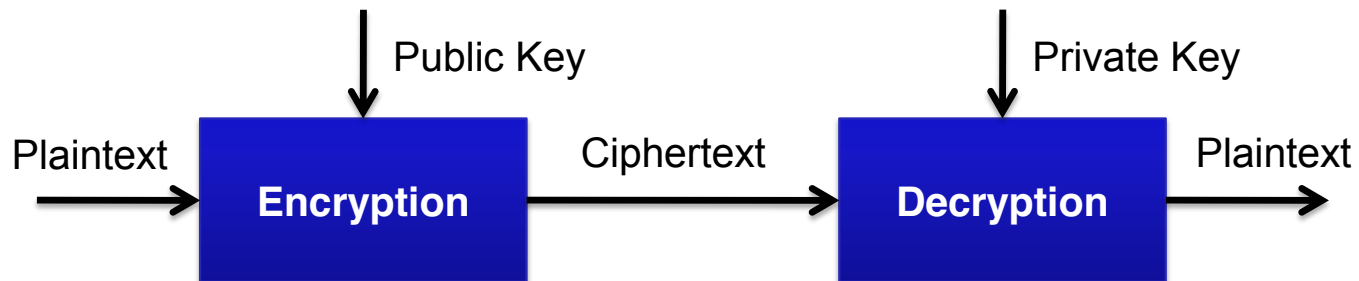


Public Key Cryptography



Public Key Cryptography

- Symmetric Key:
 - Same key used for encryption and decryption
 - Same key used for message integrity and validation
- Public-Key Cryptography
 - Use one key to encrypt or sign messages
 - Use another key to decrypt or validate messages
- Keys
 - Public key known to the world and used to send you a message
 - Only your private key can decrypt the message



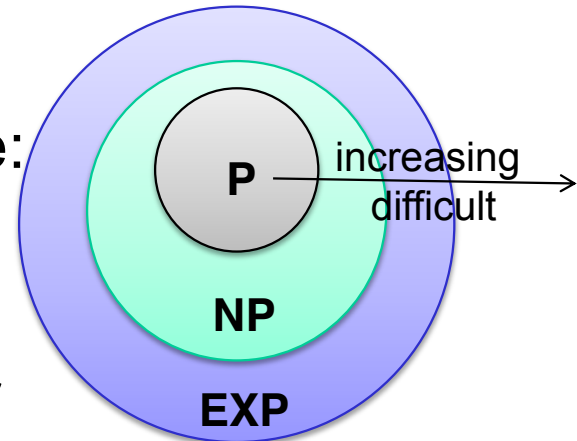
Public Key Cryptography

- Motivations
 - In symmetric key cryptography, a key was needed between every pair of users wishing to securely communicate
 - $O(n^2)$ keys
 - Problem of establishing a key with remote person with whom you wish to communicate
- Advantages to Public Key Cryptography
 - Key distribution much easier: everyone can know your public key as long as your private key remains secret
 - Fewer keys needed
 - $O(n)$ keys
- Disadvantages
 - Slow, often up to 1000x slower than symmetric-key cryptography



Cryptography and Complexity

- Three classes of complexity:
 - P: solvable in polynomial time, $O(n^c)$
 - NP: nondeterministic solutions in polynomial time, deterministic solutions in exponential time
 - EXP: exponential solutions, $O(c^n)$
- Cryptographic problems should be:
 - Encryption should be P
 - Decryption should be P with key
 - Decryption should be NP for attacker
- Need problems where complexity of solution depends on knowledge of a key



Modular Arithmetic Review

- Integers modulo prime p form an algebraic ring
- Example:
 - $\mathbb{Z} \pmod{7} = \{0, 1, 2, 3, 4, 5, 6\}$
 - Addition: $4 + 5 = 9 = 2 \pmod{7}$
 - Multiplication: $4 * 5 = 20 = 6 \pmod{7}$
 - Additive Identity: $4 + 0 = 4 \pmod{7}$
 - Multiplicative Identity: $4 * 1 = 4 \pmod{7}$
 - Inverse: $4 * 2 = 8 = 1 \pmod{7}$
 - $4^{-1} = 2 \pmod{7}$
 - $2^{-1} = 4 \pmod{7}$
 - Can use Euclidean Algorithm to find inverses \pmod{p} in polynomial time



