YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467b: Cryptography and Computer Security *Handout #11*
Xueyuan Su *March 3, 2010*

Solutions to Problem Set 3

Due on Tuesday, February 23, 2010.

In the problems below, "textbook" refers to Wade Trapp and Lawrence C. Washington, *Introduction to Cryptography with Coding Theory, Second Edition*, Prentice-Hall, 2006.

Problem 1: Simplified DES

Textbook, problem 4-1.
Solution:

(a) From the encryption function $M_{j+2} = M_j \oplus f(K, M_{j+1})$, we know that

$$M_j = M_j \oplus f(K, M_{j+1}) \oplus f(K, M_{j+1}) = M_{j+2} \oplus f(K, M_{j+1})$$

Therefore, given $M_m M_{m+1}$, we compute

$$M_{m-1} = M_{m+1} \oplus f(K, M_m)$$

Repeating the same process until M_0 is computed. Then $M_0 M_1$ is the corresponding plaintext.

(b) If the encryption runs for 2 rounds, then the resulting ciphertext is $M_2 M_3$, where $M_2 = M_0 \oplus f(K, M_1) = M_0 \oplus M_1 \oplus K$, and $M_3 = M_1 \oplus f(K, M_2) = M_1 \oplus M_0 \oplus M_1 \oplus K \oplus K = M_0$. Therefore, the first part of the plaintext is known as $M_0 = M_3$, but neither the second part of the plaintext M_1 nor the key K is known. However, if you know a plaintext-ciphertext pair, then it is easy to deduce K by the following operation

$$M_2 \oplus M_0 \oplus M_1 = M_0 \oplus M_1 \oplus K \oplus M_0 \oplus M_1 = K$$

(c) If the encryption runs for 3 rounds, then the resulting ciphertext is $M_3 M_4$, where $M_3 = M_0$, and $M_4 = M_2 \oplus f(K, M_3) = M_0 \oplus M_1 \oplus K \oplus M_0 \oplus K = M_1$. Therefore, ciphertext is the same as plaintext.

Problem 2: DES Complementation Property

Textbook, problem 4-4.

Solution: Define $C = DES(P, K)$ and $C' = DES(\overline{P}, \overline{K})$. The heart of DES is the Feistel network, whose one stage algorithm is described by (1) and (2).

$$\begin{aligned} L_{i+1} &= R_i && (1) \\ R_{i+1} &= L_i \oplus f(R_i, K_i) && (2) \end{aligned}$$

For $DES(L_0 R_0, K)$, define $L'_0 = \overline{L_0}$, $R'_0 = \overline{R_0}$ and $K'_i = \overline{K_i}$, which leads to another instance $DES(L'_0 R'_0, K')$. We will show that for any stage of the Feistel network, $L'_i = \overline{L_i}$ and $R'_i = \overline{R_i}$.

- Base: the case when $i = 1$.

  For instance $DES(L_0R_0, K)$,

$$\begin{aligned} L_1 &= R_0 && (3) \\ R_1 &= L_0 \oplus f(R_0, K_0) && (4) \end{aligned}$$

  For instance $DES(L'_0R'_0, K')$,

$$\begin{aligned} L'_1 &= R'_0 = \overline{R_0} = \overline{L_0} && (5) \\ R'_1 &= L'_0 \oplus f(R'_0, K'_0) \\ &= \overline{L_0} \oplus f(\overline{R_0}, \overline{K_0}) && (6) \end{aligned}$$

  Since $f(R_i, K_i)$ uses the bitwise $\oplus$ operation to combine input bits of R_i (after expansion) and K_i before the permutation in S-boxes, and $\oplus$ operation is associative and commutative, the following holds for any r and k of the same length,

$$\overline{r} \oplus \overline{k} = r \oplus k \qquad (7)$$

  Combining (6) and (7) gives

$$R'_1 = \overline{L_0 \oplus f(R_0, K_0)} = \overline{R_1} \qquad (8)$$

- Induction: Assume the claim holds for all $i < n$, consider the case when $i = n$.

