Announcements

Homework 1 is released

- Available on the course website
- Due in **two weeks**: 10/22/19 11:59pm
- Submit through **GradeScope**
  - TA Sam gave a tutorial last Wednesday
Lecture 4

Encryption II

Suggested Readings:

• Chs 3 & 4 in KPS (recommended)
• Ch 3 in Stinson (optional)

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]
Conventional (Symmetric) Cryptography

plaintext $m$ $\xrightarrow{K_{AB}}$ ciphertext $\xrightarrow{K_{AB}}$ plaintext

$m = K_{AB}(K_{AB}(m))$
“Modern” Block Ciphers
Data Encryption Standard (DES)
Encryption Process

64 Bit Plaintext

Initial Permutation

32 Bit L₀  32 Bit R₀

F(R₀, K₁)

32 Bit L₁  32 Bit R₁

32 Bit L₁₅  32 Bit R₁₅

F(R₁₅, K₁₆)

32 Bit L₁₆  32 Bit R₁₆

Final Permutation

64 Bit Ciphertext

Key Schedule

64 Bit Key

Permutation Choice 1

56 Bit Key

28 Bit C₀  28 Bit D₀

Left Shift

C₁  D₁

K₁(48 bits)

K₁₆(48 bits)

Permutation Choice 2

C₁₆  D₁₆

Building Blocks

Permutation Choice 2

27
Function F

- $L_{i-1}$ 32 bits
- $R_{i-1}$ 32 bits
- Expansion (E)
  - Permutation 48 bits
- S-Box
  - Substitution
  - choses 32 bits
- P-box Permutation
- 56 bits Key
  - Permuted Choice
  - 48 bits
- $L_i$ 32 bits
- $R_i$ 32 bits
**DES Substitution Boxes Operation**

<table>
<thead>
<tr>
<th>Bit</th>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
<th>$B_4$</th>
<th>$B_5$</th>
<th>$B_6$</th>
<th>$B_7$</th>
<th>$B_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>1-6</td>
<td>7-12</td>
<td>13-18</td>
<td>19-24</td>
<td>25-30</td>
<td>31-36</td>
<td>37-42</td>
<td>43-48</td>
</tr>
<tr>
<td>Bits</td>
<td>1-4</td>
<td>5-8</td>
<td>9-12</td>
<td>13-16</td>
<td>14-20</td>
<td>21-24</td>
<td>25-28</td>
<td>29-32</td>
</tr>
</tbody>
</table>

Expanded $R_{i-1} \oplus \text{Key}$

![Diagram of substitution boxes]
### Operation Tables of DES (IP, IP⁻¹, E and P)

#### Initial Pemutation (IP)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>50</td>
<td>42</td>
<td>34</td>
<td>26</td>
<td>18</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>52</td>
<td>44</td>
<td>36</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>62</td>
<td>54</td>
<td>46</td>
<td>38</td>
<td>30</td>
<td>22</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>64</td>
<td>56</td>
<td>48</td>
<td>40</td>
<td>32</td>
<td>24</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>57</td>
<td>49</td>
<td>41</td>
<td>33</td>
<td>25</td>
<td>17</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>59</td>
<td>51</td>
<td>43</td>
<td>35</td>
<td>27</td>
<td>19</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>61</td>
<td>53</td>
<td>45</td>
<td>37</td>
<td>29</td>
<td>21</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>63</td>
<td>55</td>
<td>47</td>
<td>39</td>
<td>31</td>
<td>23</td>
<td>15</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Inverse Initial Pemutation (IP⁻¹)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>8</td>
<td>48</td>
<td>16</td>
<td>56</td>
<td>24</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>39</td>
<td>7</td>
<td>47</td>
<td>15</td>
<td>55</td>
<td>23</td>
<td>63</td>
<td>31</td>
</tr>
<tr>
<td>38</td>
<td>6</td>
<td>46</td>
<td>14</td>
<td>54</td>
<td>22</td>
<td>62</td>
<td>30</td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>45</td>
<td>13</td>
<td>53</td>
<td>21</td>
<td>61</td>
<td>29</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>44</td>
<td>12</td>
<td>52</td>
<td>20</td>
<td>60</td>
<td>28</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>43</td>
<td>11</td>
<td>51</td>
<td>19</td>
<td>59</td>
<td>27</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>42</td>
<td>10</td>
<td>50</td>
<td>18</td>
<td>58</td>
<td>26</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>41</td>
<td>9</td>
<td>49</td>
<td>17</td>
<td>57</td>
<td>25</td>
</tr>
</tbody>
</table>

