Encryption

What is it and why do we need it?

Admin

Assignment 6

Midterm next Thursday
- Covers everything from 2/16 – 3/24 + some minor SML
- Will not have to write any assembly
- 2 pages of notes
- Review sessions next week (TBA)

Encryption

I like bananas

Encryption
Encryption

I like bananas

Encryption: a bad attempt

I like bananas

Encryption: the basic idea

I like bananas

Encryption: a better approach

I like bananas

The hawk sleeps at midnight
Encryption uses

Where have you seen encryption used?

Private key encryption

I like bananas

decrypt message

send encrypted message

I like bananas

Any problems with this?
Private key encryption

Public key encryption

Two keys, one you make publicly available and one you keep to yourself

Share your public key with everyone
Public key encryption

I like bananas

encrypt message

send encrypted message

decrypt message

Only the person with the private key can decrypt!

Modular arithmetic

Normal arithmetic:
\[ a = b \]
\[ a \text{ is equal to } b \text{ or } a-b = 0 \]

Modular arithmetic:
\[ a \equiv b \pmod{n} \]
\[ a-b = n \times k \text{ for some integer } k \text{ or} \]
\[ a = b + n \times k \text{ for some integer } k \text{ or} \]
\[ a \% n = b \% n \text{ (where } \% \text{ is the mod operator) \]

Which of these statements are true?

12 \equiv 5 \pmod{7} \]
52 \equiv 92 \pmod{10} \]
17 \equiv 12 \pmod{6} \]
65 \equiv 33 \pmod{32} \]
Modular arithmetic

Which of these statements are true?

12 \equiv 5 \pmod{7}

12 - 5 = 7 = 1 \times 7
12 \% 7 = 5 = 5 \% 7

52 \equiv 92 \pmod{10}

92 - 52 = 40 = 4 \times 10
92 \% 10 = 2 = 52 \% 20

17 \equiv 12 \pmod{6}

17 - 12 = 5
17 \% 6 = 5
12 \% 6 = 0

65 \equiv 33 \pmod{32}

65 - 33 = 32 = 1 \times 32
65 \% 32 = 1 = 33 \% 32

Modular arithmetic properties

\begin{align*}
\text{If:} & \quad a \equiv b \pmod{n} \\
\text{then:} & \quad a \mod n = b \mod n
\end{align*}

More importantly:

\begin{align*}
(a + b) \mod n &= ((a \mod n) + (b \mod n)) \mod n \\
(a \times b) \mod n &= ((a \mod n) \times (b \mod n)) \mod n
\end{align*}

What do these say?

Modular arithmetic

Why talk about modular arithmetic and congruence? How is it useful? Why might it be better than normal arithmetic?

We can limit the size of the numbers we’re dealing with to be at most n (if it gets larger than n at any point, we can always just take the result mod n)

The mod operator can be thought of as mapping a number in the range 0 … number-1
What does GCD stand for?

GCD

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

\[
gcd(25, 15) = 5
\]

Divisors:

<table>
<thead>
<tr>
<th></th>
<th>25</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

\[
gcd(100, 52) = ?
\]
### Greatest Common Divisor

gcd(a, b) is the largest positive integer that divides both numbers without a remainder

\[ \text{gcd}(100, 52) = 4 \]

<table>
<thead>
<tr>
<th>100</th>
<th>52</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>52</td>
</tr>
<tr>
<td>50</td>
<td>26</td>
</tr>
<tr>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Divisors:

- 100
- 50
- 25
- 20
- 10
- 5
- 4
- 2
- 1

### Greatest Common Divisor

- gcd(14, 63) = ?
- gcd(23, 5) = ?
- gcd(7, 56) = 7
- gcd(100, 9) = ?
- gcd(111, 17) = ?

### Greatest Common Divisor

- gcd(14, 63) = 7
- gcd(23, 5) = 1
- gcd(7, 56) = 7
- gcd(100, 9) = 1
- gcd(111, 17) = 1

Any observations?

