CRYPTOGRAPHIC PRIMITIVES

AN INTRODUCTION TO THE THEORY AND PRACTICE BEHIND MODERN CRYPTOGRAPHY
Reactive.IO

Robert Sosinski
Founder & Engineering Fellow
Known as "America's Cryptologic Wing", is the only Air Force wing that supports the National Security Agency, Air Intelligence Agency and the entire United States Air Force (USAF) with cryptologic intelligence.

~ wikipedia.org
AGENDA

• Definition: what cryptography is and is not
• The Basics: making your first algorithm
• Cryptographic Attacks: break your first code
• Symmetric Encryption: one key goes both ways
• Message Digests: many sizes in, one size out
• Asymmetric Encryption: two keys go many ways
• Randomness: don’t just pick a number
• Summary: bringing everything together
• Questions: should be a couple
WHAT IS CRYPTOGRAPHY?
WHAT IS CRYPTOGRAPHY?

Cryptology

hidden or secret

Graphic

writing or text

Cryptography

hidden text
WHAT IS CRYPTOGRAPHY?

The study and practice of securing communication in the presence of adversaries.

Cryptography is a tool that can help provide:

• Confidentiality: adversary cannot read
• Integrity: adversary cannot change
• Authenticity: recipient can trust
• Acknowledged: sender cannot repudiate
Cryptography has many disciplines, known as cryptographic primitives

- Encryption and Decryption
- One-Way Functions (Hash, Digest)
- Authentication
- Digital Signatures
- Entropy and Randomness
MAKING (OR BREAKING) THE SYSTEM

Combining multiple cryptographic primitives together makes a cryptographic system

- SSL and TLS
- PGP and GnuPG
- SSH and VPN
- SFTP and FTPS
- OAuth and SAML
- Even Bitcoin
LET’S MAKE AND BREAK YOUR VERY FIRST CODE

Let’s make a basic cryptographic algorithm and see how it fares
OUR ENCRYPTION ALGORITHM

P
Plaintext

K
Key

F
Function (XOR)

C
Ciphertext
OUR DECRYPTION ALGORITHM

Ciphertext → Function (XOR) → Plaintext

C → F → P
K → F

C = Ciphertext
K = Key
P = Plaintext
**CIPHER AND DECIPHER “A”**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0100 0001</td>
</tr>
<tr>
<td>K</td>
<td>0100 1011</td>
</tr>
<tr>
<td>\n</td>
<td>0000 1010</td>
</tr>
<tr>
<td>\n</td>
<td>0000 1010</td>
</tr>
<tr>
<td>K</td>
<td>0100 1011</td>
</tr>
<tr>
<td>A</td>
<td>0100 0001</td>
</tr>
</tbody>
</table>
CIPHER AND DECIPHER “Z”

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>z</strong></td>
<td>0111 1010</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td>0100 1011</td>
</tr>
<tr>
<td><strong>1</strong></td>
<td>0011 0001</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
<td>0011 0001</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td>0100 1011</td>
</tr>
<tr>
<td><strong>z</strong></td>
<td>0111 1010</td>
</tr>
</tbody>
</table>
PLAIN TEXT ATTACK!

<table>
<thead>
<tr>
<th>A</th>
<th>0100 0001</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>0000 1010</td>
</tr>
<tr>
<td>K</td>
<td>0100 1011</td>
</tr>
<tr>
<td>z</td>
<td>0111 1010</td>
</tr>
<tr>
<td>1</td>
<td>0011 0001</td>
</tr>
<tr>
<td>K</td>
<td>0100 1011</td>
</tr>
</tbody>
</table>
ENSURING KEY PROTECTION

A good cryptographic algorithm protects the key
CRYPTANALYSIS

Cryptology
*hidden or secret*

Analysis
*untie or loosen*

Cryptanalysis
*untie secrets*
CRYPTOGRAPHIC ATTACKS

Known Plaintext Attacks
Attacker knows the plaintext associated with the ciphertext.
- Encrypting information already made public
- Using the same email signature for all recipients

Chosen Plaintext Attacks
Attacker has the message sender encrypt plaintext of his choosing, and gets access to the corresponding ciphertext
- “Gardening” during WWII with minesweeper ships
- Sending queries to a system and matching their output
**SYMMETRIC ENCRYPTION**

Stream Ciphers

Each *plaintext character* is encrypted with a different *key character*.

