CSci 5271
Introduction to Computer Security
Day 17: Cryptography part 2: symmetric

Stephen McCamant

University of Minnesota, Computer Science & Engineering

Outline

Stream ciphers

- Closest computational version of one-time pad
- Key (or seed) used to generate a long pseudorandom bitstream
- Closely related: cryptographic RNG

Shift register stream ciphers

- Linear-feedback shift register (LFSR): easy way to generate long pseudorandom sequence
  - But linearity allows for attack
- Several ways to add non-linearity
- Common in constrained hardware, poor security record

RC4

- Fast, simple, widely used software stream cipher
  - Previously a trade secret, also "ARCFOUR"
- Many attacks, none yet fatal to careful users (e.g. TLS)
  - Famous non-careful user: WEP
- Now deprecated, not recommended for new uses

Encryption $\neq$ integrity

- Encryption protects secrecy, not message integrity
- For constant-size encryption, changing the ciphertext just creates a different plaintext
- How will your system handle that?
- Always need to take care of integrity separately

Stream cipher mutability

- Strong example of encryption vs. integrity
- In stream cipher, flipping a ciphertext bit flips the corresponding plaintext bit, only
- Very convenient for targeted changes

Salsa and ChaCha

- Published by Daniel Bernstein 2007-2008
- Stream cipher with random access to stream
  - Related to counter mode discussed later
- Fast on general-purpose CPUs without specialized hardware
- Adopted as option for TLS and SSH
  - Prominent early adopter: Chrome on Android

Stream cipher assessment

- Currently out of fashion as a primitive in software
- Not inherently insecure
  - Other common pitfall: must not reuse key(stream)

Outline

Stream ciphers, cont'd
Announcements
Block ciphers
Discuss: bad block ciphers
Block cipher modes of operation
Hash functions and MACs
Building a secure channel

Note to early readers

- This is the section of the slides most likely to change in the final version
- If class has already happened, make sure you have the latest slides for announcements

Outline

Stream ciphers, cont'd
Announcements
Block ciphers
Discuss: bad block ciphers
Block cipher modes of operation
Hash functions and MACs
Building a secure channel

Basic idea

- Encryption/decryption for a fixed sized block
- Insecure if block size is too small
  - Barely enough: 64 bits; current standard: 128
- Reversible, so must be one-to-one and onto function

Pseudorandom permutation

- Ideal model: key selects a random invertible function
- I.e., permutation (PRP) on block space
  - Note: not permutation on bits
- "Strong" PRP: distinguisher can decrypt as well as encrypt

Confusion and diffusion

- Basic design principles articulated by Shannon
- Confusion: combine elements so none can be analyzed individually
- Diffusion: spread the effect of one symbol around to others
- Iterate multiple *rounds* of transformation

Substitution/permutation network

- Parallel structure combining reversible elements:
- Substitution: invertible lookup table ("S-box")
- Permutation: shuffle bits

AES

- Advanced Encryption Standard: NIST contest 2001
  - Developed under the name Rijndael
- 128-bit block, 128/192/256-bit key
- Fast software implementation with lookup tables (or dedicated insns)
- Allowed by US government up to Top Secret

Feistel cipher

- Split block in half, operate in turn:
  $(L_{i+1}, R_{i+1}) = (R_i, L_i \oplus F(R_i, K_i))$
- Key advantage: F need not be invertible
  - Also saves space in hardware
- Luby-Rackoff: if F is pseudo-random, 4 or more rounds gives a strong PRP

DES

- Data Encryption Standard: AES predecessor 1977-2005
- 64-bit block, 56-bit key
- Implementable in 70s hardware, not terribly fast in software
- Triple DES variant still used in places

Some DES history

- Developed primarily at IBM, based on an earlier cipher named "Lucifer"
- Final spec helped and "helped" by the NSA
  - Argued for smaller key size
  - S-boxes tweaked to avoid a then-secret attack
- Eventually victim to brute-force attack

DES brute force history

1977 est. $20m cost custom hardware
1993 est. $1m cost custom hardware
1997 distributed software break
1998 $250k built ASIC hardware
2006 $10k FPGAs
2012 as-a-service against MS-CHAPv2

Double encryption?

