CSE331: Introduction to Networks and Security

Lecture 20
Fall 2002
Announcements

• Reminder: Project 2 is due Monday, Oct. 28th

• Final Exam: Tuesday, Dec. 17
  – 8:30-10:30am
  – Room to be determined
Recap

• RSA

• Today:
  – Finish up RSA
  – Basic Security Protocols
Fermat’s Little Theorem

• Generalized by Euler.

• Theorem: If $\gcd(a,n) = 1$ then $a^{\phi(n)} \mod n = 1$.

• Easy to compute $a^{-1} \mod n$
  
  – $a^{-1} \mod n = a^{\phi(n)-1} \mod n$
  
  – Why? $a \ast a^{\phi(n)-1} \mod n$
    
    = $a^{\phi(n)-1+1} \mod n$
    
    = $a^{\phi(n)} \mod n$
    
    = 1
RSA Key Generation

• Choose large primes p and q.
  – Should be roughly equal length (in bits)
• Let n = p*q
• Choose a random encryption exponent e
  – With requirement: e and (p-1)*(q-1) are relatively prime.
• Derive the decryption exponent d
  – \( d = e^{-1} \mod ((p-1)*(q-1)) \)
  – d is e’s inverse mod ((p-1)*(q-1))
• Public key: K = (e,n) pair of e and n
• Private key: k = (d,n)
• Discard primes p and q (they’re not needed anymore)
RSA Encryption and Decryption

• Message: $m$
• Assume $m < n$
  – If not, break up message into smaller chunks
  – Good choice: largest power of 2 smaller than $n$

• Encryption: $E((e,n), m) = m^e \mod n$
• Decryption: $D((d,n), c) = c^d \mod n$
Proof that D inverts E

\[ c^d \mod n \]

\[ = (m^e)^d \mod n \quad \text{(definition of c)} \]

\[ = m^{ed} \mod n \quad \text{(arithmetic)} \]

\[ = m^{k*(p-1)*(q-1) + 1} \mod n \quad \text{(d inverts e)} \]

\[ = m^*m^{k*(p-1)*(q-1)} \mod n \quad \text{(arithmetic)} \]

\[ = m^*1 \mod n \quad \text{(Fermat)} \]

\[ = m \quad \text{(m < n)} \]
Chinese Remainder Theorem

• Suppose:
  – p and q are relatively prime
  – \( a \equiv b \pmod{p} \)
  – \( a \equiv b \pmod{q} \)

• Then: \( a \equiv b \pmod{pq} \)

• Proof:
  – p divides \((a-b)\) (because \( a \mod p = b \mod p \))
  – q divides \((a-b)\)
  – Since p, q are relatively prime, \( pq \) divides \((a-b)\)
  – But that is the same as: \( a \equiv b \pmod{pq} \)
Finished Proof

- Note: $m^{p-1} \equiv 1 \mod p$ (if $p$ doesn’t divide $m$)
- Implies: $m^{k \phi(n)+1} \equiv m \mod p$
  - Also holds for $m = a*p$
- Same argument yields: $m^{k \phi(n)+1} \equiv m \mod q$
- Chinese Remainder Theorem implies: $m^{k \phi(n)+1} \equiv m \mod n$
Cryptographic Protocols

• Consider communication over a network…
• What is the threat model?
  – What are the vulnerabilities?
What Can the Observer Do?

- Intercept them (confidentiality)
- Modify them (integrity)
- Fabricate other messages (integrity)
- Replay them (integrity)
- Block the messages (availability)
- Delay the messages (availability)
- Cut the wire (availability)
- Flood the network (availability)
Diffie-Hellman Key Exchange

• Problem with shared-key systems: Distributing the shared key
• Suppose that Alice and Bart want to agree on a secret (i.e. a key)
  – Communication link is public
  – They don’t already share any secrets
Diffie-Hellman by Analogy: Paint

1. Alice & Bart decide on a public color, and mix one liter of that color.
2. They each choose a random secret color, and mix two liters of their secret color.
3. They keep one liter of their secret color, and mix the other with the public color.
4. They exchange the mixtures over the public channel.

5. When they get the other person’s mixture, they combine it with their retained secret color.

6. The secret is the resulting color: Public + Alice’s + Bart’s
Diffie-Hellman Key Exchange

• Choose a prime $p$ (publicly known)
  – Should be about 512 bits or more
• Pick $g < p$ (also public)
  – $g$ must be a *primitive root* of $p$.
  – A primitive root generates the finite field $p$.
  – Every $n$ in \{1, 2, ..., $p$-1\} can be written as $g^k \bmod p$
  – Example: 2 is a primitive root of 5
    – $2^0 = 1 \quad 2^1 = 2 \quad 2^2 = 4 \quad 2^3 = 4 \pmod{5}$
    – Intuitively means that it’s hard to take logarithms base $g$ because there are many candidates.
1. Alice & Bart decide on a public prime \( p \) and primitive root \( g \).

2. Alice chooses secret number \( A \). Bart chooses secret number \( B \).

3. Alice sends Bart \( g^A \mod p \).

4. The shared secret is \( g^{AB} \mod p \).
Details of Diffie-Hellman

• Alice computes $g^{AB} \mod p$ because she knows A:
  – $g^{AB} \mod p = (g^B \mod p)^A \mod p$

• An eavesdropper gets $g^A \mod p$ and $g^B \mod p$
  – They can easily calculate $g^{A+B} \mod p$ but that doesn’t help.
  – The problem of computing discrete logarithms (to recover A from $g^A \mod p$ is hard.)
Example

• Alice and Bart agree that $q=71$ and $g=7$.
• Alice selects a private key $A=5$ and calculates a public key $g^A \equiv 7^5 \equiv 51 \pmod{71}$. She sends this to Bart.
• Bart selects a private key $B=12$ and calculates a public key $g^B \equiv 7^{12} \equiv 4 \pmod{71}$. He sends this to Alice.
• Alice calculates the shared secret: $S \equiv (g^B)^A \equiv 4^5 \equiv 30 \pmod{71}$
• Bart calculates the shared secret $S \equiv (g^A)^B \equiv 51^{12} \equiv 30 \pmod{71}$
Why Does it Work?

• Security is provided by the difficulty of calculating discrete logarithms.

• Feasibility is provided by
  – The ability to find large primes.
  – The ability to find primitive roots for large primes.
  – The ability to do efficient modular arithmetic.

• Correctness is an immediate consequence of basic facts about modular arithmetic.