Symmetric Encryption & Authentication

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Encrypting a Large Message

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

- **Electronic Code Book (ECB) mode**
  - Split plaintext into blocks, encrypt each one separately using the block cipher

- **Cipher Block Chaining (CBC) mode**
  - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks

- **Counter (CTR) mode**
  - Use block cipher to generate keystream, like a stream cipher

- …
**ECB Mode**

- Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks
CBC Mode: Encryption

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

Initialization vector

plaintext

ciphertext

decrypt

decrypt

decrypt

decrypt
CTR Mode: Encryption

- Identical blocks of plaintext encrypted differently
- Still does not guarantee integrity
CTR Mode: Decryption

Initial ctr → ctr → block cipher → ct → pt
ctr+1 → block cipher → ct → pt
ctr+2 → block cipher → ct → pt
ctr+3 → block cipher → ct → pt
ECB vs. CBC

AES in ECB mode

AES in CBC mode

Similar plaintext blocks produce similar ciphertext blocks (not good!)
Information Leakage in ECB Mode

[Wikipedia]
CBC and Electronic Voting

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)
```
When Is a Cipher “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?

Assume that the attacker knows the encryption algorithm and wants to decrypt some ciphertext.

Main question: what else does attacker know?
  - Depends on the application in which cipher is used!

- Ciphertext-only attack

- Known-plaintext attack (stronger)
  - Knows some plaintext-ciphertext pairs

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

- Chosen-ciphertext attack (very strong)
  - Can decrypt any ciphertext except the target
  - Sometimes very realistic model
The Chosen-Plaintext Game

- Attacker does not know the key
- He chooses as many plaintexts as he wants, and learns the corresponding ciphertexts
- When ready, he picks two plaintexts $M_0$ and $M_1$
  - He is even allowed to pick plaintexts for which he previously learned ciphertexts!
- He receives either a ciphertext of $M_0$, or a ciphertext of $M_1$
- He wins if he guesses correctly which one it is
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
- D-Day: Pas-de-Calais or Normandy?
  - Allies convinced Germans that invasion will take place at Pas-de-Calais
    - Dummy landing craft, feed information to double spies
  - Goal: hide a 1-bit secret
- Also, want a strong definition, that implies others
Defining Security

- Idea: attacker should not be able to learn even a single bit of the encrypted plaintext
- Define $\text{Enc}(M_0, M_1, b)$ to be a function that returns encrypted $M_b$
  - Given two plaintexts, Enc returns a ciphertext of one or the other depending on the value of bit $b$
  - Think of Enc as a magic box that computes ciphertexts on attacker’s demand. He can obtain a ciphertext of any plaintext $M$ by submitting $M_0 = M_1 = M$, or he can try to learn even more by submitting $M_0 \neq M_1$.
- Attacker’s goal is to learn just one bit $b$
Chosen-Plaintext Security

Consider two experiments (A is the attacker)

Experiment 0
A interacts with Enc(-,-,0)
and outputs bit d
• Identical except for the value of the secret bit
• d is attacker’s guess of the secret bit

Experiment 1
A interacts with Enc(-,-,1)
and outputs bit d

Attacker’s advantage is defined as

\[ | \text{Prob}(A \text{ outputs } 1 \text{ in Exp0}) - \text{Prob}(A \text{ outputs } 1 \text{ in Exp1}) | \]

Encryption scheme is \textit{chosen-plaintext secure} if this advantage is negligible for any efficient A

If A “knows” secret bit, he should be able to make his output depend on it.
Simple Example

- **Any** deterministic, stateless symmetric encryption scheme is insecure
  - Attacker can easily distinguish encryptions of different plaintexts from encryptions of identical plaintexts
  - This includes ECB mode of common block ciphers!

Attacker $A$ interacts with $\text{Enc}(\cdot,\cdot,b)$