Knapsack Problem

- Finding subset of items that completely fill a knapsack
- Cast mathematically, find a binary selection vector v_i such that:
$$\sum_i v_i a_i = T$$
- Vector a_i represents the size of the items and , and T is the total size of the knapsack
- Example:
 - $a = \{5, 8, 2, 9, 11, 4\}$
 - $T = 14$
 - Solution: $v = \{0, 1, 1, 0, 0, 1\}$



Knapsack Problem

- Finding vector v for an arbitrary knapsack is an NP problem
 - Deterministic exponential solution: try every vector 2^n
 - More efficient: recursive algorithm on sorted knapsack
- Superincreasing knapsack:
 - Special case where $a_n > \sum_{i=1}^{n-1} a_i$
 - Polynomial-time solution exists
- Example:
 - $a = \{1, 3, 6, 13, 25, 51\}$
 - $T = 32$
 - Solution
 - Can't have 51
 - Must have 25, result is 7
 - Can't have 13,
 - Must have 6, result is 1, etc



Knapsack Problem

- Use knapsack problem for cryptography
 - Plaintext is vector v
 - Ciphertext is target T
 - Key is vector a
- Need two equivalent knapsacks
 - Regular knapsack for encryption, k_e (public key)
 - Superincreasing knapsack for decryption k_d (private key)
 - Need a way to convert a superincreasing knapsack to a regular knapsack
 - Technique: use modular arithmetic
 - $k_e = c k_d \pmod{n}$



Knapsack Problem

- Example:
 - $k_d = \{1, 3, 6, 13, 25, 53\}$
 - $k_e = 51 k_d \pmod{107} = \{51, 46, 92, 21, 98, 28\}$
 - Message $M = \{0, 1, 1, 0, 1, 1\}$
 - Ciphertext $T = 264$
 - Decrypt using k_d
 - Need to “undo” multiplication by 51 (mod 107), use Euclidean algorithm to determine that $51 * 21 \pmod{107} = 1$, so $21 = 51^{-1}$
 - Compute new ciphertext $T' = 264 * 21 \pmod{107} = 87$
 - Must have 53, result is 34
 - Must have 25, result is 9
 - Cannot have 13
 - Must have 6, 3, result is $\{0, 1, 1, 0, 1, 1\}$



Knapsack Problem

- Proposed in 1978 as a public-key encryption scheme
- Analysis in 1983 showed flaws
 - Heuristic techniques for determining multiplier and modulus
 - Results in a polynomial-time algorithm to derive k_d from k_e
 - Flaw means that cryptosystems based on transforming a superincreasing knapsack are insecure



RSA

- Rivest-Shamir-Adleman
- Also introduced in 1978
- Based on the difficulty of factoring a large composite number into two large primes
 - Believed to be an exponential-time problem
 - Polynomial-time algorithms exist for Quantum computers
- Relies on generalization of Fermat's theorem:

$$x^{\varphi(n)} = 1 \pmod{n}$$

- $\varphi(n)$ is the number of numbers less than n , coprime with n
 - Euler's Totient Function
 - For $n = p$, $\varphi(n) = n-1$, for any prime p
 - For $n = pq$, $\varphi(n) = \varphi(p) \varphi(q) = (p-1)(q-1)$, for any primes p, q



RSA

- Uses modular arithmetic, for plaintext P , ciphertext C

$$C = P^e \pmod{n} \quad P = C^d \pmod{n}$$

- Need values d , e , n to make it work
- Using Fermat's Theorem:
 - Let $n = pq$ for primes p , q , test for prime is polynomial
 - Let $d = e^{-1} \pmod{\phi(n)}$, Euclidean algorithm is polynomial
- Then:
$$\begin{aligned} P &= C^d \pmod{n} \\ &= (P^e)^d \pmod{n} \\ &= P^{ed} \pmod{n} \\ &= P^1 \pmod{n} \end{aligned}$$



RSA

- Direct Attack
 - Attacker needs to be able to compute “Discrete Logarithm”
 - That is, $C = P^e \pmod n$
 - If C, e, n known, compute P
 - $\log_P(C) = e \pmod n$
 - Solving in R is easy, but in $Z \pmod n$ is EXP
- Rather than attack directly, try to find private key
 - Adversary needs to know $\phi(n)$ to compute d from e
 - To know $\phi(n)$, attacker must know p, q used to compute n
 - Attack requires factorization