Benefits are increased speed

Block Ciphers

A *block of plaintext* is manipulated, often several times, with a different *block of key material*.

Benefits are increased security
A5/1 – USED IN GSM PHONES
HARDENING THE ALGORITHM

Confusion

• Finding the relationship between the a key and ciphertext should be as complex and involved as possible.

• The key should be protected from exposure even when an attacker has large amounts of ciphertext to analyze.

Diffusion

• The statistical structure of plaintext should be dissipated over the bulk of ciphertext.

• Changing 1 bit of plaintext should change 50% of the overall ciphertext structure (an avalanche).
MIXING THINGS UP WITH S&P NETWORKS

Substitution

• Take one unique set of input bits and returns another completely different set output bits
• Adds confusion to a cryptographic algorithm

Permutation

• Take one unique set of input bits and returns the same set of output bits, however with different ordering
• Adds diffusion to a cryptographic algorithm
PRODUCT CIPHERS

Block Ciphers that also have the following criteria:

• Operate on a fixed size block of text (e.g. 128bit)
• Utilizes a specific key size (e.g. 128bit, 192bit, 256 bit)
• Undergo multiple rounds of encryption (rounds)
• Keys are expanded via a key schedule
• Each round of encryption performs:
  • Substitution: adding confusion
  • Permutation: adding diffusion
  • Key Mixing: encrypting data
  • Expansion and reduction
KEY EXPANSION

- **K**: Key
- **KS**: Key Schedule
- **EK**: Expanded Key
- **Rotation Permutation**
- **Key Expander (E)**
ROUNDS OF ENCRYPTION

Input Block PT1

Block PT2

Block PT3

Block PT4

Cipher Text CT

F

F

F

F

KS1

KS2

KS3

KS4
EACH ROUND OF ENCRYPTION IS PROCESSED

Input Block $PT_x$

Substitution SBOX

Permutation PBOX

XOR

Output Block $PT_{x+1}$

KS_x
## PERMUTATION BOX - P

<table>
<thead>
<tr>
<th>Input Block</th>
<th>PTx</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 20</td>
<td>15</td>
</tr>
<tr>
<td>5 32</td>
<td>19</td>
</tr>
<tr>
<td>25 1</td>
<td>31</td>
</tr>
<tr>
<td>27 8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permutation PBOX</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 7 20 21 29 12 28 17</td>
</tr>
<tr>
<td>1 15 23 26 5 18 31 10</td>
</tr>
<tr>
<td>2 8 24 14 32 27 3 9</td>
</tr>
<tr>
<td>19 13 30 6 22 11 4 25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output Block</th>
<th>PTx+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 25 39 22 26 18 20 1</td>
<td></td>
</tr>
<tr>
<td>7 24 12 10 8 17 32 14</td>
<td></td>
</tr>
<tr>
<td>4 16 27 20 21 28 19 31</td>
<td></td>
</tr>
<tr>
<td>3 23 2 29 6 9 13 11</td>
<td></td>
</tr>
</tbody>
</table>
## SUBSTITUTION BOX - S1

```
<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>00</th>
<th>01</th>
<th>01</th>
<th>01</th>
<th>01</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>10</th>
<th>11</th>
<th>11</th>
<th>11</th>
<th>11</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>14</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>11</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>15</td>
<td>7</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1</td>
<td>14</td>
<td>8</td>
<td>13</td>
<td>6</td>
<td>2</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>12</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>11</td>
<td>3</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Permutation PBOX**