### Greatest Common Divisor

- When the gcd = 1, the two numbers share no factors/divisors in common
- If gcd(a, b) = 1 then a is relatively prime to b
- This a weaker condition than primality, since any two prime numbers are also relatively prime, but not vice versa
Greatest Common Divisor

A useful property:

If two numbers, \(a\) and \(b\), are relatively prime (i.e. \(gcd(a,b) = 1\)), then there exists a \(c\) such that

\[a^c \mod b = 1\]

RSA public key encryption

Have you heard of it?

What does it stand for?

RSA public key encryption

RSA is one of the most popular public key encryption algorithms in use

RSA = Ron Rivest, Adi Shamir and Leonard Adleman

RSA public key encryption

1. Choose a bit-length \(k\)
   Security increase with the value of \(k\), though so does computation

2. Choose two primes \(p\) and \(q\) which can be represented with at most \(k\) bits

3. Let \(n = pq\) and \(\varphi(n) = (p-1)(q-1)\)
   \(\varphi()\) is called Euler's totient function

4. Find \(d\) such that \(0 < d < n\) and \(gcd(d,\varphi(n)) = 1\)

5. Find \(e\) such that \(de \mod \varphi(n) = 1\)
   Remember, we know one exists!
RSA public key encryption

\[ p: \text{prime number} \quad q(n) = (p-1)(q-1) \]
\[ q: \text{prime number} \quad d: \quad 0 < d < n \quad \text{and} \quad \gcd(d, q(n)) = 1 \]
\[ n = pq \quad e: \quad \text{de mod } q(n) = 1 \]

Given this setup, you can prove that given a number \( m \):

\[ (m^e)^d = m^{ed} = m \pmod n \]

What does this do for us, though?

RSA encryption/decryption

You have a number \( m \) that you want to send encrypted

\[ \text{Encrypt}(m) = m^e \pmod n \quad \text{(uses the public key)} \]

How does this encrypt the message?

RSA public key encryption

\[ p: \text{prime number} \quad q(n) = (p-1)(q-1) \]
\[ q: \text{prime number} \quad d: \quad 0 < d < n \quad \text{and} \quad \gcd(d, q(n)) = 1 \]
\[ n = pq \quad e: \quad \text{de mod } q(n) = 1 \]

private key

\( (d, n) \)

color: red

public key

\( (e, n) \)

color: green

RSA encryption/decryption

private key

\( (d, n) \)

color: red

public key

\( (e, n) \)

color: green

You have a number \( m \) that you want to send encrypted

\[ \text{Encrypt}(m) = m^e \pmod n \quad \text{(uses the public key)} \]

- Maps \( m \) onto some number in the range 0 to \( n-1 \)
- If you vary \( e \), it will map to a different number
- Therefore, unless you know \( d \), it's hard to know what the original \( m \) was after the transformation
RSA encryption/decryption

<table>
<thead>
<tr>
<th>Private key</th>
<th>Public key</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(d, n)$</td>
<td>$(e, n)$</td>
</tr>
</tbody>
</table>

You have a number $m$ that you want to send encrypted.

encrypt($m$) = $m^e \mod n$ (uses the public key)

decrypt($z$) = $z^d \mod n$ (uses the private key)

Does this work?

$decrypt(z) = decrypt(m^e \mod n)$

$= (m^e \mod n)^d \mod n$ (definition of decrypt)

$= (m^e)^d \mod n$ (modular arithmetic)

$= m \mod n$ [since $m^d \equiv m \mod n$]

z is some encrypted message

Definition of decrypt

Did we get the original message?