#### Bit-Selection Table E

|   |   |   |   |   |   |
|---|---|---|---|---|
| 32 |  1 |  2 |  3 |  4 |  5 |
|  4 |  5 |  6 |  7 |  8 |  9 |
|  8 |  9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 |  1 |

#### Permutation P

|   |   |   |   |   |
|---|---|---|---|
| 16 |  7 | 20 | 21 |
| 19 | 12 | 18 | 17 |
|  1 | 15 | 23 | 26 |
|  5 | 18 | 31 | 10 |
|  2 |  8 | 24 | 14 |
| 32 | 27 |  3 |  9 |
| 19 | 13 | 30 |  6 |
| 22 | 11 |  4 | 25 |
Key Schedule -- KS
### Key schedule of shifts

<table>
<thead>
<tr>
<th>Iteration(i)</th>
<th>No. of shifts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>

### Key permutation PC-1

<table>
<thead>
<tr>
<th>57</th>
<th>49</th>
<th>41</th>
<th>33</th>
<th>25</th>
<th>17</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58</td>
<td>50</td>
<td>42</td>
<td>34</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>59</td>
<td>51</td>
<td>43</td>
<td>35</td>
<td>27</td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>3</td>
<td>60</td>
<td>52</td>
<td>44</td>
<td>36</td>
</tr>
<tr>
<td>63</td>
<td>55</td>
<td>47</td>
<td>39</td>
<td>31</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>62</td>
<td>54</td>
<td>46</td>
<td>38</td>
<td>30</td>
<td>22</td>
</tr>
<tr>
<td>14</td>
<td>6</td>
<td>61</td>
<td>53</td>
<td>45</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>13</td>
<td>5</td>
<td>28</td>
<td>20</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>

### Key permutation PC-2

<table>
<thead>
<tr>
<th>14</th>
<th>17</th>
<th>11</th>
<th>24</th>
<th>1</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>28</td>
<td>15</td>
<td>6</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>23</td>
<td>19</td>
<td>12</td>
<td>4</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
<td>27</td>
<td>20</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>41</td>
<td>52</td>
<td>31</td>
<td>37</td>
<td>47</td>
<td>55</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>51</td>
<td>45</td>
<td>33</td>
<td>48</td>
</tr>
<tr>
<td>44</td>
<td>49</td>
<td>39</td>
<td>56</td>
<td>34</td>
<td>54</td>
</tr>
<tr>
<td>46</td>
<td>42</td>
<td>50</td>
<td>36</td>
<td>29</td>
<td>32</td>
</tr>
</tbody>
</table>

**Operation Tables of DES**

(Key Schedule, PC-1, PC-2)
Breaking DES (Cryptanalysis)

DES Key size = 56 bits

• Brute force = $2^{55}$ attempts on avg
• Differential cryptanalysis $\Rightarrow 2^{47}$ chosen plaintexts [BS’89]
• Linear cryptanalysis $\Rightarrow 2^{43}$ known plaintexts [M’93]

• More than 16 rounds do not make it any stronger

• DES Key Problems:
  • Weak keys (all 0s, all 1s, a few others)
  • Key size = 56 bits = 8 * 7-bit ASCII
  • Alphanumeric-only password converted to uppercase
    8 * $\sim$5-bit chars = 40 bits
Modes of Operation
(not just for DES, for any block cipher)

\[
\begin{align*}
P_1 & \quad P_2 & \quad \cdots & \quad P_i & \quad P_{i+1} & \quad \cdots & \quad P_{n-1} & \quad P_n \\
C_1 & \quad C_2 & \quad \cdots & \quad C_i & \quad C_{i+1} & \quad \cdots & \quad C_{n-1} & \quad C_n \\
\end{align*}
\]

http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
"Native" ECB Mode

Electronic Code-Book (ECB) Mode

• Input to encryption algorithm is current plaintext block:

\[ C_i = E(K, P_i) \]
\[ P_i = D(K, C_i) \]

• Duplicate plaintext blocks (patterns) visible in ciphertext
  • What if Alice encrypts one word per plaintext block?