```
1 0101 0
```
DES - DATA ENCRYPTION STANDARD

- Developed in 1976 by IBM
- Selected by the National Bureau of Standards
- 64 Bit key, although really only 56 Bits, as every 8\textsuperscript{th} Bit is used for error-checking purposes
- 64 Bit block and 16 Rounds of processing
- Often used three times, known as Triple DES
- Not secure since 1999, use AES instead
- You can learn more by reading FIPS46-3
KEY SCHEDULE AND ROUNDS

Key (64 bits)

PC1

Subkey 1 (48 bits)

PC2

Subkey 2 (48 bits)

PC2

Subkey 15 (48 bits)

PC2

Subkey 16 (48 bits)

PC2

Plaintext (64 bits)

IP

F

for 16 rounds

F

FP

Ciphertext (64 bits)
FEISTEL NETWORK

Half Block (32 bits)  →  E  →  Subkey (48 bits) → P

S1  S2  S3  S4  S5  S6  S7  S8
AES - ADVANCED ENCRYPTION STANDARD

- Known as Rijndael, it was developed by Belgian cryptographers Vincent Rijmen and Joan Daemen during the AES competition
- Selected by the NIST to deprecate DES in 2001
- Supports a 128, 192, or 256 Bit Key
- 128 Bit block, that performs 10, 12, or 14 rounds
- When you need to use symmetric encryption, you should probably use AES
- You can learn more by reading FIPS 197
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>63</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>f2</td>
<td>6b</td>
<td>6f</td>
<td>d5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2b</td>
<td>fe</td>
<td>d7</td>
<td>ab</td>
<td>76</td>
</tr>
<tr>
<td>10</td>
<td>ca</td>
<td>82</td>
<td>c9</td>
<td>7d</td>
<td>fa</td>
<td>59</td>
<td>47</td>
<td>f0</td>
<td>ad</td>
<td>d4</td>
<td>a2</td>
<td>af</td>
<td>9c</td>
<td>a4</td>
<td>72</td>
<td>c0</td>
</tr>
<tr>
<td>20</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>26</td>
<td>36</td>
<td>3f</td>
<td>f7</td>
<td>cc</td>
<td>34</td>
<td>a5</td>
<td>e5</td>
<td>f1</td>
<td>71</td>
<td>d8</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>04</td>
<td>c7</td>
<td>23</td>
<td>c3</td>
<td>18</td>
<td>96</td>
<td>05</td>
<td>9a</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>e2</td>
<td>eb</td>
<td>27</td>
<td>b2</td>
<td>75</td>
</tr>
<tr>
<td>40</td>
<td>09</td>
<td>83</td>
<td>2c</td>
<td>1a</td>
<td>1b</td>
<td>6e</td>
<td>5a</td>
<td>a0</td>
<td>52</td>
<td>3b</td>
<td>d6</td>
<td>b3</td>
<td>29</td>
<td>e3</td>
<td>2f</td>
<td>84</td>
</tr>
<tr>
<td>50</td>
<td>53</td>
<td>d1</td>
<td>00</td>
<td>ed</td>
<td>20</td>
<td>fc</td>
<td>b1</td>
<td>fb</td>
<td>6a</td>
<td>cb</td>
<td>be</td>
<td>39</td>
<td>4a</td>
<td>4c</td>
<td>58</td>
<td>cf</td>
</tr>
<tr>
<td>60</td>
<td>d0</td>
<td>ef</td>
<td>aa</td>
<td>fb</td>
<td>43</td>
<td>4d</td>
<td>33</td>
<td>85</td>
<td>45</td>
<td>f9</td>
<td>02</td>
<td>7f</td>
<td>50</td>
<td>3c</td>
<td>9f</td>
<td>a8</td>
</tr>
<tr>
<td>70</td>
<td>51</td>
<td>a3</td>
<td>40</td>
<td>8f</td>
<td>92</td>
<td>9d</td>
<td>38</td>
<td>f5</td>
<td>bc</td>
<td>b6</td>
<td>da</td>
<td>21</td>
<td>10</td>
<td>ff</td>
<td>f3</td>
<td>d3</td>
</tr>
<tr>
<td>80</td>
<td>cd</td>
<td>0c</td>
<td>13</td>
<td>ec</td>
<td>5f</td>
<td>97</td>
<td>44</td>
<td>17</td>
<td>c4</td>
<td>a7</td>
<td>7e</td>
<td>3d</td>
<td>64</td>
<td>5d</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>90</td>
<td>60</td>
<td>81</td>
<td>4f</td>
