads</td>
<td>2011</td>
</tr>
</tbody>
</table>

- Brute force requires time $2^{128}$
- Approximately factor 4 speedup
Are block ciphers good for record layers?

Functional limitations:
- Only encrypt messages that fit in n bits

Security limitations:
- Confidentiality: $M = M' \implies E(K,M) = E(K,M')$
- Authenticity: any C of length n is valid ciphertext
Block cipher modes of operation

How can we build an encryption scheme for arbitrary message spaces out of block cipher?

Electronic codebook (ECB) mode
Pad message $M$ to $M_1, M_2, M_3, ...$ where each block $M_i$ is $n$ bits

Then:

$$
\begin{align*}
&\text{M1} \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C_1 \\
\end{align*}
\quad
\begin{align*}
&\text{M2} \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C_2 \\
\end{align*}
\quad
\begin{align*}
&\text{M3} \\
&\downarrow \\
&E_K \\
&\downarrow \\
&C_3 \\
\end{align*}
$$
ECB mode is a more complicated looking substitution cipher

Recall our credit-card number example.
ECB: substitution cipher with alphabet n-bit strings instead of digits

Images courtesy of
OTP-like encryption using block cipher

Counter mode (CTR)
Pad message $M$ to $M_1, M_2, M_3, \ldots$ where each is $n$ bits except last
Choose random $n$-bit string $IV$
Then:

How do we decrypt?

Maybe use less than full $n$ bits of $P_3$
Another option: CBC mode

Ciphertext block chaining (CBC)
Pad message M to M1, M2, M3,... where each block Mi is n bits
Choose random n-bit string IV
Then:

How do we decrypt?
Security of CBC mode

Can attacker learn $K$ from $C_0, C_1, C_2, C_3$?

This would imply one can recover block cipher key

Can attacker learn $M = M_1, M_2, M_3$ from $C_0, C_1, C_2, C_3$?

This would imply inverting the block cipher without knowing $K$

Passive adversaries cannot learn anything about messages
Active security of CBC mode

What about forging a message? Pick any $C_0'$, $C_1'$ ...

Better yet for any $D$: $C_0' \oplus D$
### Chosen ciphertext attacks against CBC

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vaudenay</td>
<td>10’s of chosen ciphertexts, recovers message bits from a ciphertext. Called “padding oracle attack”</td>
<td>2001</td>
</tr>
<tr>
<td>Canvel et al.</td>
<td>Shows how to use Vaudenay’s ideas against TLS</td>
<td>2003</td>
</tr>
<tr>
<td>Degabriele, Paterson</td>
<td>Breaks IPsec encryption-only mode</td>
<td>2006</td>
</tr>
<tr>
<td>Albrecht et al.</td>
<td>Plaintext recovery against SSH</td>
<td>2009</td>
</tr>
<tr>
<td>Duong, Rizzo</td>
<td>Breaking ASP.net encryption</td>
<td>2011</td>
</tr>
<tr>
<td>Jager, Somorovsky</td>
<td>XML encryption standard</td>
<td>2011</td>
</tr>
<tr>
<td>Duong, Rizzo</td>
<td>“Beast” attacks against TLS</td>
<td>2011</td>
</tr>
</tbody>
</table>
Hash functions and message authentication

Hash function $H$ maps arbitrary bit string to fixed length string of size $m$

$$M \xrightarrow{H} H(M)$$

MD5: $m = 128$ bits
SHA-1: $m = 160$ bits
SHA-256: $m = 256$ bits

Some security goals:
- collision resistance: can’t find $M \neq M'$ such that $H(M) = H(M')$
- preimage resistance: given $H(M)$, can’t find $M$
- second-preimage resistance: given $H(M)$, can’t find $M'$ s.t. $H(M') = H(M)$
Hash function application example

Password hashing. Choose random salt and store (salt,h) where:

\[
\text{salt} \ || \ pw \xrightarrow{H} h
\]

The idea: Attacker, given (salt,h), should not be able to recover pw

Or can they?

For each guess pw':
If \( H(\text{salt} \ || \ pw') = h \) then
Ret pw'

Rainbow tables speed this up in practice by precomputing. Large salts make rainbow tables impractical.
Message authentication

Optional. If no randomness, then called a Message Authentication Code (MAC)

Correctness: \( \text{Ver}(K, \text{Tag}(K,M,R)) = 1 \) with probability 1 over randomness used

Unforgeability: Attacker can’t find \( M', T \) such that \( V(K, M', T) = 1 \)
Message authentication with HMAC

Use a hash function $H$ to build MAC. $K_g$ outputs uniform bit string $K$

$Tag(K,M) = HMAC(K,M)$ defined by:

$$K \oplus \text{ipad} || M \xrightarrow{H} K \oplus \text{opad} || h \xrightarrow{H} T$$

ipad $\neq$ opad are constants

To verify a $M,T$ pair, check if $HMAC(K,M) = T$
Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac
Build a new scheme from CBC and HMAC
Kg outputs CBC key K1 and HMAC key K2

Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac

Thm. If encryption scheme provides confidentiality against passive attackers and MAC provides unforgeability, then Encrypt-then-MAC provides secure authenticated encryption
TLS record protocol: MAC-Encode-Encrypt (MEE)

- **MAC**
  - HMAC-MD5, HMAC-SHA1, HMAC-SHA256

- **Encrypt**
  - CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

Padding is not MAC’d. Implementations must handle padding checks very carefully.