Let $X,Y$ be any two different plaintexts

$C_1 \leftarrow \text{Enc}(X,Y,b); \quad C_2 \leftarrow \text{Enc}(Y,Y,b);$

If $C_1 = C_2$ then $b=1$ else say $b=0$

- The advantage of this attacker $A$ is 1

$$\text{Prob}(A \text{ outputs 1 if } b=0)=0 \quad \text{Prob}(A \text{ outputs 1 if } b=1)=1$$
International Criminal Tribunal for Rwanda


Credits: Alexei Czeskis, Karl Koscher, Batya Friedman
Integrity

Software manufacturer wants to ensure that the executable file is received by users without modification. It sends out the file to users and publishes its hash in NY Times. The goal is integrity, not secrecy.

Idea: given goodFile and hash(goodFile), very hard to find badFile such that hash(goodFile) = hash(badFile)
Integrity vs. Secrecy

- **Integrity**: attacker cannot tamper with message
- **Encryption does not always guarantee integrity**
  - Intuition: attacker may able to modify message under encryption without learning what it is
    - One-time pad: given key $K$, encrypt $M$ as $M \oplus K$
    - This guarantees perfect secrecy, but attacker can easily change unknown $M$ under encryption to $M \oplus M'$ for any $M'$
    - Online auction: halve competitor’s bid without learning its value
  - This is recognized by industry standards (e.g., PKCS)
    - “RSA encryption is intended primarily to provide confidentiality... It is not intended to provide integrity” (from RSA Labs Bulletin)
**Hash Functions: Main Idea**

- **H is a lossy compression function**
  - **Collisions:** $h(x) = h(x')$ for distinct inputs $x, x'$
  - Result of hashing should “look random” (make this precise later)
    - Intuition: half of digest bits are “1”; any bit in digest is “1” half the time
- **Cryptographic hash function** needs a few properties...
One-Way

◆ Intuition: hash should be hard to invert
  • “Preimage resistance”
  • Let $h(x') = y \in \{0,1\}^n$ for a random $x'$
  • Given $y$, it should be hard to find any $x$ such that $h(x) = y$

◆ How hard?
  • Brute-force: try every possible $x$, see if $h(x) = y$
  • SHA-1 (common hash function) has 160-bit output
    – Suppose have hardware that’ll do $2^{30}$ trials a pop
    – Assuming $2^{34}$ trials per second, can do $2^{89}$ trials per year
    – Will take around $2^{70}$ years to invert SHA-1 on a random image
Collision Resistance

- Should be hard to find distinct $x, x'$ such that $h(x) = h(x')$
  - Brute-force collision search is only $O(2^{n/2})$, not $O(2^n)$
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$

- Birthday paradox (informal)
  - Let $t$ be the number of values $x, x', x'', \ldots$ we need to look at before finding the first pair $x, x'$ s.t. $h(x) = h(x')$
  - What is probability of collision for each pair $x, x'$? $1/2^n$
  - How many pairs would we need to look at before finding the first collision? $O(2^n)$
  - How many pairs $x, x'$ total? $\text{Choose}(2,t) = t(t-1)/2 \sim O(t^2)$
  - What is $t$? $2^{n/2}$
One-Way vs. Collision Resistance

One-wayness does **not** imply collision resistance

- Suppose \( g \) is one-way
- Define \( h(x) = g(x') \) where \( x' \) is \( x \) except the last bit
  - \( h \) is one-way (to invert \( h \), must invert \( g \))
  - Collisions for \( h \) are easy to find: for any \( x \), \( h(x_0) = h(x_1) \)

Collision resistance does **not** imply one-wayness

- Suppose \( g \) is collision-resistant
- Define \( h(x) \) to be 0x if \( x \) is n-bit long, 1g(x) otherwise
  - Collisions for \( h \) are hard to find: if \( y \) starts with 0, then there are no collisions, if \( y \) starts with 1, then must find collisions in \( g \)
  - \( h \) is not one way: half of all \( y \)'s (those whose first bit is 0) are easy to invert (how?); random \( y \) is invertible with probab. 1/2
Weak Collision Resistance

- Given randomly chosen x, hard to find x’ such that \( h(x) = h(x') \)
  - Attacker must find collision for a specific x. By contrast, to break collision resistance, enough to find any collision.