<td>dc</td>
<td>22</td>
<td>2a</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>ee</td>
<td>b8</td>
<td>14</td>
<td>de</td>
<td>5e</td>
<td>0b</td>
<td>db</td>
</tr>
<tr>
<td>a0</td>
<td>e0</td>
<td>32</td>
<td>3a</td>
<td>0a</td>
<td>49</td>
<td>06</td>
<td>24</td>
<td>5c</td>
<td>c2</td>
<td>d3</td>
<td>ac</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>e4</td>
<td>79</td>
</tr>
<tr>
<td>b0</td>
<td>e7</td>
<td>c8</td>
<td>37</td>
<td>d6</td>
<td>8d</td>
<td>d5</td>
<td>4e</td>
<td>a9</td>
<td>6c</td>
<td>56</td>
<td>f4</td>
<td>ea</td>
<td>65</td>
<td>7a</td>
<td>ae</td>
<td>08</td>
</tr>
<tr>
<td>c0</td>
<td>ba</td>
<td>78</td>
<td>25</td>
<td>2e</td>
<td>1c</td>
<td>a6</td>
<td>b4</td>
<td>c6</td>
<td>c8</td>
<td>dd</td>
<td>74</td>
<td>1f</td>
<td>4b</td>
<td>bd</td>
<td>8b</td>
<td>8a</td>
</tr>
<tr>
<td>d0</td>
<td>70</td>
<td>3e</td>
<td>b5</td>
<td>66</td>
<td>48</td>
<td>03</td>
<td>f6</td>
<td>0e</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>b9</td>
<td>86</td>
<td>c1</td>
<td>1d</td>
<td>94</td>
</tr>
<tr>
<td>e0</td>
<td>e1</td>
<td>f8</td>
<td>98</td>
<td>11</td>
<td>69</td>
<td>d9</td>
<td>8e</td>
<td>94</td>
<td>9b</td>
<td>1e</td>
<td>87</td>
<td>e9</td>
<td>ce</td>
<td>55</td>
<td>28</td>
<td>df</td>
</tr>
<tr>
<td>f0</td>
<td>8c</td>
<td>a1</td>
<td>89</td>
<td>0d</td>
<td>bf</td>
<td>e6</td>
<td>42</td>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2d</td>
<td>0f</td>
<td>b0</td>
<td>54</td>
<td>bb</td>
<td>16</td>
</tr>
</tbody>
</table>
SUB BYTES

\[
\begin{array}{cccc}
a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\
a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\
a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\
a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\
\end{array}
\]

\[
\begin{array}{cccc}
b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\
b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\
b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\
b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\
\end{array}
\]
### SHIFT ROWS

<table>
<thead>
<tr>
<th>No change</th>
<th>$a_{0,0}$</th>
<th>$a_{0,1}$</th>
<th>$a_{0,2}$</th>
<th>$a_{0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift 1</td>
<td>$a_{1,0}$</td>
<td>$a_{1,1}$</td>
<td>$a_{1,2}$</td>
<td>$a_{1,3}$</td>
</tr>
<tr>
<td>Shift 2</td>
<td>$a_{2,0}$</td>
<td>$a_{2,1}$</td>
<td>$a_{2,2}$</td>
<td>$a_{2,3}$</td>
</tr>
<tr>
<td>Shift 3</td>
<td>$a_{3,0}$</td>
<td>$a_{3,1}$</td>
<td>$a_{3,2}$</td>
<td>$a_{3,3}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$a_{0,0}$</th>
<th>$a_{0,1}$</th>
<th>$a_{0,2}$</th>
<th>$a_{0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{1,1}$</td>
<td>$a_{1,2}$</td>
<td>$a_{1,3}$</td>
<td>$a_{1,0}$</td>
</tr>
<tr>
<td>$a_{2,2}$</td>
<td>$a_{2,3}$</td>
<td>$a_{2,0}$</td>
<td>$a_{2,1}$</td>
</tr>
<tr>
<td>$a_{3,3}$</td>
<td>$a_{3,0}$</td>
<td>$a_{3,1}$</td>
<td>$a_{3,2}$</td>
</tr>
</tbody>
</table>
MIX COLUMNS

\[
\begin{align*}
&a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\
&a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\
&a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\
&a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\
