The Mathematics Behind Public–Key Cryptography

Davenport University ITS Lunch and Learn
February 2, 2012
Sneden Center Meeting Hall
Presented by: Scott Radtke
Public–Key Cryptosystem

A discussion on the mathematics behind coding and decoding using RSA Public–Key Cryptography.
Take Away

- Have FUN with FUNctions
- Play with the Number Theory of Euclid and Euler
- Discuss this thing called “clock” arithmetic
- Understand how messages can be encoded and decoded
- Use a really old C program!
- Find out how a spreadsheet can be used as a calculation tool
- Leave with a newly found desire to become a mathematician
- Fill up on Pizza
Mathematical Review
  ◦ Working in different Bases
  ◦ Idea of Inverse Functions
  ◦ Numeric Representations of Character Sets

Encryption Systems Overview

The RSA Algorithm
  ◦ Modular Arithmetic
    • Creation of Public and Private Keys
    • The Euclidean Algorithm
    • Euler’s Theorem

Encryption of a message using Public Key
Decryption of a message using Private Key
Encrypted Signatures
Translating from one Base to another
Most humans are comfortable with Base 10 and don’t even think about how it works
  ◦ There are 10 placeholders in each column
  ◦ Recall $127_{10} = 1 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

What about Base 16 (Hexadecimal)
  ◦ 16 placeholders per column
  ◦ Convert an Hexadecimal number to Base 10
  ◦ $15FB_{16}=1(16^3) + 5(16^2) + 15(16^1) + 11(16^0) = 5627_{10}$

How to go from Base 10 to Base 16?
  ◦ $16|5627$
    ◦ $16|351$ r 11 (B)
    ◦ $16|21$ r 15 (F)
    ◦ $16|1$ r 5 (5)
    ◦ 0 r 1 (1)
  • Notice the importance of the remainder here
Inverse function examples

- Example: $X^2$ and $\sqrt{X}$
- Applying an inverse function to a result of a function gives you back what you started with
- Squares and Square Roots are inverse functions
  - $3^2 = 9$ so $\sqrt{9} = 3$
- Exponents and Logs are inverses: $x = b^y$ then $y = \log_b(x)$
  - $10^3 = 1000$ so $\log_{10} 1000 = 3$
  - $2^6 = 64$ so $\log_2 64 = 6$

Multiplicative Inverses

- $X$ and $Y$ are Multiplicative Inverses if $X \cdot Y = 1$
  - 5 and $1/5$ are multiplicative inverses.
Numerical Representations of Characters

- **ASCII**
  - American Standard Code for Information Interchange
  - Characters – 128 (Base 128)
  - 7bit – Base 2

- **Extended ASCII**
  - Characters – 256 (Base 256)
  - 8bit – Base 2

- We need to represent the message to be encoded as a number.
  - Computers use Base 2 numbers
  - Humans prefer Base 10
Scott’s Simplified Code for Information Interchange.
- Base 37
- 10 digits – 26 letters – underscore

We can use this to represent a plain text message as a numeric message for encryption, similar to ASCII.
Converting Plain Text

- We are going to use SSCII and Base 10 instead of ASCII and Base 2 for simplicity.
- We also keep our messages small so we can use a standard calculator.
  - No round-off is allowed.

Let's look at a simple message: ITS
- I = 18, T = 29, S = 28 (according to the SSCII table).
- \( ITS_{37} \rightarrow 18(37^2) + 29(37^1) + 28(37^0) = 25743 \)
- Our "plain text" message: \( ITS_{37} \) is equal to
- \( 25,743_{10} \) (numeric). This is the numeric message we will encrypt later.
Encryption Types
Symmetric vs. Asymmetric Systems

- Symmetric–key systems
  - Single key for coding and decoding
  - Easier to use
  - Difficult to distribute and keep key private
  - Our SSCII is an example of a Symmetric–Key system

- Asymmetric–key systems
  - Separate keys used for coding and decoding
  - Public Key used to encode
    - Public Key becomes a “trap–door”
  - Private Key used to decode
  - Simplified Key distribution (Pub–key is not secret)
  - Analogy – Send an open lockbox to someone and keep the key. You alone can unlock the box when returned.
Brian wants to send encrypted message to me
- I make my public-key known
- Brian encrypts the message using my public-key
- In order to read the message I must decrypt it using my private key

Let’s do this using the RSA Algorithm
The RSA Algorithm

- Introduced in 1977
- Contribution of three mathematicians
  - Ronald Rivest
  - Adi Shamir
  - Leonard Adleman
- Relies on classical number theory of Euclid and Euler
  - Euclid – Greek Mathematician ~300BC
    - Euclidean Algorithm
    - Euclid’s Elements
    - Euclidean Geometry
  - Euler – German Mathematician/Physicist ~ 1750
    - Euler’s Theorem – forms the basis of RSA encryption system
    - Uses modular or “clock” arithmetic
What is Modular Arithmetic?

