Recap
Recap

• Number theory
  o What is a prime number?
  o What is prime factorization?
  o What is a GCD?
  o What does relatively prime mean?
    • What does co-prime mean?
  o What does congruence mean?
  o What is the additive inverse of 13 % 17?
  o What is the multiplicative inverse of 7 % 8?
Recap: Diffie-Hellman

- You’re trapped in your spaceship
- You have enough energy to send a single message to your HQ
- You have:
  - HQ’s public DH values
    - $g=5$, $p = 875498279345…$
    - $g^a = 32477230478…$
  - Your AES implementation from Labs #1 & 2
  - An arbitrary precision calculator
- How can you construct your message so that it will be safe from eavesdroppers?
Asymmetric Encryption
Public Key Terminology

- Public Key
- Private Key
- Digital Signature

- You encrypt with a public key, and you decrypt with a private key

- You sign with a private key, and you verify with a public key
Model for Encryption with Public Key Cryptography

Alice

Plaintext

Encryption Algorithm

Bob’s Public Key

Ciphertext

Decryption Algorithm

Plaintext

Bob

Bob’s Private Key
Model for Digital Signature with Public Key Cryptography

Alice

Alice's Private Key

Alice's Public Key

Bob

Plaintext

Signing Algorithm

Ciphertext

Verification Algorithm

Plaintext

Signing Algorithm

Ciphertext

Verification Algorithm
History of RSA

• Invented in 1977 by
  o Ron Rivest
  o Adi Shamir
  o Leonard Adleman

• Patent expired in the year 2000

• It’s withstood years of extensive cryptanalysis
  o Suggests a level of confidence in the algorithm
RSA

• $m =$ message
• $c =$ ciphertext
• $e =$ public exponent
• $d =$ private exponent
• $n =$ modulus

• RSA Encryption
  - $c = m^e \mod n$

• RSA Decryption
  - $m = c^d \mod n$
The Math Behind RSA

- RSA encrypt/decrypt operations are simple

- The math to get to the point where these operations work is not so simple (at first)
  - Fermat’s little theorem
  - Euler’s generalization of Fermat’s little theorem
Fermat’s Little Theorem

- If
  - p is prime
  - a is relatively prime to p
    - (co-prime)

- Then Fermat’s theorem states
  - $a^{p-1} \equiv 1 \pmod{p}$
  - for all $0 < a < p$

- This serves as the basis for
  - Fermat’s primality test
  - Euler’s generalization

Which values of $a$ aren’t co-prime to $p$?

Pierre de Fermat
(1601-1655)
Proof of Fermat’s Little Theorem

• Need this lemma
  o If \( xa \equiv ya \pmod{p} \), then \( x \equiv y \pmod{p} \)

• Proof of lemma
  o By contradiction

• Proof of theorem
  o Using lemma to create sets \( P \) and \( N \)
    o \( P = \{1a, 2a, 3a, ..., (p-1)a\} \)
    o \( N = \{1, 2, 3, ..., (p-1)\} \)
Application of Fermat’s Little Theorem

- Compute $2^{12} \mod 11$

- Compute $2^{565} \mod 561$ (561 is prime)
Euler’s Generalization of Fermat’s Little Theorem

• Euler said
  o \( a^{\phi(n)} \equiv 1 \pmod{n} \)

  *n doesn’t need to be prime*
  *a must still be co-prime to n*

• \( \phi(n) \)
  o Euler’s totient function \( \varphi(n) \)
  o The number of values less than \( n \) which are relatively prime to \( n \)
    • Multiplicative group of integers \( (\mathbb{Z}_n^*) \)

• RSA is interested in values of \( n \) that are the product of two prime numbers \( p \) and \( q \)
Computing \( \phi(n) \) in RSA

- \( \phi(n) \) is the number of integers between 0 and \( n \) that are co-prime to \( n \)
- When \( p \times q = n \), and \( p \) and \( q \) are prime, what is the \( \phi(n) \)?
- Proof (When \( p \times q = n \))

**Observations**
1) there are \( p-1 \) multiples of \( q \) between 1 and \( n \)
2) there are \( q-1 \) multiples of \( p \) between 1 and \( n \)
   These multiples are **not** co-prime to \( n \)

