To do RSA we need fast Modular Exponentiation and Primality generation which we have shown.

We also need Modular Division.

To do Modular Division we use the extended Euclid Algorithm which we will now build towards.
Euclid’s Rule for GCD

How do you find the greatest common divisor of two integers

- Largest integer that divides both
- Could factor but that is exponential

Back in ancient Greece Euclid discovered the rule:

If $x$ and $y$ are positive integers with $x \geq y$ then

$$\text{gcd}(x,y) = \text{gcd}(x \mod y, y)$$

As soon as $x \mod y$ is 0, then $y$ is the gcd

Must swap parameters each time to keep largest as $1^{st}$ parameter
Euclid’s Algorithm

- This rule leads to the following algorithm

Function Euclid \((a,b)\)
Input: Two integers \(a\) and \(b\) with \(a \geq b \geq 0\) (\(n\)-bit integers)
Output: \(\text{gcd}(a,b)\)

if \(b=0\): return \(a\)
else: return \(\text{Euclid}(b, a \mod b)\)

- Examples
- Complexity?
Euclid’s Algorithm

- This rule leads to the following algorithm

  Function Euclid \((a,b)\)
  
  Input: Two integers \(a\) and \(b\) with \(a \geq b \geq 0\) (\(n\)-bit integers)
  
  Output: \(\text{gcd}(a,b)\)

  - if \(b=0\): return \(a\)
  - else: return Euclid\((b, a \mod b)\)

- Examples

- Complexity?
  - Each call reduces the arguments by at least 1/2
  - Thus \(O(n = \log_2(a))\) calls each with an \(n^2\) division
  - Complexity is \(O(n^3)\)
function extended-Euclid \((a, b)\)
Input: Two positive integers \(a\) and \(b\) with \(a \geq b \geq 0\) (\(n\)-bits)
Output: Integers \(x, y, d\) such that \(d = \gcd(a, b)\)
and \(ax + by = d\)

if \(b = 0\): return \((1, 0, a)\)
\((x', y', d) = \text{extended-Euclid}(b, a \mod b)\)
return \((y', x' - \text{floor}(a/b)y', d)\)

- Exact same as Euclid except during stack unraveling
- This gives us the results we need for modular division
- We'll do an example in a minute
Modular Division - Multiplicative Inverses

- Every real number $a \neq 0$ has an inverse $1/a$, $(a \cdot 1/a = 1)$
- Dividing by $a$ is the same as multiplying by the inverse $1/a$
- In modular arithmetic we say that $x$ is the multiplicative inverse of $a$ modulo $N$ if $ax = 1 \pmod{N}$
- Multiplicative inverse of 3 mod 5 = ?
Every real number \( a \neq 0 \) has an inverse \( 1/a \), \((a \cdot 1/a = 1)\)

Dividing by \( a \) is the same as multiplying by the inverse \( 1/a \)

In modular arithmetic we say that \( x \) is the multiplicative inverse of \( a \) modulo \( N \) if \( ax = 1 \) (mod \( N \))

Multiplicative inverse of 3 mod 5 = 2
- We will also call 2, \( a^{-1} \) mod \( N \) in this case
- If a multiplicative inverse exists, it is unique mod \( N \)

Unlike regular arithmetic most numbers do not have a multiplicative inverse in modular arithmetic

What is the multiplicative inverse of 2 mod 4?

An Algorithm to find the inverse?
Modular Division - Multiplicative Inverses

- In fact, the only time $a$ has a multiplicative inverse mod $N$ is when $a$ and $N$ are relatively prime.
- Two numbers $a$ and $b$ are relatively prime if $\gcd(a, b) = 1$.
- If $a$ and $N$ are relatively prime then we know the multiplicative inverse exists (e.g. $4 \mod 7$).
- We will use extended-Euclid($a, N$) to find the multiplicative inverse $x$ of $a \mod N$.
  - Must return 1 for the $\gcd$ to confirm that $a$ and $N$ are relatively prime.
- When $a$ and $N$ are relatively prime, the extended-Euclid algorithm returns $x$ and $y$ such that $ax + Ny = 1$.
- $Ny = 0 \mod N$ for all integers $y$.
- Thus, $ax \equiv 1 \mod N$.
- Then $x$ is the multiplicative inverse of $a \mod N$.
- Modular $N$ division can only be done for numbers relatively prime to $N$ and the division is actually carried out by multiplying by the inverse.
What is multiplicative inverse of 20 Mod 79