• Ciphertext block rearrangement is possible
  • To detect it, need explicit block numbering in plaintext

• Parallel encryption and decryption (random access)

• Error in one ciphertext block \( \rightarrow \) one-block loss

• One-block loss in ciphertext?
CBC Mode

Cipher-Block Chaining (CBC) Mode

• Input to encryption algorithm is the XOR of current plaintext block and preceding ciphertext block:

\[ C_i = E(K, P_i \text{ XOR } C_{i-1}) \quad C_0=\text{IV} \]
\[ P_i = D(K, C_i \text{ XOR } C_{i-1}) \]

• Duplicate plaintext blocks (patterns) NOT exposed
• Block rearrangement is detectable
• No parallel encryption
  • How about parallel decryption?
• Error in one ciphertext block \(\Rightarrow\) two-block loss
• One-block ciphertext loss?
Figure 2.7 Cipher Block Chaining (CBC) Mode
OFB Mode

Output Feedback (OFB) Mode

• Key-stream is produced by repeated encryption of $V_0$:

\[
C_i = E \left( K, V_{i-1} \right) \text{XOR} P_i \\
V_0 = IV, \ldots, V_i = E \left( K, V_{i-1} \right) \\
P_i = E \left( K, V_{i-1} \right) \text{XOR} C_i
\]

• Duplicate plaintext blocks (patterns) NOT exposed
• Block rearrangement is detectable
• Key-stream is independent of plaintext
  • How does that affect speed of encryption? Parallelism?
• Bit error in one ciphertext block $\Rightarrow$ one-bit error in plaintext
• One-block ciphertext loss $\Rightarrow$ big mess 😊
• Can encrypt less than block size
CFB Mode

Cipher Feedback (CFB) Mode

• Key-stream is produced by re-encryption of preceding ciphertext -- \( C_{i-1} \):

\[
C_i = P_i \ XOR \ E(K, C_{i-1}) \quad C_0 = IV
\]
\[
P_i = E(K, C_{i-1}) \ XOR \ C_i
\]

• Duplicate plaintext blocks (patterns) NOT exposed
• Block rearrangement is detectable

• Key-stream is **dependent on** plaintext
  • How does that affect speed of encryption? Parallelism?

• Bit error in one ciphertext block \( \Rightarrow \) one-bit + one-block loss in plaintext
  • Adversary can still selectively flip/change bits

• One-block ciphertext loss \( \Rightarrow \) 1-extra-block loss
• Can encrypt less than block size
CTR Mode

Counter (CTR) Mode

• Key-stream is produced by encryption increasing counter:

\[ C_i = \text{E} ( K, \text{CTR} ) \text{ XOR } P_i \quad \text{CTR} ++\]

\[ P_i = \text{E} ( K, \text{CTR} ) \text{ XOR } C_i \]

• Duplicate plaintext blocks (patterns) NOT exposed, unless?
• Block rearrangement is detectable
• Key-stream is independent of plaintext
• Parallel encryption and decryption (random access)
• Bit error in one ciphertext block \(\rightarrow\) one-bit error in plaintext
• One-block ciphertext loss \(\rightarrow\) big mess
• Can encrypt less than block size
MAC Mode

Message Authentication Code (MAC) Mode

• Encryption is the same as in CBC mode, but, ciphertext is NOT sent!

\[ C_i = E(K, P_i \text{ XOR } C_{i-1}) \quad C_0 = IV \]

What is sent or stored: \( P_1, \ldots, P_n, C_n = \text{MAC} \)

