Reminders

• Project 2 available. Due next week.

• Midterms (mean 71, stdev 14)

• Project 1 graded
  • 1 exploit 40 pts, 2 exploits 7 pts, 3 exploits 100 pts, 4 exploits 120 pts
Cryptography

• Symmetric key cryptography (secret key crypto): sender and receiver keys identical
• Asymmetric key cryptography (public key crypto): encryption key public, decryption key secret (private)
Applications of Public Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric key cryptography, must know secret key to encrypt
  - Only someone who knows private key can decrypt
  - Key management is simpler (maybe)
    - Secret is stored only at one site
- Digital signatures for authentication
  - Can “sign” a message with private key
- Session Key establishment
  - Exchange messages to create a special session key
  - Then use symmetric key cryptography
Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- Public info: p and g
  - p is a large prime number, g is a generator of $\mathbb{Z}_p^*$
  - $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}; \forall a \in \mathbb{Z}_p^* \exists i$ such that $a = g^i \mod p$
  - Modular arithmetic (numbers wrap around after they reach p)
    - $0 = p \mod p$

\[
\text{Compute } k = (g^y)^x = g^{xy} \mod p
\]
\[
\text{Compute } k = (g^x)^y = g^{xy} \mod p
\]
Modular arithmetic exercise

- Come up with addition and multiplication tables for integers mod 4
Why is Diffie-Hellman Secure?

- **Discrete Log (DL) problem:**
  given $g^x \mod p$, it’s hard to extract $x$
  - There is no known efficient algorithm for doing this
  - This is not enough for Diffie-Hellman to be secure! (Why?)

- **Computational Diffie-Hellman problem:**
  given $g^x$ and $g^y$, it’s hard to compute $g^{xy} \mod p$
  - … unless you know $x$ or $y$, in which case it’s easy

- **Decisional Diffie-Hellman (DDH) problem:**
  given $g^x$ and $g^y$, it’s hard to distinguish between $g^{xy} \mod p$ and $g^r \mod p$ where $r$ is random
Properties of Diffie-Hellman

• Assuming DDH problem is hard, Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  • Eavesdropper can’t distinguish between established key and a random value
  • Can use new key for symmetric cryptography
    • Approx. 1000 times faster than modular exponentiation

• Diffie-Hellman protocol (by itself) does not provide authentication
Diffie-Hellman Handshake

Alice $\rightarrow$ $E_{Bob}(g^x)$ $\rightarrow$ Bob

$g^y, H(K)$

K = $g^{xy}$

This depends on the hardness of discrete log
(hard to find x from $g^x$)

Now both sides have a symmetric key, K = $g^{xy}$,

Why do we need to encrypt $g^x$?
Why do we need $H(K)$?
What’s still broken?
Crypto is hard: Diffie-Hellman

Alice $\xrightarrow{E_{Bob}(g^x)}$ Mallory $\xrightarrow{E_{Bob}(g^0)}$ Bob

$g^0, H(1^y)$

$g^y, H(1^y)$

K = $g^{xy}$

Alice and Bob happily agree on K=1

What other keys are a problem?
Requirements for Public Key Crypto

- **Key generation:** computationally easy to generate a pair (public key PK, private key SK)
  - Computationally hard to obtain private key SK given only public key PK
- **Encryption:** given plaintext M and public key PK easy to compute ciphertext C = $E_{PK}(M)$
- **Decryption:** given ciphertext C = $E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to compute M from C without SK
  - Even infeasible to learn partial information about M
  - Trapdoor function: $\text{Decrypt}(SK, Encrypt(PK, M)) = M$
RSA: Number Theory

- Euler totient function $\phi(n)$, where $n \geq 1$, is the number of integers in the $[1,n]$ interval that are relatively prime to $n$
  - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
- Euler’s theorem: if $a \in \mathbb{Z}_n^*$, then $a^{\phi(n)} \equiv 1 \mod n$
- Special case: Fermat’s Little Theorem: if $p$ is prime and $\gcd(a,p) = 1$, then $a^{p-1} \equiv 1 \mod p$
RSA Cryptosystem

• **Key generation:**
  • Generate large primes p, q (1024 bits? 2048?) use primality test
  • Compute $n=pq$ and $\phi(n)=(p-1)(q-1)$
  • Choose small $e$, relatively prime to $\phi(n)$
    • Typically, $e=3$ or $e=2^{16}+1=65537$
  • Compute unique $d$ such that $ed = 1 \mod \phi(n)$
  • Public key = $(e,n)$; private key = $d$

• **Encryption of m:** $c = m^e \mod n$
  • Modular exponentiation by repeated squaring

• **Decryption of c:** $c^d \mod n = (m^e)^d \mod n = m$
Why Decryption Works

- $e \cdot d \equiv 1 \pmod{\phi(n)}$
- Thus $e \cdot d = 1 + k \cdot \phi(n) = 1 + k(p-1)(q-1)$ for some $k$
- Let $m$ be any integer in $\mathbb{Z}_n$
- If $\gcd(m, p) = 1$, then $m^{ed} \equiv m \pmod{p}$
  - By Fermat’s Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$
  - Raise both sides to the power $k(q-1)$ and multiply by $m$
  - $m^{1+k(p-1)(q-1)} \equiv m \pmod{p}$, thus $m^{ed} \equiv m \pmod{p}$
  - By the same argument, $m^{ed} \equiv m \pmod{q}$
- Since $p$ and $q$ are distinct primes and $p \cdot q = n$,
  - $m^{ed} \equiv m \pmod{n}$
Why is RSA Secure?

