CSci 5271
Introduction to Computer Security
Cryptography, symmetric and public-key

Stephen McCamant
University of Minnesota, Computer Science & Engineering

Outline

Certificational attacks

- Good primitive claims no attack more effective than brute force
- Any break is news, even if it's not yet practical
  - Canary in the coal mine
- E.g., $2^{126.1}$ attack against AES-128
- Also watched: attacks against simplified variants

Fundamental ignorance

- We don't really know that any computational cryptosystem is secure
- Security proof would be tantamount to proving $\mathrm{P} \neq \mathrm{NP}$
- Crypto is fundamentally more uncertain than other parts of security

Relative proofs

- Prove security under an unproved assumption
- In symmetric crypto, prove a construction is secure if the primitive is
  - Often the proof looks like: if the construction is insecure, so is the primitive
- Can also prove immunity against a particular kind of attack

Random oracle paradigm

- Assume ideal model of primitives: functions selected uniformly from a large space
  - Anderson: elves in boxes
- Not theoretically sound; assumption cannot be satisfied
- But seems to be safe in practice

Pseudorandomness and distinguishers

- Claim: primitive cannot be distinguished from a truly random counterpart
  - In polynomial time with non-negligible probability
- We can build a distinguisher algorithm to exploit any weakness
- Slightly too strong for most practical primitives, but a good goal

Open standards

- How can we get good primitives?
- Open-world best practice: run competition, invite experts to propose then attack
- Run by neutral experts, e.g. US NIST
- Recent good examples: AES, SHA-3

A certain three-letter agency

- National Security Agency (NSA): has primary responsibility for "signals intelligence"
- Dual-mission tension:
  - Break the encryption of everyone in the world
  - Help US encryption not be broken by foreign powers

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

Stream cipher assessment

- Currently out of fashion as a primitive in software
- Not inherently insecure
  - Other common pitfall: must not reuse key(stream)
- Currently no widely vetted primitives

Outline

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

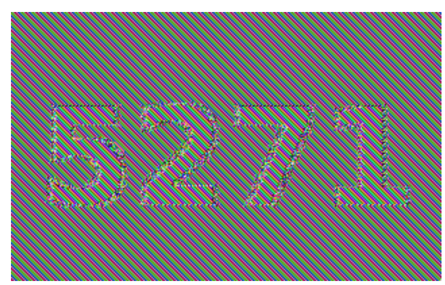
Modes of operation

- 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

Crypto primitive question

Which of these is a cryptographic primitive based on a Feistel cipher design?

A. DES
B. AES
C. DSA
D. CBC
E. HMAC

Last week's midterm

- Handing back in class today
  - May bring leftovers tomorrow, safest is my office hours
- Solution set will be available later
- +12 point adjustment to compensate for excessive difficulty
  - Visible on Canvas, not shown on paper exams

Midterm raw stem display

```
9 | 9
8 | 111345
7 | 01112335
6 | 0011233344557899
5 | 01135556666677889
4 | 13
3 | 19
2 | 67
```

Midterm adjusted stem display

```
9 | 333567
8 | 01123334557
7 | 0012233455566779
6 | 23357778888899
5 | 135
4 | 3
3 | 89
```

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 $a = \mathtt{0x5c}^*$, $b = \mathtt{0x36}^*$
- Probably the most widely used MAC

Outline

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

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

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

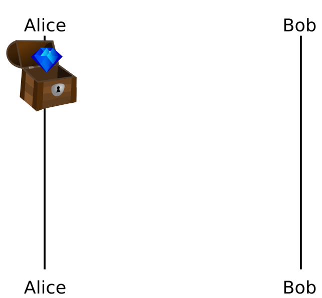
Box and locks analogy

- 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

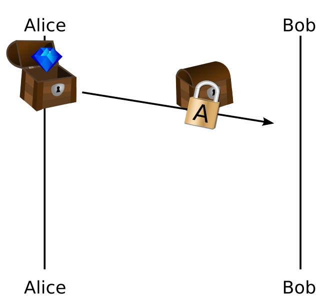
Box and locks analogy

- 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

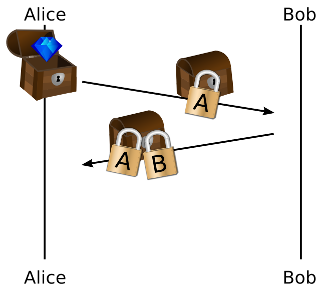
Protocol with clip art



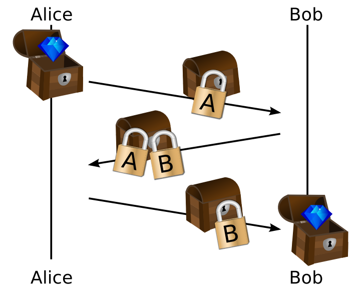
Protocol with clip art



Protocol with clip art



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→Bob: $A = g^a \pmod{p}$
- Bob→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 ~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× security level

Outline

General description

- 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)

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 ⇒ 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

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

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