Outline

Block ciphers and modes of operation, cont'd
- Hash functions and MACs
- Building a secure channel
- Announcements
Public-key crypto basics
Public key encryption and signatures

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

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
- Block ciphers and modes of operation, cont’d
- Hash functions and MACs
- Building a secure channel
- Announcements
- Public-key crypto basics
- Public key encryption and signatures
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(-O): 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 $H(\alpha \| b)$ in terms of $H(\alpha)$
  - $\|$ 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

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

Outline

- Block ciphers and modes of operation, cont’d
- Hash functions and MACs
- Building a secure channel
- Announcements
- Public-key crypto basics
- Public key encryption and signatures

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

Outline

- Block ciphers and modes of operation, cont’d
- Hash functions and MACs
- Building a secure channel
- Announcements
- Public-key crypto basics
- Public key encryption and signatures

Grading progress

- Exercise set 2 grades will be posted tonight
  - Re-grade request deadline will be next Wednesday 11/10
  - Midterm grades and solutions by this Wednesday

Research project meetings

- Look for emails soon inviting you to project meetings over the rest of this week
- Also next written progress reports next Wednesday

Pre-history of public-key crypto

- First invented in secret at GCHQ
- Proposed by Ralph Merkle for UC Berkeley grad. security class project
  - First attempt only barely practical
  - Professor didn’t like it
- Merkle then found more sympathetic Stanford collaborators named Diffie and Hellman
**Box and locks analogy**

- Alice wants to send Bob a gift in a locked box
  - They don't share a key
  - Can't send key separately, don't trust UPS
  - Box locked by Alice can't be opened by Bob, or vice-versa

**Math perspective: physical locks commute**

---

**Protocol with clip art**

---

**Public key primitives**

- Public-key encryption (generalizes block cipher)
  - Separate encryption key $EK$ (public) and decryption key $DK$ (secret)
- Signature scheme (generalizes MAC)
  - Separate signing key $SK$ (secret) and verification key $VK$ (public)

**Modular arithmetic**

- Fix modulus $n$, keep only remainders mod $n$
  - $\mod 12$: clock face; $\mod 2^{32}$: unsigned int
  - $+$, $-$, and $\times$ work mostly the same
- Division: see Exercise Set 1
- Exponentiation: efficient by square and multiply
Generators and discrete log

- Modulo a prime $p$, non-zero values and $\times$ have a nice ("group") structure
- $g$ is a generator if $g^0, g, g^2, g^3, \ldots$ cover all elements
- Easy to compute $x \mapsto g^x$
- Inverse, discrete logarithm, hard for large $p$

Diffie-Hellman key exchange

- Goal: anonymous key exchange
- Public parameters $p, g$; Alice and Bob have resp. secrets $a, b$
- Alice $\rightarrow$ Bob: $A = g^a \pmod{p}$
- Bob $\rightarrow$ Alice: $B = g^b \pmod{p}$
- Alice computes $B^a = g^{ba} = k$
- Bob computes $A^b = g^{ab} = k$

Relationship to a hard problem

- We're not sure discrete log is hard (likely not even NP-complete), but it's been unsolved for a long time
- If discrete log is easy (e.g., in P), DH is insecure
- Converse might not be true: DH might have other problems

Categorizing assumptions

- Math assumptions unavoidable, but can categorize
- E.g., build more complex scheme, shows it's "as secure" as DH because it has the same underlying assumption
- Commonly "decisional" (DDH) and "computational" (CDH) variants

Key size, elliptic curves

- Need key sizes $\sim 10$ times larger then security level
- Attacks shown up to about 768 bits
- Elliptic curves: objects from higher math with analogous group structure
  - (Only tenuously connected to ellipses)
- Elliptic curve algorithms have smaller keys, about $2 \times$ security level

RSA setup

- Choose $n = pq$, product of two large primes, as modulus
- $n$ is public, but $p$ and $q$ are secret
- Compute encryption and decryption exponents $e$ and $d$ such that
  \[ M^{ed} = M \pmod{n} \]
RSA encryption

- Public key is \((n, e)\)
- Encryption of \(M\) is \(C = M^e \pmod{n}\)
- Private key is \((n, d)\)
- Decryption of \(C\) is \(C^d = M^{ed} = M \pmod{n}\)

RSA signature

- Signing key is \((n, d)\)
- Signature of \(M\) is \(S = M^d \pmod{n}\)
- Verification key is \((n, e)\)
- Check signature by \(S^e = M^{de} = M \pmod{n}\)
- Note: symmetry is a nice feature of RSA, not shared by other systems

RSA and factoring

- We're not sure factoring is hard (likely not even NP-complete), but it's been unsolved for a long time
- If factoring is easy (e.g., in P), RSA is insecure
- Converse might not be true; RSA might have other problems

Homomorphism

- Multiply RSA ciphertexts \(\Rightarrow\) multiply plaintexts
- This homomorphism is useful for some interesting applications
- Even more powerful: fully homomorphic encryption (e.g., both + and \(\times\))
  - First demonstrated in 2009; still very inefficient

Problems with vanilla RSA

- Homomorphism leads to chosen-ciphertext attacks
- If message and \(e\) are both small compared to \(n\), can compute \(M^{1/e}\) over the integers
- Many more complex attacks too

Hybrid encryption

- Public-key operations are slow
- In practice, use them just to set up symmetric session keys
  - Only pay RSA costs at setup time
  - Breaks at either level are fatal

Padding, try #1

- Need to expand message (e.g., AES key) size to match modulus
- PKCS#1 v. 1.5 scheme: prepend 00 01 FF FF .. FF
- Surprising discovery (Bleichenbacher '98): allows adaptive chosen ciphertext attacks on SSL
  - Variants recurred later (c.f. "ROBOT" 2018)

Modern “padding”

- Much more complicated encoding schemes using hashing, random salts, Feistel-like structures, etc.
- Common examples: OAEP for encryption, PSS for signing
- Progress driven largely by improvement in random oracle proofs
Simpler padding alternative

- "Key encapsulation mechanism" (KEM)
- For common case of public-key crypto used for symmetric-key setup
  - Also applies to DH
- Choose RSA message $r$ at random mod $n$, symmetric key is $H(r)$
  - Hard to retrofit, RSA-KEM insecure if $e$ and $r$ reused with different $n$

Post-quantum cryptography

- One thing quantum computers would be good for is breaking crypto
- Square root speedup of general search
  - Countermeasure: double symmetric security level
- Factoring and discrete log become poly-time
  - DH, RSA, DSA, elliptic curves totally broken
  - Totally new primitives needed (lattices, etc.)
- Not a problem yet, but getting ready

Box and locks revisited

- Alice and Bob's box scheme fails if an intermediary can set up two sets of boxes
  - Man-in-the-middle (or middleperson) attack
- Real world analogue: challenges of protocol design and public key distribution

Next time

- Building crypto into more complex protocols
- Failures of cryptosystems
- Toward more paranoid crypto design