RSA Example

A small (and insecure) example [Stinson]:

- Bob:
  - chooses $p = 101$, $q = 113$
  - computes $n = pq = 11413$ and $\phi(n) = (p-1)(q-1) = 11200$
  - chooses $e = 3533$ (note: $\gcd(e, \phi(n)) = 1$)
  - computes $d = e^{-1} \mod \phi(n) = 6597$
  - publishes $n$ and $e$ (keeps $p$, $q$, and $d$ private)

- Alice wants to send 9726 to Bob:
  - computes $9726^e \mod n = 9726^{3533} \mod 11413 = 5761$
  - sends 5761 to Bob

- Bob:
  - computes $5761^d \mod n = 5761^{6597} \mod 11413 = 9726$
RSA Correctly Decrypts

Need to show that for any plaintext \( m \in \mathbb{Z}_n \), \((m^e)^d \mod n = m\).

We will need some number theory:

If \( \gcd(m,n)=1 \):
- we will use Euler’s Theorem (pg 81):
  
  If \( \gcd(a,n)=1 \), then \( a^{\phi(n)} \equiv 1 \pmod{n} \)
  
  if \( n \) is a prime, then \( \forall a \in \mathbb{Z}_n \): \( a^{n-1} \equiv 1 \pmod{n} \)
- can we conclude that \((m^e)^d \mod n = m\) ?

\[
(m^e)^d \equiv m^{1+k\phi(n)} \equiv m \cdot (m^{\phi(n)})^k \equiv m \cdot 1^k \equiv m \pmod{n}
\]

If \( \gcd(m,n)>1 \):
- we will use the Chinese Remainder Theorem (Section 3.4)
Suppose \( m_1, \ldots, m_k \) are pairwise relatively prime positive integers, and suppose \( a_1, \ldots, a_k \) are integers. Then the system of congruences

\[
    x \equiv a_i \pmod{m_i} \quad (1 \leq i \leq k)
\]

has a unique solution modulo \( M = m_1m_2\ldots m_k \), which is given by

\[
    x = \sum_{i=1}^{k} a_i M_i y_i \pmod{M},
\]

where \( M_i = M/m_i \) and \( y_i = M_i^{-1} \pmod{m_i} \), for \( 1 \leq i \leq k \).
Chinese Remainder Theorem

Example:

We want to find $x$ that satisfies the following congruencies:

\[
\begin{align*}
    x &\equiv 5 \pmod{7} \\
    x &\equiv 3 \pmod{11} \\
    x &\equiv 10 \pmod{13}
\end{align*}
\]

The solutions is unique mod ____:

\[x = \]
RSA Correctly Decrypts, part 2

If $m$ is not divisible by $p$:

Let $x = (m^e)^d$ want to show $x \equiv m \pmod{p}$

$x = (m^e)^d = m^{1+k \cdot \phi(n)} = m^{1+k(p-1)(q-1)} = m \cdot (m^{p-1})^k \cdot (q-1)$

we know:

$x = m \cdot (m^{p-1})^k \cdot (q-1) \equiv m\cdot (m^{\phi(p)})^k \cdot (q-1) \equiv m \cdot 1^k \cdot (q-1) \equiv m \pmod{p}$

If $m$ is divisible by $p$:

$x = (m^e)^d \equiv 0 \equiv m \pmod{p}$

$p | m$

we want to prove:

$(m^e)^d \equiv m \pmod{n}$

we need to show this in the case when $\gcd(m,n) > 1$

we will first show:

$(m^e)^d \equiv m \pmod{p}$

by Euler

\[\checkmark\]
RSA Correctly Decrypts, part 2

So, for all \( m \): \( (m^e)^d \equiv m \pmod{p} \).

\[ (m^e)^d \equiv m \pmod{q} \]

What can we conclude?

We want to show \( (m^e)^d \equiv m \pmod{n} \)

by CRT:

\[ \begin{align*}
    x &\equiv m \pmod{p} \\
    x &\equiv m \pmod{q}
\end{align*} \]

there is a unique \( x \) s.t. \( x \equiv m \pmod{p \cdot q} \)

We want:

to show \( (m^e)^d \pmod{n} \) is that solution

by definition

we know:

\[ \begin{align*}
    x-m &\equiv 0 \pmod{p} \\
    x-m &\equiv 0 \pmod{q}
\end{align*} \]

\[ \begin{align*}
    p \mid (x-m) \\
    q \mid (x-m)
\end{align*} \]

\[ x \equiv m \pmod{p \cdot q} \quad \Rightarrow \quad p \cdot q \mid (x-m) \]
We showed that RSA works (i.e., correctly decrypts). But...
Can we actually compute $m^e \mod n$ for very large numbers?

idea 1: keep multiplying $m$ by itself, at the end take $\mod n$
problem: memory!

idea 2: keep multiplying $m$ by itself, after every product, take $\mod n$
--> running time: #iterations is $e$, if $e$ has e.g. 1000 bits, $e$ could be up to $2^{1000} - 1$
problem: running time
(memory is ok)

idea 3: suppose that e.g., $e = 4$

$m^4 = (m^2)^2$

$m^{2^{10}} = (((m^2)^2)^2)^2 \ldots)^2$

Repeated squaring.
We showed that RSA works (i.e., correctly decrypts). But...

Can we actually compute \( m^e \mod n \) for very large numbers?

- **Note:** An efficient implementation exists that does not need to store the intermediate squares.
RSA: Computational Issues

Want: algorithms polynomial in the length of $n$.
What is the length of $n$?

$\log n$

Note: typically when estimating the running time of an algorithm, arithmetic operations are counted as taking constant time - but this is NOT feasible in cryptography where the numbers can be very large (recommended size of $p,q$ in RSA: 1024 bits)
RSA: Computational Issues

Modular Arithmetic:

Give upper bounds (using the big-Oh notation) on the running time of the following operations in \( \mathbb{Z}_n \):

- **addition**: \( O(\log n) \)
- **difference**: \( O(\log n) \)
- **multiplication**: \( O(\log^2 n) \) \( \leftarrow \) can be done faster, ask Dan
- **exponentiation**: \( O(\log^2 n \cdot \log e) \)
- **gcd\((a,b)\)**: \( O(\log n) \) iterations, each iter: addition, division: \( O(\log^2 a) \)
- **inverse**: Same as gcd
- **RSA encryption**: \( \equiv \) same as exponentiation
- **RSA decryption**: \( \equiv \) inverse, multiplication + find \( m \) \( \rightarrow \) \( n \)