\end{align*}
\]

\[
\begin{align*}
&b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\
&b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\
&b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\
&b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\
\end{align*}
\]

\[\otimes c(x)\]
ADD ROUND KEY
CIPHER MODES

When the amount of plaintext exceeds the block size of an algorithm, a cipher mode must be used.

**ECB - Electronic Code Book**
- Message is divided into blocks and encrypted.

**CBC - Cipher Block Chaining**
- Each block is XORed with previous block.

**Counter**
- Generates a key stream by incrementing a counter or making a nonce (number used once).
ELECTRONIC CODE BOOK

Electronic Codebook (ECB) mode encryption
QUALITY OF ENCRYPTION

ECB

Before

After
CIPHER BLOCK CHAINING

Cipher Block Chaining (CBC) mode encryption
QUALITY OF ENCRYPTION
CBC

Before

After
Counter (CTR) mode encryption
Cryptographic Digests, also known as Hashing Functions or One-Way Functions:

1. A **variable length input** (Pre Image) relates deterministically to a **fixed length output** (Digest)

2. The process of computing a pre image to a digest should be computationally easy

3. The process of inversely relating a digest to the it’s corresponding pre image should be extremely difficult and computationally expensive
DIGESTS AT WORK

P1 → Function → D1
P1 → D2
P2 → Function → D2
P2 → D3
P3 → Function → D3
P3 → D1
TYPES OF DIGEST FUNCTIONS

• Any Block Cipher: Such as AES or DES
• MD5: Machine Digest (version 5)
• SHA1: Secure Hashing Algorithm
• SHA2: 224 Bit through 512 Bit
• Whirlpool: Made by the designers of AES
• RIPEMD160: Designed in the open
WHEN TO USE DIGESTS

Generating unique “fingerprints” for complex pre images
• Giving each multi-gigabyte video a unique id
• Compiling multiple data points into a single opaque

Concealing pre images while preserving relationships
• Storing passwords securely in a database
• Storing personally identifiable information as an opaque

Verifying data integrity
• Digest large files before and after transmission; if both digests match, then the file is not corrupt.
• Send the digest though a different medium then the file, if both digests match, then the file was not modified.
MESSAGE AUTHENTICATION CODES

PI
Pre Image

Hash Function

Key or Salt

MAC
WHEN TO USE MACS

More secure digests
• Using a unique “Salt” before digesting password

Verifying authenticity
• If the MAC of the received file matches the MAC sent with the file, then the sender can be authenticated

Verifying data integrity
• Any modification of the data by an adversary while in transit will result in a different MAC

Key Recycling
• Having two keys, a “Key Encryption Key” and “Transmission Key”. When you digest both, the resulting output is the key.
ASYMMETRIC ENCRYPTION

Two different, but mathematically linked keys (one public, one private) that secure data when used in pairs.

Benefits:
- Key Exchange can be performed within a system that is compromised by an adversary.
- Keys can be used to encrypt or to sign data.

Drawbacks:
- Key Generation is computationally expensive
- Encryption is slow and relies on integer factorization
