Lecture for February 3, 2016

ECS 235A

UC Davis

Matt Bishop

Public Key Cryptography

- Two keys
  - *Private key* known only to individual
  - *Public key* available to anyone
    - Public key, private key inverses
- Idea
  - Confidentiality: encipher using public key, decipher using private key
  - Integrity/authentication: encipher using private key, decipher using public one

Requirements

1. It must be computationally easy to encipher or decipher a message given the appropriate key
2. It must be computationally infeasible to derive the private key from the public key
3. It must be computationally infeasible to determine the private key from a chosen plaintext attack

RSA

- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*

Background

- Totient function $\phi(n)$
  - Number of positive integers less than n and relatively prime to n
    - *Relatively prime* means with no factors in common with n
- Example: $\phi(10) = 4$
  - 1, 3, 7, 9 are relatively prime to 10
- Example: $\phi(21) = 12$
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

Algorithm

- Choose two large prime numbers p, q
  - Let $n = pq$; then $\phi(n) = (p-1)(q-1)$
  - Choose $e < n$ such that e is relatively prime to $\phi(n)$.
  - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n); private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$

Example: Confidentiality

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message HELLO (07 04 11 11 14)
  - $07^{17} \bmod 77 = 28$
  - $04^{17} \bmod 77 = 16$
  - $11^{17} \bmod 77 = 44$
  - $11^{17} \bmod 77 = 44$
  - $14^{17} \bmod 77 = 42$
- Bob sends 28 16 44 44 42

Example

- Alice receives 28 16 44 44 42
- Alice uses private key, $d = 53$, to decrypt message:
  - $28^{53} \bmod 77 = 07$
  - $16^{53} \bmod 77 = 04$
  - $44^{53} \bmod 77 = 11$
  - $44^{53} \bmod 77 = 11$
  - $42^{53} \bmod 77 = 14$
- Alice translates message to letters to read HELLO
  - No one else could read it, as only Alice knows her private key and that is needed for decryption

Example: Integrity/ Authentication

- Take $p = 7$, $q = 11$, so $n = 77$ and $\phi(n) = 60$
- Alice chooses $e = 17$, making $d = 53$
- Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
  - $07^{53} \bmod 77 = 35$
  - $04^{53} \bmod 77 = 09$
  - $11^{53} \bmod 77 = 44$
  - $11^{53} \bmod 77 = 44$
  - $14^{53} \bmod 77 = 49$
- Alice sends 35 09 44 44 49

Example

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- Bob translates message to letters to read HELLO
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

Example: Both

- Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)
  – Alice's keys: public (17, 77); private: 53
  – Bob's keys: public: (37, 77); private: 13
- Alice enciphers HELLO (07 04 11 11 14):
  – $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  – $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  – $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  – $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  – $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- Alice sends 07 37 44 44 14

Security Services

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key
- Authentication
  - Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner

More Security Services

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

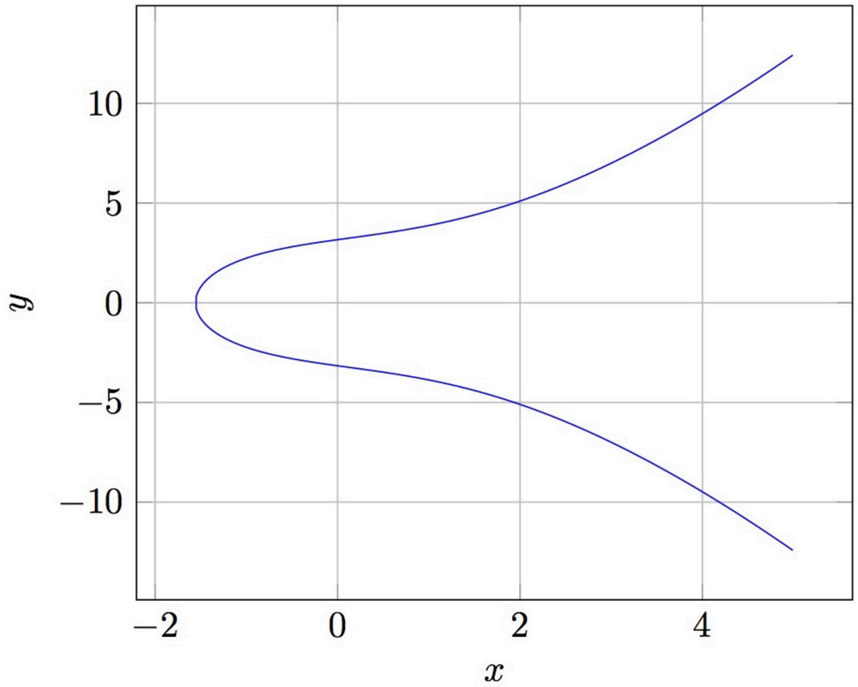
Warnings

- Encipher message in blocks considerably larger than the examples here
  - If 1 character per block, RSA can be broken using statistical attacks (just like classical cryptosystems)
  - Attacker cannot alter letters, but can rearrange them and alter message meaning
    - Example: reverse enciphered message of text ON to get NO

Elliptic Curve Ciphers

- $y^2 = x^3 + ax + b$
- Curve for

  $y^2 = x^3 + 4x + 10$



Addition on the Curve

- P_1, P_2 points on curve; draw line through them
  - If $P_1 = P_2$, use tangent
- Line intersects curve at $P_3 = (x_3, y_3)$
  - Define $P_4 = (x_3, -y_3)$ as sum of P_1, P_2
- Line doesn't intersect curve
  - Take $P_1 = (x, y)$; treat ∞ as point of intersection
  - Third point is $P_2 = (x, -y)$

Mathematically

- $P_1 = (x_1, y_1)$; $P_2 = (x_2, y_2)$
  - If $P_1 \neq P_2$, $m = (y_2 - y_1)/(x_2 - x_1)$
  - If $P_1 = P_2$, $m = (3x_1^2 + a)/2y_1$
- Define $P_3 = (x_3, y_3) = P_1 +_E P_2$, where
  - $x_3 = m^2 - x_1 - x_2$
  - $y_3 = m(x_1 - x_3) - y_1$
- Define $P_4 = -P_3 = (x_3, -y_3)$

A Hard Problem

- Use modular arithmetic, mod p prime

$$y^2 = x^3 + ax + b \bmod p$$

where $4a^3 + 27b^2 \neq 0$

- Let $Q = nP = P +_E \ldots +_E P$, n large
  - Generally computationally infeasible to find n given P and Q
- A version of the discrete log problem
  - Given b (mod p) and b^n (mod p), find n

Elliptic Curve Cryptosystem

- Parameters (a, b, p, P)
- Private key: randomly chosen integer $k < p$
  - In practice, this is less than the number of integer points on the curve
- Public key $K = kP$

ECC Version of Diffie-Hellman

- Curve is $y^2 = x^3 + 4x + 14 \bmod 2503$
  - Curve has 2477 integer points on it
- P = (1002, 493)
- $k_{\text{Alice}} = 1379$
  - Public key $K_{\text{Alice}} = k_{\text{Alice}}P \bmod p = (1041, 1659)$
- $k_{\text{Bob}} = 2011$
  - Public key $K_{\text{Bob}} = k_{\text{Bob}}P \bmod p = (629, 548)$

Communication

- Alice, Bob want to derive common key
- Bob computes:
  - $k_{\text{Bob}}K_{\text{Alice}} \bmod p = 2011(1041, 1659) \bmod 2503 = (2075, 2458)$
- Alice computes:
  - $k_{\text{Alice}}K_{\text{Bob}} \bmod p = 1379(629, 548) \bmod 2503 = (2075, 2458)$

About the Curves

- Parameters must be chosen carefully
  - Example: if $b = 0$, $p \bmod 4 = 3$, underlying (discrete log) problem much easier to solve
- Keys much shorter than non-ECC versions of cryptosystems
  - Computation times shorter
  - Example: ECC with key length of 246–383 bits gives same level of security as RSA with modulus 3072 bits

Cryptographic Checksums

- Mathematical function to generate a set of k bits from a set of n bits (where $k \leq n$).
  - k is smaller then n except in unusual circumstances
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is "parity"
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

Example Use

- Bob receives "10111101" as bits.
  - Sender is using even parity; 6 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly

Definition

- Cryptographic checksum h: $A \rightarrow B$:
  1. For any $x \in A$, $h(x)$ is easy to compute
  2. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
  3. It is computationally infeasible to find two inputs x, $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$
     - Alternate form (stronger): Given any $x \in A$, it is computationally infeasible to find a different $x' \in A$ such that $h(x) = h(x')$.