RSA

- Security of RSA
 - Used in nearly every secure transaction over the Internet
 - Originally n was 512 bits (RSA-512)
 - Now crackable in under a year on a standard desktop computer
 - Roughly equivalent to DES
 - Most current Internet sites use RSA-1024
 - Infeasible to crack given current processing power
 - Most new standards and systems recommend RSA-2048
 - RSA-2048 keys are as difficult to crack as AES-128



El Gamal Encryption

- RSA can be cracked either by:
 - Solving Discrete Logarithm (DL) problem
 - Factoring public key
- Factoring is easier
- Need a cryptosystem that doesn't involve factoring, and based solely on DL problem
- Result would be more secure
 - Shorter key length for the same level of security
- Invented by El Gamal in 1984



El Gamal Encryption

- Use multiplicative group of integers (mod p)
 - Any algebraic group will work
- Key generation
 - Select prime p , integers a , $x < p$ / private key $\{x\}$
 - Compute $r = a^x \pmod{p}$ / public key $\{p, a, r\}$
- Encryption
 - Select random integer $y < p$
 - Compute $c_1 = a^y$, $c_2 = Mr^y$ / ciphertext $\{c_1, c_2\}$
- Decryption
 - Compute plaintext $= c_1^{-x} c_2$
 - $c_1^{-x} c_2 = (a^y)^{-x} Mr^y = M a^{-xy} (a^x)^y = M a^{-xy} a^{xy} = M \pmod{p}$



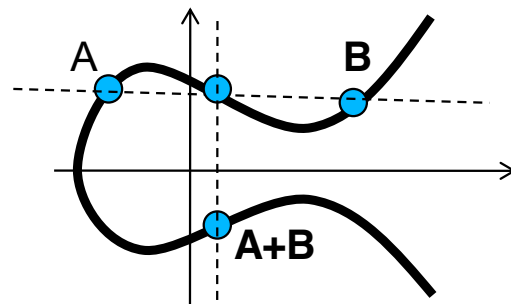
El Gamal Encryption

- Basic security provided by the discrete logarithm problem
- Other attacks: security actually limited
 - Computational Diffie-Hellman problem
 - Decisional Diffie-Hellman problem
 - Will discuss these in detail next week
- System is malleable
 - Example: adversary can change $c_2' = 2c_2$
 - Adversary decrypts $c_1^{-x} c_2' = 2M$
 - Deterministic change to ciphertext yields deterministic change in plaintext
 - Still need integrity protection



Elliptic Curve Cryptography

- Elliptic curves can be used to create an algebraic group
- Combined with El Gamal Encryption to perform Elliptic Curve Cryptography
- Basic idea:
 - Points on a curve are group elements
 - Can be “added together” by:
 - Find third point colinear with first two
 - Reflect across axis
 - Efficient algorithm exists for computation
 - Exponentiation: Compute $c * A$, where c is an integer constant, as $c * A = A + A + A + \dots + A$ (c times)
 - Forms an algebraic group with difficult discrete logarithm problem



Elliptic Curve Cryptography

- Advantages
 - Security bounded by DL problem rather than factoring problem
 - Can use significantly shorter key sizes
 - ECC-160 roughly equivalent to RSA-1024
 - MUCH shorter key sizes, better for storage, transmission
 - Still secure even if someone finds polynomial time for factoring integers
 - As RSA keys get longer, equivalently-secure ECC is more efficient in both hardware and software
- Disadvantages
 - Less institutionalized, most certificates don't support it
- Future of ECC
 - Patents by Certicom discourage use, expiring soon
 - USG pushing for use within USG systems