- Not possible to use ciphering modes
PUBLIC / PRIVATE ENCRYPTION

Bob

Hello Alice!
Encrypt
6EB69570 08E03CE4
Alice's public key

Alice

Hello Alice!
Decrypt
Alice's private key
SHARED SECRET

Alice

Bob's public key

Alice's private key

Combine keys

751A696C24D97009

Alice and Bob's shared secret

Bob

Alice's public key

Bob's private key

Combine keys

751A696C24D97009

Alice and Bob's shared secret
RSA

The first practicable public-key cryptographic algorithm.

- Invented by Ron Rivest, Adi Shamir, and Leonard Adleman
- Published in 1977, used in many cryptosystems including SSL, SSH and PGP/GnuPG
- Typical Key Sizes are from 1024 through 4096
- Relies on the mathematical difficulty of factoring the product of two prime numbers
- RSA have been broken with key sizes of 768 bits and lower, 768 was broken on December 12th 2009
GENERATING RSA KEYS

1. Pick two large, random prime numbers; called $p$ and $q$
   
   $p = 11$
   $q = 5$

2. Multiply $p$ and $q$ to calculate the RSA modulus; called $n$
   
   $n = p \times q$
   $n = 11 \times 5$
   $n = 55$

3. Calculate the Totient of the RSA modulus; called $z$
   
   $z = (p - 1) \times (q - 1)$
   $z = (11 - 1) \times (5 - 1)$
   $z = 10 \times 4$
   $z = 40$
GENERATING RSA KEYS CONTINUED

4. Select a co-prime for z for your public key; called e
   \[ 1 < e < z \text{ and } \gcd(e, z) = 1 \]
   \[ e = 7 \]

5. Find the multiplicative inverse of e mod z by extended Euclidian algorithm for your private key; called d
   \[ e \times d \text{ mod } z = 1 \]
   \[ 7 \times d \text{ mod } 40 = 1 \]
   \[ d = 23 \]
The **Public Key** is the modulus $n$ and exponent $e$

$$pub = (n, e)$$

$$pub = (55, 7)$$

The **Private Key** is the modulus $n$ and exponent $d$

$$pri = (n, d)$$

$$pri = (55, 23)$$

You can now share your **Public Key** with everyone, however you must ensure that your **Private Key** stays a closely held secret.
ENCRYPTING DATA WITH RSA

1. Choose an integer value for the message; $m$ where $0 \leq m < n$
   $m = 42$

2. Solve for the ciphered message; $c$ with
   the exponent $e$ and modulus $n$
   $c = m^e \mod n$
   $c = 42^7 \mod 55$
   $c = 48$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>11</td>
</tr>
<tr>
<td>$q$</td>
<td>5</td>
</tr>
<tr>
<td>$n$</td>
<td>55</td>
</tr>
<tr>
<td>$z$</td>
<td>40</td>
</tr>
<tr>
<td>$e$</td>
<td>7</td>
</tr>
<tr>
<td>$d$</td>
<td>23</td>
</tr>
<tr>
<td>pub</td>
<td>(55,7)</td>
</tr>
<tr>
<td>pri</td>
<td>(55,23)</td>
</tr>
</tbody>
</table>
1. Solve for the plaintext message; $m$ with the exponent $d$ and modulus $n$
   $m = c^d \mod n$
   $m = 48^{23} \mod 55$
   $m = 42$

<table>
<thead>
<tr>
<th>$p$</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>5</td>
</tr>
<tr>
<td>$n$</td>
<td>55</td>
</tr>
<tr>
<td>$z$</td>
<td>40</td>
</tr>
<tr>
<td>$e$</td>
<td>7</td>
</tr>
<tr>
<td>$d$</td>
<td>23</td>
</tr>
<tr>
<td>$pub$</td>
<td>(55,7)</td>
</tr>
<tr>
<td>$pri$</td>
<td>(55,23)</td>
</tr>
<tr>
<td>$m$</td>
<td>42</td>
</tr>
<tr>
<td>$c$</td>
<td>48</td>
</tr>
</tbody>
</table>
HOW TO USE RSA

Transmit Keying Material
1. Randomly choose a key for a symmetric algorithm
2. Encrypt the key with an RSA public key and transmit