Receiver recomputes \( C_n \) with \( K \) and compares

• Any change in plaintext results in unpredictable changes in MAC
How to strengthen DES: the case of double DES

- **2DES:**  \( C = \text{DES} ( K_1, \text{DES} ( K_2, P ) ) \)
- Seems to be hard to break by “brute force”, approx. \( 2^{111} \) trials
- Assume Eve is trying to break 2DES and has a single \((P,C)\) pair

**Meet-in-the-middle ATTACK:**

<table>
<thead>
<tr>
<th>I. For each possible ( K'_i ) (where ( 0 &lt; i &lt; 2^{56} ))</th>
<th>II. For each possible ( K''_i ) (where ( 0 &lt; i &lt; 2^{56} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Compute ( C'_i = \text{DES} ( K'_i, P ) )</td>
<td>1. Compute ( C''_i = \text{DES}^{-1} ( K''_i, C ) )</td>
</tr>
<tr>
<td>2. Store: ([C'_i, K'_i]) in look-up table ( T ) (indexed by ( C'_i ))</td>
<td>2. Look up ( C''_i ) in ( T )</td>
</tr>
<tr>
<td></td>
<td>3. If lookup succeeds, output: ( K1=K'_i, K2=K''_i )</td>
</tr>
</tbody>
</table>

**TOTAL COST:** \( O(2^{56} + 2^{56}) \) operations + \( O(2^{64}) \) storage
DES Variants

- **2-DES:**
  - $C = E(K_2, E(K_1, P)) \rightarrow 57$ effective key bits (meet-in-the-middle attack)

- **3-DES (Triple DES):**
  - $C = E(K_3, D(K_2, E(K_1, P))) \rightarrow 112$ effective key bits (meet-in-the-middle attack)
  - $C = E(K_1, D(K_2, E(K_1, P))) \rightarrow \leq 80$ effective key bits

- **DESX**
  - $C = K_3 \text{ XOR } E(K_2, (K_1 \text{ XOR } P)) \rightarrow$ seems like 184 key bits
  - Effective key bits $\rightarrow$ approx. 118

- **Another simple variation:**
  - $C = K_2 \text{ XOR } E(K_1, P) \rightarrow$ weak!

**NOTE:** The same variants can be constructed out of any cipher
Why does 3-DES (or generally n-DES) work?

Because, as a function, DES is not a group...

A “group” is an algebraic structure. One of its properties is that, taking any 2 elements of the group \((a,b)\) and applying an operator \(F()\) yields another element \(c\) in the group.

Suppose: \(C = \text{DES}(K_1, \text{DES}(K_2, P))\)

There is no \(K\), such that:

for each possible plaintext \(P\), \(\text{DES}(K, P) = C\)
DES Summary

- Feistel network based block cipher
- 64-bit data blocks
- 56-bit keys (8 parity bits)
- 16 rounds (shifts, XORs)
- Key schedule
- S-box selection secret ...

- DES “aging”
- 2-DES: meet-in-the-middle attack
- 3-DES: 112-bit security
- DESX: 118-bit security
Advanced Encryption Standard (AES): The Rijndael Block Cipher
Introduction and History

• National Institute of Science and Technology (NIST) regulates standardization in the US
• By mid-90s, DES was an aging standard that no longer met the needs for strong commercial-grade encryption
• Triple-DES: Endorsed by NIST as a “de facto” standard
• But ... slow in software and large footprint (code size)
• Advanced Encryption Standard (AES)
  • Goal is to define the Federal Information Processing Standard (FIPS) by selecting a new encryption algorithm suitable for encrypting (non-classified non-military) government documents
  • Candidate algorithms must be:
    • Symmetric-key ciphers supporting 128, 192, and 256 bit keys
    • Royalty-Free
    • Unclassified (i.e., public domain)
    • Available for worldwide export
  • 1997: NIST publishes request for proposal
  • 1998-1999: 15 submissions -> 5 finalists
  • 2000: NIST chooses Rijndael as AES
Introduction and History