- A clock is in mod12
  - So $13 \mod 12 = 1$
  - Or $28 \mod 12 = 4$
- Think of mod as giving you the remainder
  - Divide by the modulus, what is the remainder
  - $18 \mod 12 = 6$
  - Where else do we use modular arithmetic?
- To develop our encryption keys we need to calculate the Modulus we will use
Let’s Create our Keys

To create our keys we need 3 things

- A **Modulus**: $M = p(q)$ where $p$ and $q$ are prime numbers
- An Exponent ($E$) for our public key
  - Helps if prime
  - Must not be $p$ or $q$
  - Must be $< M$
- The Inverse Exponent ($E^{-1}$) for our private key
  - We use Euler’s Theorem and the Euclidean Algorithm to find this
We pick two Prime Numbers, p and q

- Size of primes used
  - Size of message to be encrypted
    - Modular arithmetic limits the size of the numeric message
  - Security level desired
    - Factoring a composite number into primes can be “hard”
    - These 2 numbers are kept secret
    - In real life these will be large (> 100 digits)
- For simplicity we will use p=421, q=523

- Modulus (M) = p(q) = 421(523) = 220183
- Our numeric message must be limited to a number < 220183
Now we choose our Public-Key exponent (E)

- Requirements
  - Helps if prime
  - Must be < M
  - Must not be equal to p or q
- We will choose \( E = 379 \) (a prime)
- We now have our Public-Key!
- We tell the world if they wish to send us an encrypted message, to raise the numeric representation of the message to the power of \( E \mod M \).
- Formula: \((\text{numeric msg})^E \mod M\)
Finding our Private-Key

- This must be the inverse function so we need to find $E^{-1}$.
  - Remember $2^7 = 128$
  - So $128^{-7} = 2$
- **Euler’s Theorem** tells us that to calculate the inverse power for a particular modulus made from two primes we use $\phi(M) = (p-1)(q-1)$.
  - For us $\phi(220183) = (421-1)(523-1) = 219240$
- Euler says we can find the multiplicative inverse for a value (our exponent 379) using $\phi(M)$ and the Extended Euclidean Algorithm
Euclidean Algorithm

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>219240</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>578</td>
<td>379</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>178</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>-17</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>49</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-66</td>
</tr>
</tbody>
</table>

- $219240(-66) + 379(38179) = 1 \mod 219240$ so $38179$ is the Multiplicative inverse of $379 \mod 219240$.
- So according to Euler’s theorem, $38179$ is the inverse power for $379 \mod 220183$
We have completed Private–Key

- We found \( E^{-1} = 38179 \)
- Applying this exponent to a message we receive that is encrypted using our public key will restore the original numeric message
- \( (\text{MsgPuKey(Scott)})^{E^{-1}} \mod 220183 \)
Summary of Key Values

- **Public Key:**
  - Exponent: 379 (E)
  - Modulus: 220183 (M)

- **Private Key:**
  - Exponent: 38179 (E–1)
  - Modulus: 220183 (M)

- **Message we will send:**
  - $\text{ITS}_{37}$ (character) = 25,743$_{10}$ (numeric)
Encryption of message

- Remember ITS?
  - ITS is represented numerically as $25,743_{10}$
- Using Public Key ($\text{msg}^{379} \mod 220183$)
  - Calculate $25,743^{379} \mod 220183$
  - Use PowerMod Program
  - Encrypted numeric message is: 65,208
  - We can send numeric message or convert to text and send
    - Convert to text by division using utility
    - 65,208 $\Rightarrow$ 1ANE
Decryption of message

- If we received “text” message
  - Convert to Numeric message
- Decrypt numeric message (65,208) using our Private-Key \(\text{msg}^{38179} \mod 220183\)
  - \(65208^{38179} \mod 220183\)
  - Use **PowerMod** Program
  - Numeric message = 25,743
  - **Convert** this to “text”
    - 25,743 => ITS
- We see the original message
Observations

- Once Keys are determined a program is used to do these calculations automatically.
- Computers use built in Numeric coding such as ASCII and base 2 instead of our example of using SSCII and base 10.
- Security is a function of the size of the prime numbers used and is based on the assumption that factorization of large primes is difficult.
How do I know the message came from the person who claims to have sent it?

Inverse functions work both ways

◦ Attach signature encrypted with your private key
◦ The recipient uses your public key to decrypt the signature.
◦ If it is from you, your signature will be readable.
Saw classic number theory applied to modern day problems
Difference between Symmetric vs. Asymmetric encryption
Public–Key/Private–Key resembles a trap–door
How RSA encryption works
Where is Public–Key encryption is used:
  ◦ common in browsers
  ◦ used in banking transactions
  ◦ in many software products like Intuit’s Quicken
  ◦ secure when large numbers are used to create the modulus because factoring of large numbers is “hard”
What’s next – Elliptic Curve Cryptography
Questions or Comments