**Definition:**
\[ \phi(n) = \# \text{ of values between 0 and } n \text{ that are co-prime to } n \]
\[ \phi(n) = \# \text{ of values between 0 and } n \text{ minus } \# \text{ of values between 0 and } n \text{ not co-prime to } n \]

\[ \phi(n) = [n-1] - [(p-1) + (q-1)] \]
\[ = [pq-1] - (p-1) - (q-1) \]
\[ = pq - p - q + 1 \]
\[ = (p-1)(q-1) \]
RSA

- Euler said: $a^{\phi(n)} \equiv 1 \pmod{n}$
  - $m^{(p-1)(q-1)} \equiv 1 \pmod{n}$

- Notice: $m^{(p-1)(q-1)} \times m \equiv m^{(p-1)(q-1)+1} \equiv m \pmod{n}$
  - $m^{\phi(n)+1} \equiv m \pmod{n}$

- Let $e \times d = k \times \phi(n) + 1$
  - Then $e \times d \equiv 1 \pmod{\phi(n)}$
  - Therefore $m^{ed} \equiv m^{k \times \phi(n)+1} \equiv m^{\phi(n)} \times m^{\phi(n)} \times \ldots \times m \equiv m \pmod{n}$

- RSA Encryption
  - $m^e \equiv c \pmod{n}$

- RSA Decryption
  - $c^d \equiv m \pmod{n}$
Steps for RSA Encryption

• Select p, q (large prime numbers)
• n = p*q
• \( \phi(n) = (p-1)(q-1) \)

• Select integer e where e is relatively prime to \( \phi(n) \)
  o Common values for e are 3 and 65537. Why?
• Calculate d, where \( de = 1 \pmod{\phi(n)} \)

• Public key is KU = \{e, n\}
• Private key is KR = \{d, n\}

• RSA encryption
  o \( m^e = c \pmod{n} \)
• RSA decryption
  o \( c^d = m \pmod{n} \)

Why is RSA Secure?

• Hard to factor large numbers
• Hard to compute d without \( \phi(n) \)
• Discrete logs are hard (\( m^d \pmod{n} \))
• Given signature, hard to find d
RSA Usage

• Given $m^e = c \pmod{n}$ and $c^d = m \pmod{n}$
  o What restrictions should be placed on $m$?

• For bulk encryption (files, emails, web pages, etc)
  o Some try using RSA as block cipher
  o *Never, never, never* encrypt data directly using RSA
    • Inefficient
    • Insecure
  o Always use symmetric encryption for data, and use RSA to encrypt the symmetric key (after adding the appropriate padding)

• Digital signatures
  o Do not “sign” the entire document
  o “Sign” (encrypt) a hash of the document using the private key
    • Makes sure the length of the hash is $< n$
How do we get p, q, e, & d?

• What is p?
  o How do we get it?

• What is q?
  o How do we get it?

• What is e?
  o How do we get it?
  o What is the relationship of e and (p-1)(q-1)?

• What is d?
  o How do we get it?
Multiplicative Inverses

• Use the extended Euclidean algorithm
  o Based on the fact that GCD can be defined recursively
    • If $x > y$, then $\text{GCD}(x,y) =_{\text{(recursively)}} \text{GCD}(y, x-y)$
    • Also if $x > y$, then $\text{GCD}(x,y) =_{\text{(recursively)}} \text{GCD}(y, x\%y)$

  o GCD can also be used as follows:
    • $ax + by = \text{gcd}(x,y)$
    • If $x$ is the modulus, and $\text{gcd} (x,y) = 1$
      o Then $ax + by = 1$ and $b$ is $y^{-1}$
Extended Euclidean algorithm

\text{GCD} \ (120, \ 23) \\
120 \ / \ 23 = 5 \ r \ 5 \\
23 \ / \ 5 = 4 \ r \ 3 \\
5 \ / \ 3 = 1 \ r \ 2 \\
3 \ / \ 2 = 1 \ r \ 1 \\
2 \ / \ 1 = 2 \ r \ 0 \\
1 \ / \ 0 \hspace{1cm} \text{GCD is 1, 120 and 23 are co-prime}

