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
- $H(K \|$ $M)$: insecure due to length extension
- $H(M \|$ $K)$, $H(K \|$ $M \|$ $K)$
- HMAC: $H(K \oplus \alpha \|$ $H(K \oplus \beta \|$ $M))$
- Standard $\alpha = \text{0x5c}$, $\beta = \text{0x36}$
- Probably the most widely used MAC

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
- MACs
- Building a secure channel
- Announcements intermission
- Public-key crypto basics
- Public key encryption and signatures

Midterm 2 is next Tuesday
- Similar in format to midterm 1
  - Any paper materials OK, but no electronics
  - Covers OS security, web security, and crypto up through this point in the lecture
  - Past exams (and later, solutions) on public site

Outline
- MACs
- Building a secure channel
- Announcements intermission
- 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
- Math perspective: physical locks commute

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
- \(+, -, \times \) work mostly the same
- Division? Multiplicative inverse by extended GCD
- 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 \( g^0, g^1, g^2, g^3 \ldots \) 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: \( A = g^a \mod p \)
- Bob \( \rightarrow \) Alice: \( B = g^b \mod p \)
- Alice computes \( B^a = g^{ab} = k \)
- Bob computes \( A^b = g^{ba} = k \)

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 \( \sim 10 \times \) 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

Outline
- MACs
- Building a secure channel
- Announcements intermission
- Public-key crypto basics
- Public key encryption and signatures

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 \( \Rightarrow \) 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 challenging

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
- Middleperson (man-in-the-middle) attack
- Real world analogue: challenges of protocol design and public key distribution