### Lecture 21: More RSA / Start inductively defined sets

<table>
<thead>
<tr>
<th>Sender</th>
<th>Network (attacker)</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows $msg$</td>
<td></td>
<td>Knows $m$ and $k$</td>
</tr>
<tr>
<td>Compute $c := [msg]^k_m$</td>
<td>$c$</td>
<td>Compute “$msg = \sqrt[2]{c}$”</td>
</tr>
</tbody>
</table>
Lecture 21: More RSA / Start inductively defined sets

<table>
<thead>
<tr>
<th>Sender</th>
<th>Network (attacker)</th>
<th>Recipient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows $msg$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute $c := [msg]^k_m$</td>
<td>$(m, k)$</td>
<td>Knows $m$ and $k$</td>
</tr>
<tr>
<td>$c$</td>
<td></td>
<td>Compute &quot;$msg = \sqrt[k]{c}$&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c^{[k]}^{-1} = [msg][k][k]^{-1} = [msg][1] = [msg]$</td>
</tr>
</tbody>
</table>
Lecture 21: More RSA / Start inductively defined sets

Sender  Network (attacker)  Recipient

Knows \( \text{msg} \)

\( \left( m, k \right) \)

Choose a unit \( [k]_{\varphi(m)} \)

Knows \( m \) and \( k \)

Find \( [k]^{-1}_{\varphi(m)} \)

Compute \( \text{c} := [\text{msg}]_{m}^{k} \)

Compute \( \text{msg} = \sqrt[k]{c} \)

\( \text{c}^{-1} \times [\text{msg}]_{[k][k]^{-1}} = [\text{msg}]_{[1]} = [\text{msg}] \)
Lecture 21: More RSA / Start inductively defined sets

Sender

```
Compute \( c := [msg]^k_m \)  
\([msg]_m \) must be a unit!
```

Recipient

```
Choose large primes \( p, q \)  
Let \( m = p \cdot q \)  
Compute \( \varphi(m) = (p - 1)(q - 1) \)  
Choose a unit \([k]_{\varphi(m)}\)  
Knows \( m \) and \( k \)  
Find \([k]^{-1}_{\varphi(m)}\)  
Compute \( "msg = \sqrt{c}" \)  
\( c[k]^{-1} = [msg]^{[k]^{-1}[1]} = [msg]^{[1]} = [msg] \)
```

Note: it's hard to find a message that isn't a unit.  
Non-unit share a factor with \( m = p \cdot q \).  
So \( \gcd(m, msg) \) is either \( p \) or \( q \).
Attacker (probably) can’t find \([k]^{-1}\_{\varphi(m)}\) (efficiently)

Proof by “contradiction”:

- If the attacker could efficiently find \([k]^{-1}\_{\varphi(m)}\), then they could use it to factor \(m\)
- We believe that the best (non-quantum) factoring algorithm takes exponential time in the number of digits of \(m\)
- There’s no proof that factoring is hard, but people have been trying for a long time to no avail

Note: one of the reasons quantum computing is a “big deal” is that there are efficient quantum factoring algorithms
Choosing large primes

- Fact: there is an efficient algorithm for (probabilistically) checking if a number is prime
- How to choose a large prime number: randomly choose a large odd number $n$.
  - If $n$ is prime, you’re done.
  - Otherwise, try again
- Fact: there are enough prime numbers of a given size to make this algorithm efficient
Choosing \([k]^{-1}\) and finding \([k]^{-1}_{\varphi(m)}\)

Checking if \([k]^{-1}_{\varphi(m)}\) exists, and finding it

- We showed that \([k]^{-1}_{\varphi(m)}\) exists if and only if \(gcd(k, \varphi(m)) = 1\).
- We showed that you can find \([k]^{-1}_{\varphi(m)}\) from the Bézout coefficients of \(k\) and \(\varphi(m)\).
- Fact: computing \(gcd(k, \varphi(m))\) and Bézout coefficients is efficient

How to find a unit in the first place?

- Most prime numbers will be units!
- Choose random prime \(k\), check whether it’s a unit, try again if not
Fast exponentiation

The sender and recipient need to compute \([a]^b\) for large \(b\).

**Example:**

\[ 5^{100} \]

\[ \underbrace{5 \cdot 5 \cdot 5 \cdot 5 \cdot \cdots \cdot 5}_\text{100 times} \]

**Trick:** Write exponent as a sum of powers of 2.

\[ 100 = 64 + 32 + 4 + 2 + 0.1 = (1000100)_2 \]

\[ 5 = 5^1 \]

\[ 5 \cdot 5 = 5^2 \]

\[ 5^4 = (5^2)^2 \]

\[ 5^6 = (5^3)^2 \]

\[ 5^{10} = (5^5)^2 \]

\[ \vdots \]

\[ (5^2)^2 = [4]^2 = 16 \]

\[ (5^4)^2 = [4]^4 = 256 \]

\[ (5^8)^2 = [4]^8 = 65536 \]

\[ (5^{16})^2 = [4]^{16} = 4^{16} \]

\[ \vdots \]
Making sure the message works

The recipient gets $a$ with $[a] = [msg]$, not $msg$:

- Sender ensures that $0 \leq msg < m$ (sending multiple messages if necessary)
- Recipient computes $\text{rem}(a, m) = msg$ by uniqueness of remainder

$[msg]$ needs to be a unit for Euler’s theorem to apply

- Almost everything is a unit!
Public key signatures

Public key cryptography can be used to ensure integrity using digital signatures:

Signer
Knows $msg$, $[k]_m^{-1}$
Computes $sig := [msg][k]^{-1}$

Network
(\(msg, sig\))

Verifier
Knows/trusts $(m, k)$
Check that $[msg] = sig^k = [msg][k]^{-1}[k]$. 
Next topic: Inductively defined sets

Inductive definitions are a more precise way to define certain sets:

- Natural numbers
- Strings
- Trees
- Arithmetic expressions: $(3+5) \cdot (4-2)$
- Programs
- Logical formulas