\text{GCD} \ (120, \ 23) \\
120 \ / \ 23 = 5 \ r \ 5 \hspace{0.5cm} \Rightarrow \hspace{0.5cm} 5 = 120(1) + 23(-5) \\
23 \ / \ 5 = 4 \ r \ 3 \hspace{0.5cm} \Rightarrow \hspace{0.5cm} 3 = 23(1) + 5(-4) \\
5 \ / \ 3 = 1 \ r \ 2 \hspace{0.5cm} \Rightarrow \hspace{0.5cm} 2 = 5(1) + 3(-1) \\
3 \ / \ 2 = 1 \ r \ 1 \hspace{0.5cm} \Rightarrow \hspace{0.5cm} 1 = 3(1) + 2(-1) \\
2 \ / \ 1 = 2 \ r \ 0 \hspace{0.5cm} \Rightarrow \hspace{0.5cm} 0 = 2(1) + 1(-2)

Notice the first line is a sum of products involving 120 and 23. We can derive a formula for each remainder to be a sum of products of 120, 23.
Extended Euclidean algorithm

GCD (120, 23)
120 / 23 = 5 r 5  =>  5 = 120(1) + 23(-5)
23 / 5  = 4 r 3  =>  3 = 23(1) + 5(-4)
5  / 3  = 1 r 2  =>  2 = 5(1) + 3(-1)
3  / 2  = 1 r 1  =>  1 = 3(1) + 2(-1)
2  / 1  = 2 r 0  =>  0 = 2(1) + 1(-2)

GCD (120, 23)
120 / 23 = 5 r 5  =>  5 = 120(1) + 23(-5)
23 / 5  = 4 r 3  =>  3 = 23(1) + 5(-4)
5  / 3  = 1 r 2  =>  2 = 5(1) + 3(-1)
3  / 2  = 1 r 1  =>  1 = 3(1) + 2(-1)
2  / 1  = 2 r 0  =>  0 = 2(1) + 1(-2)
Extended Euclidean algorithm

\[
\begin{align*}
\text{GCD (120, 23)} \\
120 / 23 = 5 \text{ r } 5 & \quad \Rightarrow 5 = 120(1) + 23(-5) \\
23 / 5 = 4 \text{ r } 3 & \quad \Rightarrow 3 = 23(21) + 120(-4) \\
5 / 3 = 1 \text{ r } 2 & \quad \Rightarrow 2 = 5(1) + 3(-1) \\
3 / 2 = 1 \text{ r } 1 & \quad \Rightarrow 1 = 3(1) + 2(-1) \\
2 / 1 = 2 \text{ r } 0 & \quad \Rightarrow 0 = 2(1) + 1(-2)
\end{align*}
\]
Extended Euclidean algorithm

GCD (120, 23)

120 / 23 = 5 r 5  =>  5 = 120(1) + 23(-5)
23 / 5  = 4 r 3  =>  3 = 23(21) + 120(-4)
5  / 3  = 1 r 2  =>  2 = 120(5) + 23(-26)
3  / 2  = 1 r 1  =>  1 = 3(1) + 2(-1)
2  / 1  = 2 r 0  =>  0 = 2(1) + 1(-2)

GCD (120, 23)

120 / 23 = 5 r 5  =>  5 = 120(1) + 23(-5)
23 / 5  = 4 r 3  =>  3 = 23(21) + 120(-4)
5  / 3  = 1 r 2  =>  2 = 120(5) + 23(-26)
3  / 2  = 1 r 1  =>  1 = [23(21) + 120(-4)] +
                      [120(5) + 23(-26)](-1)
                      = 23(47) + 120(-9)
2  / 1  = 2 r 0  =>  0 = 2(1) + 1(-2)

Notice that 1 = 23*47 + 120(-9) means that
47 is the multiplicative inverse of 23 (mod 120)
Computing “d”

- For RSA, calculate $\text{GCD}(\phi(n), e)$ to find $d$ using extended Euclidean algorithm (see handout on Lab #4 page)
  - Manual iterative method for the exam
  - Use the table method in your lab

- For RSA, the $\text{GCD}(\phi(n),e)$ will result in an equation of the form
  - $1 = e \cdot d + \phi(n) \cdot k$
  - Where $d$ or $k$ is negative
    - If $d$ is negative convert it to an equivalent positive number (mod $n$) using $\phi(n) + d$
Computing RSA Keys Example

• $p=17$
• $q=11$
• $n=?$
• $\phi(n)=?$
• $e=7$
• $d=?$
## Applications for Public Key Cryptosystems

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Encrypt/Decrypt</th>
<th>Digital Signature</th>
<th>Key Exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffie-Hellman</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RSA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DSS</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Elliptic Curve</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>