CSE 484 (Winter 2008)

Applied Cryptography

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Asymmetric Cryptography
- CELT: Center for Engineering Learning and Teaching
- Reminder: Midterm on Friday. (Closed book.)
  - Contents up through the material for Monday (through symmetric crypto)
  - Not as hard as last year's midterm.
  - Make sure you understand the core concepts so far in this course:

Requirements for Public-Key Crypto

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
  - Computationally infeasible to determine 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: Decrypt(SK,Encrypt(PK,M))=M

Some Number Theory Facts ("Skip")

- Euler totient function $\varphi(n)$ 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 $a \in Z_n^*$, then $a^{\varphi(n)}=1 \bmod n$
- Special case: Fermat's Little Theorem
  if p is prime and gcd(a,p)=1, then $a^{p-1}=1 \bmod p$

RSA Cryptosystem ("Fast")[Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 1024 bits each (need primality testing, too)
  - Compute n=pq and $\varphi(n)=(p-1)(q-1)$
  - Choose small e, relatively prime to $\varphi(n)$
    - Typically, e=3 or $e=2^{16}+1=65537$ (why?)
  - Compute unique d such that ed = 1 mod $\varphi(n)$
  - Public key = (e,n); private key = d
- Encryption of m: $c = m^e \bmod n$
  - Modular exponentiation by repeated squaring
- Decryption of c: $c^d \bmod n = (m^e)^d \bmod n = m$

Why RSA Decryption Works ("Fast")

- $e \cdot d=1 \bmod \varphi(n)$
- Thus $e \cdot d=1+k \cdot \varphi(n)=1+k(p-1)(q-1)$ for some k

- Let m be any integer in Z_n
- If gcd(m,p)=1, then $m^{ed}=m \bmod p$
  - By Fermat's Little Theorem, $m^{p-1}=1 \bmod p$
  - Raise both sides to the power k(q-1) and multiply by m
  - $m^{1+k(p-1)(q-1)}=m \bmod p$, thus $m^{ed}=m \bmod p$
  - By the same argument, $m^{ed}=m \bmod q$
- Since p and q are distinct primes and $p \cdot q=n$, $m^{ed}=m \bmod n$

Why Is RSA Secure? ("Fast")

- RSA problem: given n=pq, e such that gcd(e,(p-1)(q-1))=1 and c, find m such that $m^e=c \bmod n$
  - 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

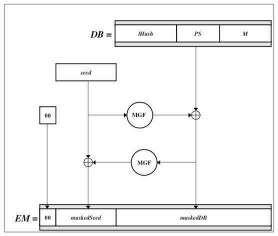
Caveats ("Fast;" Note first bullet)

- 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 $c_i = m^3 \bmod n_i$ - same message is encrypted to three people
    - Adversary can compute $m^3 \bmod n_1n_2n_3$ (using CRT)
    - Then take ordinary cube root to recover m

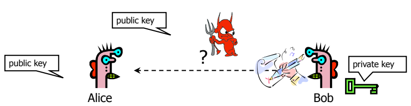
Integrity in RSA Encryption

- Plain RSA does <u>not</u> provide integrity
  - Given encryptions of m_1 and m_2, attacker can create encryption of $m_1 \cdot m_2$
    - $(m_1^e) \cdot (m_2^e) \bmod n = (m_1 \cdot m_2)^e \bmod n$
  - Attacker can convert m into m^k without decrypting
    - $(m_1^e)^k \bmod n = (m^k)^e \bmod n$
- In practice, OAEP is used: instead of encrypting M, encrypt $M \oplus G(r)$; $r \oplus H(M \oplus G(r))$
  - 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 (image from PKCS #1 v2.1)



Digital Signatures: Basic Idea



<u>Given</u>: Everybody knows Bob's public key
Only Bob knows the corresponding private key

<u>Goal</u>: 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

RSA Signatures

- Public key is (n,e), private key is d
- To sign message m: $s = m^d \bmod n$
  - Signing and decryption are the same operation in RSA
  - It's infeasible to compute s on m if you don't know d
- To verify signature s on message m:
  $s^e \bmod n = (m^d)^e \bmod n = m$
  - 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?)

Encryption and Signatures

- Books often say: Encryption and decryption are inverses, so use decryption as signatures
- That's a common view
  - True for the RSA primitive
- But not the cryptographic view
  - To really use RSA, we need padding
  - Some encryption schemes don't have natural signature analogs and vice versa.

Advantages of Public-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - No "chicken-and-egg" key establishment problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
    - Caveats to come
- Authentication without shared secrets
  - Use digital signatures to prove the origin of messages
- Reduce protection of information to protection of authenticity of public keys
  - No need to keep public keys secret, but must be sure that Alice's public key is <u>really</u> her true public key

Disadvantages of Public-Key Crypto

- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - We'll see this in IPSec and SSL
- Keys are longer
  - 1024 bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - What if factoring is easy?
    - Factoring is <u>believed</u> to be neither P, nor NP-complete
  - (Of course, symmetric crypto also rests on unproven assumptions)