The RSA Algorithm

The RSA Algorithm (Rivest-Shamir-Adleman algorithm) is the most important public-key cryptosystem.

The RSA works because:

1. If \( n = pq \), where \( p \) and \( q \) are large primes (several hundred digits), then
   - i) Given \( p \) and \( q \), we can easily multiply them to obtain \( n \), but
   - ii) Given \( n \), there is no known way to factor \( n \) as \( pq \) in any reasonable amount of time.

We also need these lemmas.

**Lemma 1.** If \( n = p_1 p_2 \ldots p_h \) is a product of distinct primes, then

i) \( \varphi(n) = (p_1-1)(p_2-1)\ldots(p_h-1) \), and

ii) \( p_i-1 \) divides \( \varphi(n) \) for all \( i \).

**Proof:** We know in general that

\[
\varphi(n) = n \prod_{i=1}^{h} \left(1 - \frac{1}{p_i} \right) = p_1 p_2 \ldots p_h \prod_{i=1}^{h} \left(1 - \frac{1}{p_i} \right) = \prod_{i=1}^{h} \left(p_i - 1\right) = (p_1-1)(p_2-1)\ldots(p_h-1).
\]

This proves (i), and (ii) follows immediately.

**Lemma 2:** If \( n = p_1 p_2 \ldots p_h \) is a product of distinct primes, then

\[
k \equiv 1 \pmod{\varphi(n)} \Rightarrow a^k \equiv a \pmod{n}
\]

for any \( a \).

**Proof:** It suffices to show that, for any \( a \),

\[
a^k \equiv a \pmod{p_i} \quad \text{for } i = 1, 2, \ldots, h.
\]

(If this holds, \( p_i \) divides \( a^k - a \) for all \( i \), so \( n \) must divide \( a^k - a \), showing that \( a^k \equiv a \pmod{n} \).

Consider each prime \( p_i \) separately.

i) \( a \equiv 0 \pmod{p_i} \), then \( a^k \equiv 0 \equiv a \pmod{p_i} \).

ii) Otherwise Fermat’s Little Theorem tells us that \( a^{p_i-1} \equiv 1 \pmod{p_i} \). Since \( p_i-1 \) divides \( \varphi(n) \), \( a^{\varphi(n)} \equiv 1 \pmod{p_i} \). So if \( k \equiv 1 \pmod{\varphi(n)} \),

\[
k = \varphi(n) t + 1 \quad \text{for some integer } t,
\]

\[
a^k \equiv a^{\varphi(n) t + 1} \equiv (a^{\varphi(n)})^t a \equiv a \pmod{p_i}.
\]

**Note:** None of these results hold if the square of some prime divides \( n \).

For example, if \( n = 12 = 2^2 3 \), then

\[
\varphi(12) = 4 \neq (2^2 - 1)(3 - 1).
\]

\[
5 \equiv 1 \pmod{\varphi(12)}, \text{ but } 2^5 = 32 \neq 2^1 = 2 \pmod{12}.
\]
The RSA works like this:

i) Alice chooses two large primes \( p_A \) and \( q_A \).

ii) Alice computes \( n_A = p_A q_A \) and \( \phi(n_A) = (p_A-1)(q_A-1) \).

iii) Alice chooses an integer \( e_A \) with \( \gcd(e_A, \phi(n_A)) = 1 \), possibly at random.

iv) Alice computes \( d_A \equiv e_A^{-1} \pmod{\phi(n_A)} \).

v) Alice’s public key is \((n_A, e_A)\). She distributes this. Her private key is \(d_A\). She keeps this secret.

vi) If \( 2^k \leq n_A < 2^{k+1} \), Alice’s encryption function for short messages (\( k \) bits or less, so \( M < n_A \)) is:

\[
E_A(M) = M^{e_A} \pmod{n_A}.
\]

Anyone can compute \( E_A(M) \). A longer message is encrypted by splitting it into \( k \)-bit blocks, and encrypting each block separately. Note that each encrypted block has \( k+1 \) bits.

vii) Alice’s decryption function for short messages is:

\[
D_A(M) = M^{d_A} \pmod{n_A}, \text{ provided } 0 \leq M < n_A.
\]

No one except Alice (or someone else who has discovered Alice’s private key) can compute this.