  - Brute-force attack requires \( O(2^n) \) time
  - AKA second-preimage collision resistance

- Weak collision resistance does not imply collision resistance
Which Property Do We Need?

- UNIX passwords stored as hash(password)
  - One-wayness: hard to recover password
- Integrity of software distribution
  - Weak collision resistance
  - But software images are not really random... maybe need full collision resistance
- Auction bidding
  - Alice wants to bid B, sends H(B), later reveals B
  - One-wayness: rival bidders should not recover B
  - Collision resistance: Alice should not be able to change her mind to bid B’ such that H(B)=H(B’)
Common Hash Functions

- **MD5**
  - 128-bit output
  - Designed by Ron Rivest, used very widely
  - Collision-resistance broken (summer of 2004)

- **RIPEMD-160**
  - 160-bit variant of MD5

- **SHA-1 (Secure Hash Algorithm)**
  - 160-bit output
  - US government (NIST) standard as of 1993-95
    - Also the hash algorithm for Digital Signature Standard (DSS)

- **SHA-256, SHA-512 (today)**

- **AHS (NIST competition, future)**
Basic Structure of SHA-1 (Skip)

Split message into 512-bit blocks

\[ L \times 512 \text{ bits} = N \times 32 \text{ bits} \]

Padding (1 to 512 bits)

Message length \((K \mod 2^{64})\)

160-bit buffer (5 registers) initialized with magic values

Compression function
- Applied to each 512-bit block and current 160-bit buffer
- This is the heart of SHA-1
SHA-1 Compression Function (Skip)

Current message block

Current buffer (five 32-bit registers A,B,C,D,E)

Four rounds, 20 steps in each

Let's look at each step in more detail...

Fifth round adds the original buffer to the result of 4 rounds

Buffer contains final hash value

Very similar to a block cipher, with message itself used as the key for each round

Let's look at each step in more detail…

Fifth round adds the original buffer to the result of 4 rounds

Buffer contains final hash value

Note: addition (+) is mod $2^{32}$
One Step of SHA-1 (80 steps total) (Skip)

5 bitwise left-rotate

Logic function for steps
- \((B \land C) \lor \neg (B \land D)\) 0..19
- \(B \oplus C \oplus D\) 20..39
- \((B \land C) \lor (B \land D) \lor (C \land D)\) 40..59
- \(B \oplus C \oplus D\) 60..79

Current message block mixed in
- For steps 0..15, \(W_{0..15} = \text{message block}\)
- For steps 16..79,
  \(W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}\)

Multi-level shifting of message blocks

30 bitwise left-rotate

Special constant added
(same value in each 20-step round, 4 different constants altogether)
How Strong Is SHA-1?

- Every bit of output depends on every bit of input
  - Very important property for collision-resistance
- Brute-force inversion requires $2^{160}$ ops, birthday attack on collision resistance requires $2^{80}$ ops
- Some recent weaknesses
  - Collisions can be found in $2^{63}$ ops (2005)
Authentication Without Encryption

Integrity and authentication: only someone who knows KEY can compute MAC for a given message.

Recomputes MAC and verifies whether it is equal to the MAC attached to the message.
HMAC

- Construct MAC by applying a cryptographic hash function to message and key
  - Could also use encryption instead of hashing, but...
  - Hashing is faster than encryption in software
  - Library code for hash functions widely available
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption
- Invented by Bellare, Canetti, and Krawczyk (1996)
  - HMAC strength established by cryptographic analysis
- Mandatory for IP security, also used in SSL/TLS
Structure of HMAC

- Secret key padded to block size
- Magic value (flips half of key bits)
- Block size of embedded hash function
- Another magic value (flips different key bits)
- Embedded hash function (strength of HMAC relies on strength of this hash function)
- "Amplify" key material (get two keys out of one)
- Very common problem: given a small secret, how to derive a lot of new keys?
- hash(key, hash(key, message))