Message Passing, RSA Encryption

Robert Y. Lewis

CS 0220 2022

March 11, 2022
Overview

1. Exam details

2. The RSA Algorithm (8.11)
   - Sending Secrets
Midterm exam

- Beware daylight savings!
- You will get an email with a link to your exam. *Use the form provided in this link.*
- Don’t wait until 11:59pm to submit!

- Collaboration *is not allowed.* Do not discuss the exam in any way before Tuesday. Do not share your exam form with anyone, at any point.

- EdStem posts on Monday will be private. Ask clarifying questions, but we will not help with answers to the problems.
Sending messages

Motivations:

- Why do we send each other messages? Communication is a pretty human activity. Coordination is a practical application.


- Why might the message need to be encrypted? Message can be intercepted, stolen/broadcast by a third party, accidentally revealed like by being left on screen connected to projector. Communication channel is open.
Encoding messages

We’ll assume all messages are fixed-length bit strings.

Is that sufficiently general? Can we encode arbitrary messages this way?
The Alphabet

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00000 space</td>
<td>01000 H</td>
</tr>
<tr>
<td>00001 A</td>
<td>01001 I</td>
</tr>
<tr>
<td>00010 B</td>
<td>01010 J</td>
</tr>
<tr>
<td>00011 C</td>
<td>01011 K</td>
</tr>
<tr>
<td>00100 D</td>
<td>01100 L</td>
</tr>
<tr>
<td>00101 E</td>
<td>01101 M</td>
</tr>
<tr>
<td>00110 F</td>
<td>01110 N</td>
</tr>
<tr>
<td>00111 G</td>
<td>01111 O</td>
</tr>
<tr>
<td>10000 P</td>
<td>10001 Q</td>
</tr>
<tr>
<td>10010 R</td>
<td>10011 S</td>
</tr>
<tr>
<td>10100 T</td>
<td>10101 U</td>
</tr>
<tr>
<td>10110 V</td>
<td>10111 W</td>
</tr>
<tr>
<td>11000 X</td>
<td>11001 Y</td>
</tr>
<tr>
<td>11010 Z</td>
<td>11011 .</td>
</tr>
<tr>
<td>11100 !</td>
<td>11101 ?</td>
</tr>
<tr>
<td>11110 ,</td>
<td>11111 @</td>
</tr>
</tbody>
</table>
Alice and Bob share 60 random bits (the “one-time pad”) in advance:

10110 00011 00000 00110 10010 00010
00000 11000 00110 01001 11111 11110

Alice: Wants to send a private message to Bob. She turns it into a sequence of 60 bits. She then computes the bitwise “xor” of her message and the one-time pad and transmits it:

10101 01100 00110 00000 10111 00111
00000 11000 00110 01001 11111 11110
One-time pad: Decryption

Bob: Wants to read Alice’s message.

10101 01100 00110 00000 10111 00111
00000 11000 00110 01001 11111 11110

How can he recover it? Bitwise “xor” with the one-time pad will undo the encryption operation.

```
| encrypted line 1: | 10101 01100 00110 00000 10111 00111 |
| pad line 1:       | 10110 00011 00000 00110 10010 00010 |
| xor line 1:       | 00011 01111 00110 00110 00101 00101 |
| text line 1:      | C O F F E E |
| encrypted line 2: | 00000 11000 00110 01001 11111 11110 |
| pad line 2:       | 00000 11000 00110 01001 11111 11110 |
| xor line 2:       | 00000 00000 00000 00000 00000 00000 |
| text line 2:      | _ _ _ _ _ _ |
```
One-time pad: Cracking

Eve: Sees the encrypted message and wants to understand it. She doesn’t have the one-time pad.

The encrypted message gives no information about the unencrypted message. All possible messages are equally likely.

Although, if one-time pad is reused, information is leaked.
Public key cryptography

The one-time pad is essential to the one-time pad scheme. How can Alice and Bob agree on the one-time pad if Eve is listening? Could send it encrypted with a one-time pad, but they’d use up $n$ bits of pad to transmit $n$ bits of pad, so that doesn’t help.