\textit{Note:} \( D_A(E_A(M)) \equiv (M^{e_A})^{d_A} \equiv M^{e_A d_A} \equiv M \pmod{n_A} \) since \( e_A d_A \equiv 1 \pmod{\phi(n_A)} \)

Once Alice has done this, she can

1) receive encrypted messages from Bob (or anyone else), and

2) send digitally-signed messages to Bob (or anyone else).

In order for Alice to send encrypted messages to Bob, or to receive digitally-signed messages from Bob, Bob will need to choose his own public and private keys, \((n_B, e_B)\) and \(d_B\).

Bob sends a short message \( M \) (at most \( k \) bits) to Alice like this:

i) Bob encrypts \( M \) as \( M^{e_A} \pmod{n_A} \), and sends \( M^{e_A} \) to Alice. (Note Bob knows \( e_A \) and \( n_A \)).

ii) Alice decrypts \( M^{e_A} \) as \( (M^{e_A})^{d_A} \equiv M \pmod{n_A} \). Thus Alice recovers \( M \).

(Noe Alice actually recovers the value of \( M \pmod{n_A} \), but this equals \( M \) as long as \( M < n_A \)).

For longer messages, Bob could break the message up into \( k \)-bit blocks, and encrypt each block separately. Alice would break the encrypted message in \( k+1 \) bit blocks, and decrypt each block separately.
Example:

i) Alice chooses: \( p_A = 59, \ q_A = 71 \).

ii) Alice computes: \( n_A = p_A \cdot q_A = 4189 \),
\( \phi(n_A) = (59-1) \cdot (71-1) = 4060 \).

iii) Alice chooses: \( e_A = 671 \).

iv) Alice computes: \( d_A \equiv e_A^{-1} \pmod{4060} \equiv 1791 \).
   She may do this using Euclid’s extended algorithm, which uses only \( O(\log(n_A)) \) steps, so is feasible even if \( n_A \) has hundreds of digits.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( q[i] )</th>
<th>( r[i] )</th>
<th>( x[i] )</th>
<th>( y[i] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>4060</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>671</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6 34</td>
<td>1</td>
<td>-6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>19 25</td>
<td>-19</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 9 20</td>
<td>-59</td>
<td>357</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2 7</td>
<td>-59</td>
<td>357</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1 2 79</td>
<td>-478</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3 1 -296</td>
<td>1791</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

v) Alice distributes her public key \((4189, 671)\) and keeps her private key \(1791\) secret.

vi) Alice’s encryption function is: \( E_A(M) \equiv M^{671} \pmod{4189} \), provided \( 0 \leq M < 2^{12}-1 = 4095 \).

Alice’s decryption function is: \( D_A(M) \equiv M^{1791} \pmod{4189} \), provided \( 0 \leq M < 4095 \).

Both functions can be computed using at most \( 2\log_2(n_A) \) modular multiplications, using fast exponentiation.

Bob sends Alice the message "RSA" as follows:
\[
\text{RSA} = \begin{array}{llll}
01010010 & 01010011 & 01000001 \\
\end{array}
\]
in ASCII.

Bob breaks this up into two 12-bit integers:
\[
01010010 & 01010011 & 01000001, \text{ or } 1317, 833
\]
He computes \( 1317^{671} \equiv 3530, 833^{671} \equiv 3050 \pmod{4189} \).

The ciphertext is 3530, 3050, or
\[
\begin{array}{llllllllll}
011011010101 & 010111110101 & 0101000001 \\
\end{array}
\]
(Note that 13-bit blocks were used, as \( M^{671} \pmod{4189} \) could be greater than 4095.)

Alice decrypts the message by computing
\[
3530^{1791} \equiv 1317, 3050^{1791} \equiv 833 \pmod{4189}
\]
giving plaintext 1317, 833, or \[
\begin{array}{llllllllll}
01010010 & 01010011 & 01000001 \text{, or } "RSA".
\end{array}
\]