RSA encryption: an example

$p$: prime number

$q$: prime number

$n = pq$

$q(n) = (p-1)(q-1)$

d: $0 < d < n$ and gcd$(d, q(n)) = 1$

e: de mod $q(n) = 1$

$p = 3$

$q = 13$

$n = ?$

$q(n) = ?$

d = ?

e = ?
RSA encryption: an example

- p: prime number
- q: prime number
- n = pq

\[
p = 3 \\
q = 13 \\
n = 39
\]

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

- 0 < d < n and gcd\(d,\phi(n)\) = 1

\[
d \text{ s.t. } de \text{ mod } \phi(n) = 1
\]

\[
p = 3 \\
q = 13 \\
n = 3^*13 = 39
\]

\[
\phi(n) = \phi(39) = ?
\]

\[
\phi(n) = \phi(39) = 2*12 = 24
\]
### RSA encryption: an example

<table>
<thead>
<tr>
<th>p: prime number</th>
<th>q(n) = (p-1)(q-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q: prime number</td>
<td>d: 0 &lt; d &lt; n and gcd(d, φ(n)) = 1</td>
</tr>
<tr>
<td>n = pq</td>
<td>e: de mod φ(n) = 1</td>
</tr>
</tbody>
</table>

| p = 3          |
| q = 13         |
| n = 39         |
| φ(n) = 24      |
| d = ?          |
| e = ?          |

### RSA encryption: an example

<table>
<thead>
<tr>
<th>p: prime number</th>
<th>q(n) = (p-1)(q-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q: prime number</td>
<td>d: 0 &lt; d &lt; n and gcd(d, φ(n)) = 1</td>
</tr>
<tr>
<td>n = pq</td>
<td>e: de mod φ(n) = 1</td>
</tr>
</tbody>
</table>

| p = 3          |
| q = 13         |
| n = 39         |
| φ(n) = 24      |
| d = 5          |
| e = 29         |

### RSA encryption: an example

| n = 39         |
| d = 5          |
| e = 29         |

<table>
<thead>
<tr>
<th>encrypt(m) = m^e mod n</th>
</tr>
</thead>
</table>

| encrypt(10) = ?         |
### RSA encryption: an example

<table>
<thead>
<tr>
<th>n = 39</th>
<th>encrypt(m) = m^e mod n</th>
</tr>
</thead>
<tbody>
<tr>
<td>d = 5</td>
<td>decrypt(z) = z^d mod n</td>
</tr>
<tr>
<td>e = 29</td>
<td></td>
</tr>
</tbody>
</table>

encrypt(10) = 10^29 mod 39 = 4

decrypt(4) = 4^5 mod 39 = 10

crypt(2) = ?
RSA encryption: an example

\[ n = 39 \]
\[ d = 5 \]
\[ e = 5 \]

encrypt(2) = \( 2^e \mod n = 32 \mod 39 = 32 \)

decrypt(32) = ?

RSA encryption in practice

For RSA to work: \( 0 \leq m < n \)

What if our message isn’t a number?

What if our message is a number that’s larger than \( n \)?

RSA encryption in practice

For RSA to work: \( 0 \leq m < n \)

What if our message isn’t a number?

We can always convert the message into a number (remember everything is stored in binary already somewhere!)

What if our message is a number that’s larger than \( n \)?

Break it into \( n \) sized chunks and encrypt/decrypt those chunks
### RSA encryption in practice

**Encrypting “I like bananas”**

- **Encrypt:** `I like bananas` = 0101100101011100 ...
- **Encode as a binary string (i.e., number):**
  - 0101100101011100 ...
- **Break into multiple < n size numbers:**
  - 4, 15, 6, 2, 22, ...
- **Encrypt each number:**
  - 17, 1, 43, 15, 12, ...

**Decrypting the Encrypted Message**

- **Decrypt:**
  - `17, 1, 43, 15, 12, ...`
  - **Decrypt each number:**
    - 4, 15, 6, 2, 22, ...
  - **Put back together:**
    - 0101100101011100 ...
  - **Turn back into a string (or whatever the original message was):**
    - “I like bananas”

*Often encrypt and decrypt just assume sequences of bits and the interpretation is done outside.*