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - Pigeonhole principle: if there are n containers for $n+1$ objects, then at least one container will have 2 objects in it.
  - Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

Keys

- Keyless cryptographic checksum: requires no cryptographic key
  – SHA family (-2, -3, -256, -512, *etc*.) is best known; others include MD4, MD5 (both broken), HAVAL (-128 broken), SHA-0 (broken), SHA-1 (simplified version broken)
- Keyed cryptographic checksum: requires cryptographic key
  – HMAC version of keyless hash function

HMAC

- Make keyed cryptographic checksums from keyless cryptographic checksums
- h keyless cryptographic checksum function that takes data in blocks of b bytes and outputs blocks of l bytes. k' is cryptographic key of length b bytes
  - If short, pad with 0 bytes; if long, hash to length b
- *ipad* is 00110110 repeated b times
- *opad* is 01011100 repeated b times
- HMAC-$h(k, m) = h(k' \oplus opad \,\|\, h(k' \oplus ipad \,\|\, m))$
  - $\oplus$ exclusive or, $\|$ concatenation

Handling Keys

- Key exchange
  - Session vs. interchange keys
  - Classical, public key methods
  - Key generation
- Cryptographic key infrastructure
  - Certificates
- Key revocation
- Digital signatures

Notation

- $X \rightarrow Y : \{ Z \| W \} k_{X,Y}$
  - X sends Y the message produced by concatenating Z and W enciphered by key $k_{X,Y}$, which is shared by users X and Y
- $A \rightarrow T : \{ Z \} k_A \| \{ W \} k_{A,T}$
  - A sends T a message consisting of the concatenation of Z enciphered using k_A, A's key, and W enciphered using $k_{A,T}$, the key shared by A and T
- r_1, r_2 nonces (nonrepeating random numbers)

Session, Interchange Keys

- Alice wants to send a message m to Bob
  - Assume public key encryption
  - Alice generates a random cryptographic key k_s and uses it to encipher m
    - To be used for this message *only*
    - Called a *session key*
  - She enciphers k_s with Bob;s public key k_B
    - k_B enciphers all session keys Alice uses to communicate with Bob
    - Called an interchange *key*
  - Alice sends $\{ m \} k_s \{ k_s \} k_B$

Benefits

- Limits amount of traffic enciphered with single key
  - Standard practice, to decrease the amount of traffic an attacker can obtain
- Prevents some attacks
  - Example: Alice will send Bob message that is either "BUY" or "SELL". Eve computes possible ciphertexts { "BUY" } k_B and { "SELL" } k_B. Eve intercepts enciphered message, compares, and gets plaintext at once

Key Exchange Algorithms

- Goal: Alice, Bob get shared key
  - Key cannot be sent in clear
    - Attacker can listen in
    - Key can be sent enciphered, or derived from exchanged data plus data not known to an eavesdropper
  - Alice, Bob may trust third party
  - All cryptosystems, protocols publicly known
    - Only secret data is the keys, ancillary information known only to Alice and Bob needed to derive keys
    - Anything transmitted is assumed known to attacker

Classical Key Exchange

- Bootstrap problem: how do Alice, Bob begin?
  - Alice can't send it to Bob in the clear!
- Assume trusted third party, Cathy
  - Alice and Cathy share secret key k_A
  - Bob and Cathy share secret key k_B
- Use this to exchange shared key k_s

Simple Protocol

Alice $\xrightarrow{\{ \text{ request for session key to Bob } \}\, k_A}$ Cathy

Alice $\xleftarrow{\{ k_s \}\, k_A \,\|\, \{ k_s \}\, k_B}$ Cathy

Alice $\xrightarrow{\{ k_s \}\, k_B}$ Bob

Problems

- How does Bob know he is talking to Alice?
  - Replay attack: Eve records message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't
  - Session key reuse: Eve replays message from Alice to Bob, so Bob re-uses session key
- Protocols must provide authentication and defense against replay