• AES Round-3 Finalist Algorithms (ranked by vote # in AES Round-2, high to low):
  • Rijndael
    • by Joan Daemen and Vincent Rijmen (Belgium)
  • Serpent
    • by Ross Anderson (UK), Eli Biham (ISR) and Lars Knudsen (NO)
  • Twofish
    • From Counterpane Internet Security, Inc. (MN)
  • RC6
    • By Ron Rivest of MIT & RSA Labs, creator of the widely used RC4/RC5 algorithm and “R” in RSA
  • MARS
    • Candidate offering from IBM Research
The Winner: Rijndael

• Joan Daemen (of Proton World International) and Vincent Rijmen (of Katholieke Universiteit Leuven).

• Pronounced “Rhine-doll”

• Allows only 128, 192, and 256-bit key sizes (unlike other candidates)

• Variable input block length: 128, 192, or 256 bits. All nine combinations of key-block length possible.
  • A block is the smallest data size the algorithm will encrypt

• Vast speed improvement over DES in both hw and sw implementations
  • 8,416 bytes/sec on a 20MHz 8051
  • 8.8 Mbytes/sec on a 200MHz Pentium Pro
Rijndael

Key Expansion

Key Expansion

Encryption Rounds $r_1 \ldots r_n$

- Key is expanded to a set of $n$ round keys
- Input block $P$ put thru $n$ rounds, each with a distinct round sub-key.
- Strength of algorithm relies on difficulty of obtaining intermediate results (or state) of round $i$ from round $i+1$ without the round key.
Each round performs the following operations:

- **Non-linear Layer**: No linear relationship between the input and output of a round
- **Linear Mixing Layer**: Guarantees high diffusion over multiple rounds
  - Very small correlation between bytes of the round input and the bytes of the output
- **Key Addition Layer**: Bytes of the input are simply XOR’ed with the expanded round key
Three layers provide strength against known types of cryptographic attacks: Rijndael provides “full diffusion” after only two rounds.

Cryptanalysis

- Key recovery attack:
  - Best one only 4x faster than exhaustive search [BKR’11]

- Related key attack:
  - AES-256: Given $2^{99}$ input/output pairs from 4 related keys in AES-256 can recover keys in time $2^{99}$ [BK’09]
  - However, how realistic is that?
Rijndael: ByteSub

Each byte at the input of a round undergoes a non-linear byte substitution according to the following transform:

\[
\begin{bmatrix}
    y_0 \\
    y_1 \\
    y_2 \\
    y_3 \\
    y_4 \\
    y_5 \\
    y_6 \\
    y_7
\end{bmatrix}
= \begin{bmatrix}
    1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
    1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
    1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
    1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
    1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
    0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
    0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
    x_0 \\
    x_1 \\
    x_2 \\
    x_3 \\
    x_4 \\
    x_5 \\
    x_6 \\
    x_7
\end{bmatrix}
+ \begin{bmatrix}
    1 \\
    1 \\
    0 \\
    1 \\
    0 \\
    1 \\
    1 \\
    0
\end{bmatrix}
\]

Substitution ("S")-box
Rijndael: ShiftRow

Depending on the block length, each “row” of the block is cyclically shifted according to the above table.
Rijndael: MixColumn

Each column is multiplied by a fixed polynomial

\[ C(x) = '03'x^3 + '01'x^2 + '01'x + '02' \]

This corresponds to matrix multiplication \( b(x) = c(x) \otimes a(x) \):
Rijndael: Key Expansion and Addition

Each word is simply XOR'ed with the expanded round key

Key Expansion algorithm:

```
KeyExpansion(int* Key[4*Nk], int* EKey[Nb*(Nr+1)])
{
    for(i = 0; i < Nk; i++)
        EKey[i] = (Key[4*i], Key[4*i+1], Key[4*i+2], Key[4*i+3]);
    for(i = Nk; i < Nb * (Nr + 1); i++)
    {
        temp = EKey[i - 1];
        if (i % Nk == 0)
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
        EKey[i] = EKey[i - Nk] ^ temp;
    }
}
```