- RSA problem: given \( n=pq \), \( e \) such that \( \gcd(e,(p-1)(q-1))=1 \) and \( c \), find \( m \) such that \( m^e \equiv c \mod n \)
  - i.e., recover \( m \) from ciphertext \( c \) and public key \( (n,e) \) by taking \( e \)th root of \( c \)
  - There is no known efficient algorithm for doing this

- Factoring problem: given positive integer \( n \), find primes \( p_1, \ldots, p_k \) such that \( n=p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k} \)
  - If factoring is easy, then RSA problem is easy, but there is no known reduction from factoring to RSA
  - It may be possible to break RSA without factoring \( n \)
Caveats

- Don’t use RSA directly
- $e = 3$ is a common exponent
  - If $m < n^{1/3}$, then $c = m^3 < n$ and can just take the cube root of $c$ to recover $m$
    - Even problems if “pad” $m$ in some ways [Hastad]
- Let $c_i = m^3 \mod n_i$ - same message is encrypted to three people
  - Adversary can compute $m^3 \mod n_1n_2n_3$ (using Chinese Remainder Theorem)
  - Then take ordinary cube root to recover $m$
Integrity in RSA Encryption

• Plain RSA does not provide integrity
• Given encryptions of $m_1$ and $m_2$, attacker can create encryption of $m_1 \cdot m_2$
  • $(m_1e) \cdot (m_2e) \mod n = (m_1 \cdot m_2)e \mod n$
  • Attacker can convert $m$ into $m^k$ without decrypting
  • $(m^e)^k \mod n = (m^k)^e \mod n$
• In practice, OAEP is used: instead of encrypting $M$, encrypt $M \oplus G(r) \oplus H(M \oplus G(r))$
  • $r$ is random and fresh, $G$ and $H$ are hash functions
  • Resulting encryption is plaintext-aware: infeasible to compute a valid encryption without knowing plaintext
    • … if hash functions are “good” and RSA problem is hard
OAEP Encryption

- To encode,
  - messages are padded with $k_1$ zeros to be $n - k_0$ bits in length
  - $r$ is a random $k_0$ bit string
  - $G$ expands the $k_0$ bits of $r$ to $n - k_0$ bits.
  - $X = m00..0 \oplus G(r)$
  - $H$ reduces the $n - k_0$ bits of $X$ to $k_0$ bits.
  - $Y = r \oplus H(X)$
  - The output is $X || Y$ where $X$ is shown in the diagram as the leftmost block and $Y$ as the rightmost block.

- To decode,
  - recover random string as $r = Y \oplus H(X)$
  - recover message as $m00..0 = X \oplus G(r)$
Digital Signatures

**Given:** Everybody knows Bob’s public key
  Only Bob knows the corresponding private key

**Goal:** Bob sends a “digitally signed” message
  1. To compute a signature, must know the private key
  2. To verify a signature, enough to know the public key
Digital Signature Properties

• Authentication - “It’s really Bob that sent this”

• Nonrepudiation - “Bob can’t later claim he didn’t mean this”

• Integrity - “This is the thing Bob meant to send”
RSA Signatures

- Public key is (n,e), private key is d
- To sign message m: \( s = m^d \mod n \)
  - Signing and decryption are the same operation in RSA (not true for all schemes)
- It’s infeasible to compute s on m if you don’t know d
- To verify signature s on message m: \( s^e \mod n = (m^d)^e \mod n = m \)
  - Just like encryption
- Anyone who knows n and e (public key) can verify signatures produced with d (private key)
- In practice, also need padding & hashing (why?)
More on Signing

- Decryption not always signature
- Sign a hash not the message
- Signing a hash image with size equal to modulus is provably secure
Digital Signature Attacks

• Attack models (GMR)
  • key only (only public key)
  • known message (have some messages)
  • adaptive chosen message (can get chosen messages before attack)

• Attack Results
  • total break (recovery of signing key)
  • universal forgery (forge signatures in all messages)
  • selective forgery (adversary can create and sign some messages)
  • existential forgery (some valid but unchosen msg/signature pair created)

• Provably secure - No existential forgery under adaptive chosen message attack
Public Key Infrastructure (PKI)

- Only secure if binding between public key and owner is correct

- Approaches to verifying this
  - Hierarchical certificate authorities (x509)
  - Local trust model (SPKI/SDSI)
  - Web of trust (PGP/GPG)