  For instance $DES(L_0R_0, K)$,

$$\begin{aligned} L_n &= R_{n-1} && (9) \\ R_n &= L_{n-1} \oplus f(R_{n-1}, K_{n-1}) && (10) \end{aligned}$$

  For instance $DES(L'_0R'_0, K')$,

$$\begin{aligned} L'_n &= R'_{n-1} = \overline{R_{n-1}} && (11) \\ R'_n &= L'_{n-1} \oplus f(R'_{n-1}, K'_{n-1}) \\ &= \overline{L_{n-1}} \oplus f(\overline{R_{n-1}}, \overline{K_{n-1}}) \\ &= \overline{L_{n-1} \oplus f(R_{n-1}, K_{n-1})} && (12) \end{aligned}$$

Therefore, after 16 stages of Feistel network, we can get $L'_{16} = \overline{L_{16}}$ and $R'_{16} = \overline{R_{16}}$. Concatenating L'_{16} and R'_{16}, we conclude

$$y' = L'_{16}R'_{16} = \overline{L_{16}R_{16}} = \overline{y} \qquad (13)$$

Problem 3: Triple DES

Textbook, problem 4-6.

Solution: This version of Triple DES is vulnerable to a known plaintext attack, similar to Double DES is. Suppose the attacker Eve knows a plaintext-ciphertext pair (m, c). She first computes $E_K(E_K(m))$ and $D_K(c)$ for all possible K. Then Eve looks for the intersection of these two sets. We know the intersection is not empty since the actual key pair (K_1, K_2) must work. Then if there is only one matching key pair, Eve immediately knows the correct key pair. If there are more than one matching pairs, Eve can use other known plaintext-ciphertext pairs to further reduce the number of candidate key pairs. Essentially, if we define $E'_K() = E_K(E_K())$, then the meet-in-the-middle attack can be carried out in exactly the same way as in Double DES.

Problem 4: Modified Feistel Network

Textbook, problem 4-8.

Solution: We prove the correctness of the decryption function by induction.

- For the base case $i = n$, the claim is true since $A_n = L_n$, $B_n = M_n$, and $C_n = R_n$.
- By the induction hypothesis, assume the claim is true for some i, i.e., $A_i = L_i$, $B_i = M_i$, and $C_i = R_i$.
- In the induction step, we check the case for $i - 1$:

$$\begin{aligned}
A_{i-1} &= B_i \quad \texttt{(decryption function)} \\
&= M_i \quad \texttt{(induction hypothesis)} \\
&= L_{i-1} \quad \texttt{(encryption function)} \\
C_{i-1} &= A_i \quad \texttt{(decryption function)} \\
&= L_i \quad \texttt{(induction hypothesis)} \\
&= R_{i-1} \quad \texttt{(encryption function)} \\
B_{i-1} &= f(K_i, A_i) \oplus C_i \quad \texttt{(decryption function)} \\
&= f(K_i, L_i) \oplus R_i \quad \texttt{(induction hypothesis)} \\
&= f(K_i, R_{i-1}) \oplus f(K_i, R_{i-1}) \oplus M_{i-1} \quad \texttt{(encryption function)} \\
&= M_{i-1}
\end{aligned}$$

Therefore, the claim is true for all i. As a result, $A_0B_0C_0$ is the correct plaintext.

Problem 5: Extended CFB Mode

Textbook, problem 4-9. (Note: "CFB mode" as used in this problem is what we called "Extended CFB mode" in the lectures.)

Solution:

(a) By $\oplus L_{32}(E_K(X_j))$ on both sides of the encryption function, it is straightforward that

$$P_j = C_j \oplus L_{32}(E_K(X_j))$$

We also update X_j in the same way as

$$X_{j+1} = R_{32}(X_j)||C_j$$

(b) We show that the corrupted C_1 will not affect the decryption of blocks P_i for all $i \geq 4$. The reasoning is as follows.