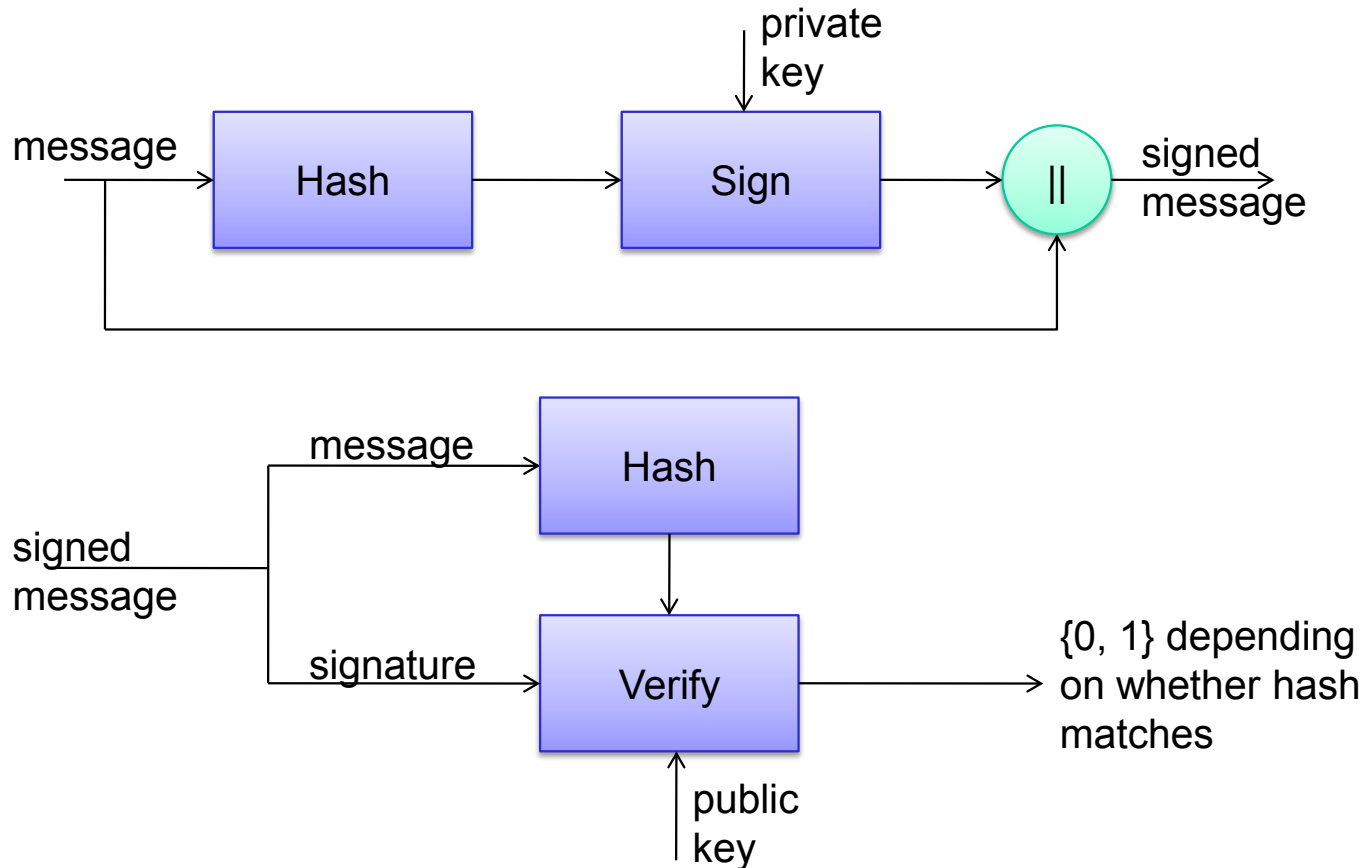
Digital Signatures

- Before, MIC provided message integrity
- Need a public-key equivalent
- Basic approach:
 - Most public-key systems have interchangeable keys
 - RSA: could use either d or e to encrypt or decrypt, one undoes the other
 - Compute a hash of the message, and “encrypt” it with the private key
 - Recipient “decrypts” with the public key, verifies the hash



Digital Signatures

- Overall Architecture:



RSA Digital Signatures

- As mentioned before: simply “encrypt” with the private key
 - M = Message, S = Signature
 - To sign: $S = \text{Hash}(M)^d \pmod{n}$
 - To verify, see if $\text{Hash}(M) = S^e \pmod{n}$
- Relies on security of hash function
 - If a collision can be found, an attacker can change M to M' such that $\text{Hash}(M) = \text{Hash}(M')$
 - Same signature S would be valid for both M and M'



Digital Signature Algorithm (DSA)