- Combine two different block ciphers?
  - Belt and suspenders
- Anderson: don't do it
- FS&K: could do it, not a recommendation
- Maurer and Massey (J.Crypt'93): might only be as strong as first cipher

Outline

Stream ciphers, cont'd
Announcements
Block ciphers
Discuss: bad block ciphers
Block cipher modes of operation
Hash functions and MACs
Building a secure channel

Motivation: why bad?

- Goal: practice how the classes of crypto attacks apply
- Let's think about how block ciphers can be insecure
- A mathematical setting for the adversarial mindset

Common kinds of bad

- Decryption not an inverse of encryption
  - Officially, not a block cipher at all
- Plaintext recovery attack possible
- Key recovery attack possible
- Distinguisher (vs. true random) possible

Chosen plaintext model

- Initially unknown "challenge" key, plaintext, and ciphertext
  - $C_C = \mathsf{Enc}(P_C, k_C)$
- Given C_C, attacker chooses $P_1, P_2, \ldots, P_n$, and gets $C_i = \mathsf{Enc}(P_i, k_C)$
- Attacker wins by efficiently computing P_C or k_C
- If any P_i values work, it's "known plaintext"

Concrete setup in C

```
uint64_t enc(uint64_t p, uint64_t k) {
    return /* some func of p and k */;
}
uint64_t dec(uint64_t c, uint64_t k) {
    /* Such that
       dec(enc(p, k), k) == p
    */
}
```

Outline

Stream ciphers, cont'd
Announcements
Block ciphers
Discuss: bad block ciphers
Block cipher modes of operation
Hash functions and MACs
Building a secure channel

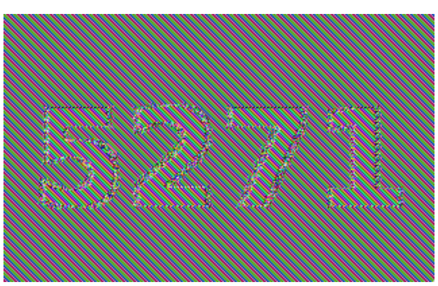
Modes of operation

- How to build a cipher for arbitrary-length data from a block cipher
- Many approaches considered
  - For some reason, most have three-letter acronyms
- More recently: properties susceptible to relative proof

ECB

- Electronic CodeBook
- Split into blocks, apply cipher to each one individually
- Leaks equalities between plaintext blocks
- Almost never suitable for general use

Do not use ECB



CBC

- Cipher Block Chaining
- $C_i = E_K(P_i \oplus C_{i-1})$
- Probably most popular in current systems
- Plaintext changes propagate forever, ciphertext changes only one block

CBC: getting an IV

- C_0 is called the initialization vector (IV)
  - Must be known for decryption
- IV should be random-looking
  - To prevent first-block equalities from leaking (lesser version of ECB problem)
- Common approaches
  - Generate at random
  - Encrypt a nonce

Stream modes: OFB, CTR

- Output FeedBack: produce keystream by repeatedly encrypting the IV
  - Danger: collisions lead to repeated keystream
- Counter: produce from encryptions of an incrementing value
  - Recently becoming more popular: allows parallelization and random access

Outline

Ideal model

- Ideal crypto hash function: pseudorandom function
  - Arbitrary input, fixed-size output
- Simplest kind of elf in box, theoretically very convenient
- But large gap with real systems: better practice is to target particular properties

Kinds of attacks

- Pre-image, "inversion": given y, find x such that $H(x) = y$
- Second preimage, targeted collision: given x, $H(x)$, find $x' \neq x$ such that $H(x') = H(x)$
- (Free) collision: find x_1, x_2 such that $H(x_1) = H(x_2)$

Birthday paradox and attack

- There are almost certainly two people in this classroom with the same birthday
- n people have $\binom{n}{2} = \Theta(n^2)$ pairs
- So only about $\sqrt{n}$ expected for collision
- "Birthday attack" finds collisions in any function