- Are they relatively prime?
- Euclid or extended-Euclid are the algorithms we use to find out (with the extension not needed). The extension only kicks in after the gcd has been found anyway.
- returns integers $x, y, d$ such that $d = \gcd(a, b)$ and $ax + by = d$
- Remember to put the largest number first and if you have to switch at the beginning, then remember to switch $x$ and $y$ at the end
function extended-Euclid(a, b)
if b = 0: return (1, 0, a)
(x', y', d) = extended-Euclid(b, a mod b)
return (y', x' – floor(a/b)y', d)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x'</th>
<th>y'</th>
<th>d</th>
<th>ret 1</th>
<th>ret 2</th>
<th>ret 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
function extended-Euclid \((a, b)\)

if \(b = 0\): return \((1, 0, a)\)

\((x', y', d) = extended-Euclid(b, a \mod b)\)

return \((y', x' - \text{floor}(a/b)y', d)\)

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(x')</th>
<th>(y')</th>
<th>(d)</th>
<th>ret 1</th>
<th>ret 2</th>
<th>ret 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>20</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\(ax + Ny = 1 = 79(-1) + 20(4)\)

\(ax + Ny = 1 = 20(4) + 79(-1)\)

Thus \(x = a^{-1} = 4\)

Since we switched initially switched 20 and 79

Complexity?
Multiplicative Inverse of 12 mod 15?

function extended-Euclid (a, b)
    if b = 0: return (1, 0, a)
    (x', y', d) = extended-Euclid(b, a mod b)
    return (y', x' – floor(a/b)y', d)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x'</th>
<th>y'</th>
<th>d</th>
<th>ret 1</th>
<th>ret 2</th>
<th>ret 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
function extended-Euclid(a, b)
if b = 0: return (1, 0, a)
(x', y', d) = extended-Euclid(b, a mod b)
return (y', x' – floor(a/b)y', d)

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>x'</th>
<th>y'</th>
<th>d</th>
<th>ret 1</th>
<th>ret 2</th>
<th>ret 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ ax + Ny = 3 = 15(1) + 12(-1) \]
However, there is no multiplicative inverse since the \( \text{gcd} = 3 \) and thus \( a \) and \( b \) are not relatively prime
Now we have all the algorithms/tools needed to do RSA

RSA = Rivest, Shamir, and Adleman

Common Public Key Encryption Approach

Assume $x$ is the initial message to be sent and $e(x)$ encrypts $x$ into $y$ while $d(y)$ decrypts $y$ back to $x$

Private key approaches - Alice and Bob both know $e$ and $d$ and can thus communicate with each other – but new/unknown people can't

Public key - $d$ is the inverse of $e$. Bob creates $e$ and $d$ and publishes $e$ to everyone, but only he knows $d$. Alice can create her own pair and publish her own $e$, etc.
RSA Encryption

- Messages are numbers modulo $N$
- Messages larger than $N$ are segmented
- Encryption is a bijection (one-to-one and onto) from $\{0, 1, \ldots, N-1\}$ to $\{0, 1, \ldots, N-1\}$
  - a permutation
- Decryption is its inverse
Pick any two primes $p$ and $q$ and let $N = p \cdot q$

Choose a number $e$ relatively prime to $(p-1)(q-1)$

- $e$ is often chosen as 3 – permits fast encoding

Then the mapping $x^e \mod N$ is a bijection onto \{0, 1, ..., $N-1$\} - Publish $(e, N)$ as the public key for encryption

Find $d$, the multiplicative inverse of $e \mod (p-1)(q-1)$ using extended-Euclid($(p-1)(q-1), e$)

Then for all $x \in \{0, 1, ..., N-1\}$ $(x^e)^d = x \mod N$

- Proof on p. 34 based on Fermat's if $p$ is prime, then $a^{p-1} \equiv 1 \mod p$

Keep $d$ private for decryption – Why can't they figure out $d$?
Let $p = 5$; $q = 11$
Then $N = p \cdot q = 55$
Let $e = 3$
  - Note that $\gcd((p-1)(q-1), e) = \gcd(40, 3) = \text{Euclid}(40, 3) = 1$
Thus, public key = $(N, e) = (55, 3)$
Private key: $d = 3^{-1} \text{mod} 40 = 27$
  - found with extended-Euclid($((p-1)(q-1), e) = \text{extended-Euclid}(40, 3)$ which gives $d = 27$

Encryption of $x$: $y = x^3 \text{mod} 55$
  - Encryption and decryption use modular exponentiation algorithm
Decryption of $y$: $x = y^{27} \text{mod} 55$

Let $x = 13$
$y = 13^3 = 52 \text{ (mod} 55)$
$x = 52^{27} = 13 \text{ (mod} 55)$
How could we break RSA - Could try factoring $N$ into primes $p$ and $q$ - no known polynomial algorithm

The crux of the security behind RSA

- Efficient algorithms / Polynomial time computability for:
  - Modular Exponentiation – modexp()
  - GCD and modular division – extended-Euclid()
  - Primality Testing and creation – primality2()
- Absence of sub-exponential algorithms for Factoring

The gulf between polynomial and exponential saves the day in this case