- DSA is NIST standard for digital signatures
- Based on El Gamal signature scheme
 - Similar to El Gamal Encryption
 - Relies on DL problem rather than factorization
- Key Generation:
 - Select prime p , integers $a, x < p$ / private key = $\{x\}$
 - Compute $y = a^x \pmod{p}$ / public key = $\{p, a, y\}$

- Signature:
 - Select random integer $k < p-1$
 - Compute $r = a^k \pmod{p}$
 - Compute $s = k^{-1} (\text{Hash}(M) - xr) \pmod{p-1}$
 - Signature: $\{r, s\}$

- Verify:
 - Compute $v = y^r r^s \pmod{p}$
 - Determine if $v = a^{\text{Hash}(M)} \pmod{p}$

$$\begin{aligned} y^r r^s &= (a^x)^r r^{(k^{-1} (\text{Hash}(M) - xr))} \pmod{p} \\ &= a^{xr} (a^k)^{(k^{-1} (\text{Hash}(M) - xr))} \pmod{p} \\ &= a^{(xr + k k^{-1} (\text{Hash}(M) - xr))} \pmod{p} \\ &= a^{xr + \text{Hash}(M) - xr} \pmod{p} \\ &= a^{\text{Hash}(M)} \pmod{p} \end{aligned}$$



Digital Signature Algorithm

- DSA can also be used with Elliptic Curve Group rather than multiplicative integers
 - Called ECDSA
 - Again requires shorter key for equivalent security
 - Based on El Gamal Signatures
- Most digital signature systems use DSA rather than RSA signatures
- Very few use ECDSA



Quantum Cryptography

- Drastically different than mathematical cryptography explored so far
- Encodes data as photons of light
- Photons can spin in different orientations: $\Rightarrow \uparrow \searrow \nearrow$
- Polarized filters can detect photons
 - + filter: detects $\Rightarrow \uparrow$ correctly, $\searrow \nearrow$ randomly
 - X filter: detects $\searrow \nearrow$ correctly, $\Rightarrow \uparrow$ randomly
- Sender assigns 0 to \Rightarrow or \searrow , 1 to \uparrow or \nearrow
- Sender's message {0, 1, 1, 0, 1} to $\{\searrow, \uparrow, \nearrow, \searrow, \uparrow\}$
- Receiver uses random filters to detect {+, +, X, X, X}
- Receiver detects {1, 1, 1, 0, 1} (first, last filters incorrect)
- Receiver sends filter list to sender, sender indicates which were correct
- Receiver now correctly knows {?, 1, 1, 0, ?}
- Use error-correcting code to communicate over channel



Quantum Cryptography

- Security is based on the Heisenberg Uncertainty Principle
 - If you measure the rotation of a photon, you randomly change the rotation
 - Sender/Receiver could detect statistically abnormal error rate in the channel
- Implementation issues
 - Currently difficult to send exactly one photon of light
 - Approaches use a laser and attenuate the output such that statistically the expected number of photons is 1 per bit
- Applications
 - Doesn't rely on DL or factorization, therefore immune to Quantum Cryptanalysis; may be one of the only viable cryptosystems
 - Currently geared toward satellite communications



Public-Key Cryptosystems (PKCS)

- PKCS encapsulations
 - RSA has defined proprietary encapsulations of data into encrypted, signed blobs
 - PKCS #1, #2, etc, defined different encodings
 - Some offer encryption, others signatures, or both
- Transaction Layer Security (TLS)
 - Fundamental basis of secure communications over the Internet
 - Uses RSA, etc, for key agreement (discuss in detail next week)
- Email standards
 - CMS (Cryptographic Message Syntax) used for SMIME, use RSA/DSS
 - PGP and GPG are commonly used for email encryption, use El Gamal



Public Key Infrastructure

- ANSI X.509 standards
 - Define how to format public keys for exchange over networks
 - Major use: definition of certificate format
- Certificates are public keys signed by an external authority
 - e.g. Verisign
 - Trusted third party, called Certificate Authority (CA)
- Prevents MITM attacks
 - Someone sends you a public key to communicate with them securely
 - How do you know it's really the public key of the person you want to communicate with?
 - Have a trusted third party sign the key as actually being owned by someone
 - Anyone can create a CA, but popular software applications only list major companies, others have to be added manually