Security levels

- For function with k-bit output:
- Preimage and second preimage should have complexity 2^k
- Collision has complexity $2^{k/2}$
- Conservative: use hash function twice as big as block cipher key
  - Though if you're paranoid, cipher blocks can repeat too

Non-cryptographic hash functions

- The ones you probably use for hash tables
- CRCs, checksums
- Output too small, but also not resistant to attack
- E.g., CRC is linear and algebraically nice

Short hash function history

- On the way out: MD5 (128 bit)
  - Flaws known, collision-finding now routine
- SHA(-0): first from NIST/NSA, quickly withdrawn
  - Likely flaw discovered 3 years later
- SHA-1: fixed SHA-0, 160-bit output.
- 2^{60} collision attack described in 2013
  - First public collision found (using 6.5 kCPU yr) in 2017

Length extension problem

- MD5, SHA1, etc., computed left to right over blocks
- Can sometimes compute $H(a \parallel b)$ in terms of $H(a)$
  - $\parallel$ means bit string concatenation
- Makes many PRF-style constructions insecure

SHA-2 and SHA-3

- SHA-2: evolutionary, larger, improvement of SHA-1
  - Exists as SHA-$\{224, 256, 384, 512\}$
  - But still has length-extension problem
- SHA-3: chosen recently in open competition like AES
  - Formerly known as Keccak, official standard Aug. 2015
  - New design, fixes length extension
  - Not yet very widely used

MAC: basic idea

- Message authentication code: similar to hash function, but with a key
- Adversary without key cannot forge MACs
- Strong definition: adversary cannot forge anything, even given chosen-message MACs on other messages

CBC-MAC construction

- Same process as CBC encryption, but:
  - Start with IV of 0
  - Return only the last ciphertext block
- Both these conditions needed for security
- For fixed-length messages (only), as secure as the block cipher

HMAC construction

- $H(K \parallel M)$: insecure due to length extension
  - Still not recommended: $H(M \parallel K)$, $H(K \parallel M \parallel K)$
- HMAC: $H(K \oplus a \parallel H(K \oplus b \parallel M))$
- Standard $a = \texttt{0x5c}^*$, $b = \texttt{0x36}^*$
- Probably the most widely used MAC

Outline

Stream ciphers, cont'd
Announcements
Block ciphers
Discuss: bad block ciphers
Block cipher modes of operation
Hash functions and MACs
Building a secure channel

Session keys

- Don't use your long term password, etc., directly as a key
- Instead, *session key* used for just one channel
- In modern practice, usually obtained with public-key crypto
- Separate keys for encryption and MACing

Order of operations

- Encrypt and MAC ("in parallel")
  - Safe only under extra assumptions on the MAC
- Encrypt then MAC
  - Has cleanest formal safety proof
- MAC then Encrypt
  - Preferred by FS&K for some practical reasons
  - Can also be secure

Authenticated encryption modes

- Encrypting and MACing as separate steps is about twice as expensive as just encrypting
- "Authenticated encryption" modes do both at once
  - Newer (circa 2000) innovation, many variants
- NIST-standardized and unpatented: Galois Counter Mode (GCM)

Ordering and message numbers

- Also don't want attacker to be able to replay or reorder messages
- Simple approach: prefix each message with counter
- Discard duplicate/out-of-order messages

Padding

- Adjust message size to match multiple of block size
- To be reversible, must sometimes make message longer
- E.g.: for 16-byte block, append either 1, or 2 2, or 3 3 3, up to 16 "16" bytes

Padding oracle attack

- Have to be careful that decoding of padding does not leak information
- E.g., spend same amount of time MACing and checking padding whether or not padding is right
- Remote timing attack against CBC TLS published 2013

Don't actually reinvent the wheel

- This is all implemented carefully in OpenSSL, SSH, etc.
- Good to understand it, but rarely sensible to reimplement it
- You'll probably miss at least one of decades' worth of attacks