Can they put their key out in the open??

Bob wants to be able to receive secret messages. Bob creates a private key, which must remain secret. Bob also creates a public key, which is made public. Anyone, Alice say, who wants to send a secret message to Bob can encrypt it with Bob’s public key. Only Bob can decrypt such messages (using his private key).

No other communication or agreements or secret emails are needed.
Beforehand

1. Bob generates two distinct (hundreds of digits long) primes, $p$ and $q$ and keeps them hidden.
2. Bob sets $n := pq$.
3. Bob selects an integer $e \in [1, n)$ such that $\gcd(e, (p - 1)(q - 1)) = 1$. The public key is the pair $(e, n)$.
4. Compute $d \in [1, n)$ such that $de \equiv 1 \pmod{(p - 1)(q - 1)}$. The private key is the pair $(d, n)$. 
Encryption: Alice wants to send unencrypted message $m$. She computes and then sends the encrypted message $m^* = \text{rem}(m^e, n)$. (Uses $e$ and $n$, which are public.)

Decryption: Bob receives $m^*$. He decrypts by computing $m' = \text{rem}((m^*)^d, n)$. (Uses $d$ and $n$, where $d$ is private.)
Decrypting works

**Claim:** \( m' = m \).

**Partial Proof** (for case when \( m \) is relatively prime to \( n \)):

\[
\begin{align*}
  m' & = \text{rem}((m^*)^d, n) \quad \text{defn } m' \\
  & = \text{rem}((\text{rem}(m^e, n))^d, n) \quad \text{def of } m^* \\
  & = \text{rem}((m^e)^d, n) \quad \text{“pre-mod” property} \\
  & = \text{rem}(m^{ed}, n) \quad \text{exponentation} \\
  & = \text{rem}(m^{1+k(p-1)(q-1)}, n) \quad \text{selection of } e \\
  & = \text{rem}(m^{1+k\phi(n)}, n) \quad \text{property of } \phi(pq) \\
  & = \text{rem}(m \times (m^{\phi(n)})^k, n) \quad \text{rearrange factors} \\
  & = \text{rem}(m, n) \quad \text{Euler’s thm} \\
  & = m \quad \text{assuming } m < n
\end{align*}
\]

Can handle gcd\((m, n) > 1\), too, but this argument doesn’t show why.
How to compute?

1. Generate big primes $p$ and $q$? Factoring is hard, but can do (randomized) primality test quickly—Miller-Rabin. For $d$-digit prime, need about $d$ tries to find one.

2. Multiply big primes $pq$ to get $n$? Standard multiplication algorithm scales well to big numbers.

3. Find number relatively prime to $(p - 1)(q - 1)$? Generate and test works again, but the chance of getting a hit at least as good and the test is much faster. Specifically, Euclid!

4. Compute $d = e^{-1} \mod (p - 1)(q - 1)$? Pulvarize! (aka gcdcombo.)

5. Compute $m^* = m^e \mod n$ or $m' = (m^*)^d \mod n$? Exponentiation by repeated squaring.

All of the steps are fast enough.
Breaches

Private: $p, q, d.$

Public: $n, e.$

- If $p$ revealed? Can get $q = n/p$. Can get $d$ as $e^{-1} \mod (p - 1)(q - 1)$.
- If $q$ is revealed? Same as $p$.
- If $d$ is revealed? Can get us $p$ and $q$, though less clear how. But, can decrypt messages, so that’s pretty bad.

We’re pretty sure it’s hard to get $p$ and $q$ from $n$. Factoring. We don’t know if knowing $e$ makes factoring easier, but no one has found a way to do it yet.
Other RSA uses

Encrypt with Bob’s public key. Can only be decrypted with Bob’s private key. Sending private messages to Bob.

Encrypt with Bob’s private key. Can be decrypted with Bob’s public key. Digital signature—only Bob can make such a message.

Encrypt with Bob’s public key, then Alice’s. Bob and Alice must work together to decrypt. Like an encrypted “and”.