CSE 484 (Winter 2008)

Goals for Today

- Cryptography Background
- Symmetric (Shared-Key Foundations)

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 ciphertext block depends on entire plaintext
- Still does not guarantee integrity

CBC Mode: Decryption

CTR Mode: Encryption

CTR Mode: Decryption
ECB vs. CBC

AES in ECB mode

Similar plaintext makes similar ciphertext block (not good!)

AES in CBC mode

Information Leakage in ECB Mode

Encrypt in ECB mode

CBC and Electronic Voting

Initialization vector (supposed to be random)

plaintext

DES DES DES DES

ciphertext

Found in the source code for Diebold voting machines:

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

Chosen-Plaintext Attack

- Crook #1 changes his PIN to a number of his choice.
- PIN is encrypted and transmitted to bank.
- Crook #2 eavesdrops on the wire and learns ciphertext corresponding to chosen plaintext PIN.
- … repeat for any PIN value.

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.

Defining Security (Skip in Class)

- Idea: attacker should not be able to learn even a single bit of the encrypted plaintext.
- Define \( Enc(M_0, M_1, b) \) to be a function that returns encrypted \( M_b \), \( b = 0 \) or \( 1 \).
  - 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 \( M_0 \) or \( M_1 \), or he can try to learn even more by submitting \( M_0 + M_1 \).
- Attacker's goal is to learn just one bit \( b \).
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, an attacker 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.

Chosen-Plaintext Security (Skip)

Consider two experiments (A is the attacker):

- Experiment 0: A interacts with Enc(-,-,0) and outputs bit d.
- Experiment 1: A interacts with Enc(-,-,1) and outputs bit d.

Identical except for the value of the secret bit.

Attacker’s advantage is defined as:

\[ |\Pr(A \text{ outputs } 1 \text{ in Exp}0) - \Pr(A \text{ outputs } 1 \text{ in Exp}1)| \]

Encryption scheme is chosen-plaintext secure if this advantage is negligible for any efficient A.

Simple Example (Skip in Class)

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 Enc(-,-,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:

\[ \Pr(A \text{ outputs } 1 \text{ if } b = 0) = 0 \]

\[ \Pr(A \text{ outputs } 1 \text{ if } b = 1) = 1 \]

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 has(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 be able to modify message under encryption without learning what it is
  - One-time pad: given key K, encrypt M as M XOR K
  - This guarantees perfect secrecy, but attacker can easily change unknown M under encryption to M XOR K for any K
  - 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 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 $2^{71}$ 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 $O(2^n)$, not $O(2^n)$
- For SHA-1, this means $O(2^{160})$ vs. $O(2^{160})$
- Birthday paradox (informal)
  - Let t be the number of values x, x', x''… we need to look at before finding the first pair x, x' such that 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?
  - $O(2^{n/2})$
  - How many pairs x, x' total?
  - $2^n - 1$
  - What is $t$?
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 \( x \), \( h(x0) = h(x1) \)

- Collision resistance does not imply one-wayness
  - Suppose \( g \) is collision-resistant
  - Define \( h(x) = 0x \) 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 prob \( 1/2 \)

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