3. Utilize the symmetric cipher for transmitting data

Digitally Sign Documents
1. Digest the document into a smaller fingerprint
2. Encrypt the digest with the senders RSA private key
3. Recipient decrypts the signature with the senders public key
4. Match the digests of the received document with the signature
DIFFIE-HELLMAN KEY EXCHANGE

Instead of generating a key that can encrypt or sign data, generates material that allows two parties to generate a shared secret for use by symmetric algorithms.

- Invented by Whitfield Diffie and Martin Hellman in 1976
- Key generation is faster, as only one large prime number is needed and the process is simpler
- As keys are quicker to generate, they can be once per transmission session; aiding in Forward Security
- By itself cannot be used to digitally sign documents, but is used by other cryptosystems that do so
1. Both parties choose a shared prime number; p and base; g
   \[ p = 23 \]
   \[ g = 5 \]

2. Both parties generate a secret integer; a and b
   \[ a = 6 \]
   \[ b = 15 \]

3. The A side calculates their public transport; A
   \[ A = g^a \mod p \]
   \[ A = 5^6 \mod 23 \]
   \[ A = 8 \]
4. The B side calculates their public transport; \( B \)
   \[
   B = g^b \mod p \\
   B = 5^{15} \mod 23 \\
   B = 19
   \]

5. The A side calculates the secret key; \( s \)
   \[
   s = B^a \mod p \\
   s = 19^6 \mod p \\
   s = 2
   \]

6. The B side calculates the same secret key; \( s \)
   \[
   s = A^b \mod p \\
   s = 8^{15} \mod p \\
   s = 2
   \]
Alice

Common paint

Secret colours

Bob

Common paint

Secret colours

Public transport

(assume that mixture separation is expensive)
OTHER ASYMMETRIC ALGORITHMS

ElGamal

- Optimized for encrypting data due to being probabilistic
- A plaintext message can produce multiple ciphertexts
- Based off of the Diffie-Hellman key exchange protocol

DSA - Digital Signature Algorithm

- Optimized for securely producing digital signatures
- Is a variant of ElGamal and is used in PGP and GnuPG
- Chosen by NIST as the Digital Signature Standard
LAST BUT NOT LEAST, SOMETHING RANDOM

Creating proper cryptographic keys high levels of entropy

1. If there is a pattern used to generate a key, then the pattern, and thus the key, can be guessed

2. If there are a limited number of keys, the surface area for an attack is smaller, and the impact is larger

3. If an attacker controls the random generator, then the attacker controls key generation

4. Randomness is very hard to obtain with computers, so we often we settle with pseudo-randomness

5. Finding pseudo-random prime numbers is very costly
GETTING RANDOM DATA

/dev/random
High Quality and High Cost
Will block if entropy pool is empty

/dev/urandom
Medium Quality and Low Cost
Will digest new data if pool is empty
SUMMARY

The Basics: what cryptography is at a fundamental level

Symmetric Encryption: fast and well understood algorithms

Digest Functions: great for fingerprints and opaque tokens

Asymmetric Encryption: important part of any crypto system

Randomness: hard to generate but very important to have
Understanding your tools lets you make more secure systems
QUESTIONS

Reactive.IO

Robert Sosinski
robert.sosinski@reactive.io