- P_1 cannot be calculated since C_1 is wrong.
- The second half of $X_2 = R_{32}(X_1)||C_1$ is wrong because C_1 is wrong.
- P_2 is wrong since X_2 is wrong.
- The first half of $X_3 = R_{32}(X_2)||C_2$ is wrong because $R_{32}(X_2) = C_1$.
- P_3 is wrong since X_3 is wrong.
- $X_4 = R_{32}(X_3)||C_3 = C_2||C_3$ is correct, because both C_2 and C_3 are correct.
- Thus, P_4 is correct. As all other ciphertexts $C_j \mid j \geq 4$ are correct, all $X_j \mid j \geq 4$ will be correct, and all $P_j \mid j \geq 4$ could be successfully decrypted as a result.

Problem 6: Fast exponentiation algorithm

A recursive algorithm for modular exponentiation was presented in class (Lecture 8, slide 17).

```
/* computes m^e mod n recursively */
int modexp( int m, int e, int n) {
  int r;
  if ( e == 0 ) return 1;          /* m^0 = 1 */
  r = modexp(m*m % n, e/2, n);     /* r = (m^2)^(e/2) mod n */
  if ( (e&1) == 1 ) r = r*m % n;   /* handle case of odd e */
  return r;
}
```

Here is a different recursive algorithm to do the same thing.

```
/* alternate method to compute m^e mod n recursively */
int modexp2( int m, int e, int n) {
  int r;
  if ( e == 0 ) return 1;
  if ( e&1 ) return m*modexp2(m, e-1, n) % n;
  r = modexp2(m, e/2, n);
  return r*r % n;
}
```

Both algorithms operate by computing $m^k \pmod{n}$ for various integers k.

(a) Explain why `modexp2` is correct.

(b) For each algorithm, list the powers of m that are multiplied together during the course of the algorithm when computing $m^{23} \pmod{n}$. For example, if an algorithm computes m^5 by computing $m^2 = m * m$, $m^3 = m^2 * m$, $m^5 = m^3 * m^2$, you would list the exponent pairs $(1, 1)$, $(2, 1)$, and $(3, 2)$.

(c) Some of the multiplications performed by these algorithms are redundant. Which ones in part b are redundant?

(d) Rewrite `modexp2` to make exactly the same useful multiplications but avoid making the redundant ones.

Solution:

(a) We prove the correctness of `modexp2` by induction.

- For the base case $e = 0$, the algorithm is correct since `modexp2`$(m, 0, n) = 1$.
- By induction hypothesis, assume the algorithm is correct for all $e \leq y - 1$, i.e., `modexp2`$(m, e, n) = m^e \bmod n$ for all $e \leq y - 1$.
- In the induction step, consider the case $e = y$. If y is odd,

$$\texttt{modexp2}(m, y, n) \equiv m * \texttt{modexp2}(m, y-1, n) \equiv m * m^{y-1} \equiv m^y \pmod{n}$$

If y is even,

$$\texttt{modexp2}(m, y, n) \equiv (\texttt{modexp2}(m, y/2, n))^2 \equiv (m^{y/2})^2 \bmod n \equiv m^y \pmod{n}$$

In both cases, `modexp2` is correct.

(b)
- For `modexp`, we have $(1,1)$, $(2,2)$, $(4,4)$, $(8,8)$, $\underline{(16,16)}$, $\underline{(0,16)}$, $(16,4)$, $(20,2)$, and $(22,1)$.
- For `modexp2`, we have $\underline{(1,0)}$, $(1,1)$, $(2,2)$, $(1,4)$, $(5,5)$, $(1,10)$, $(11,11)$, and $(1,22)$.

(c) The redundant multiplications are marked by underlines in the previous part.

(d)
```
int modexp2( int m, int e, int n) {
  int r;
  if ( e == 1 ) return m;
  if ( e == 0 ) return 1;
  if ( e&1 ) return m*modexp2(m, e-1, n) % n;
  r = modexp2(m, e/2, n);
  return r*r % n;
}
```