CPSC 418 / MATH 318 — Introduction to Cryptography
ASSIGNMENT 2

Name: Daniel Heyns (Crypto Wizard)
Student ID: 30021292

Problem 1 — Arithmetic in the AES MixColumns operation

a. (i) Let $a(y) = b_1y^3 + b_2y^2 + b_3y + b_4$ be the generic 4-byte vector.

Now,

$$\begin{aligned} y * a(y) &= y(b_1y^3 + b_2y^2 + b_3y + b_4) \\ &= b_1y^4 + b_2y^3 + b_3y^2 + b_4y \end{aligned}$$

Since $y^4 = 1$,

$$\begin{aligned} y * a(y) &= b_1 + b_2y^3 + b_3y^2 + b_4y \\ &= b_2y^3 + b_3y^2 + b_4y + b_1 \end{aligned}$$

We see the coefficients have shifted left with the MSF becoming the y^0 coefficient. Thus circular left shit. This is clearly represented with the byte vector representation of $y * a(y)$, (b2, b3, b4, b1).

(ii) Let $a(y) = b_1y^3 + b_2y^2 + b_3y + b_4$ be the generic 4-byte vector.

$$y^i * a(y) = b_1y^{3+i} + b_2y^{2+i} + b_3y^{1+i} + b_4y^i \tag{1}$$

Because $0 \leq i \leq 3$ is such a small bound, we may tackle each value as a case.

<u>$i = 0$</u>

$$\begin{aligned} y^0 * a(y) &= b_1y^{3+0} + b_2y^{2+0} + b_3y^{1+0} + b_4y^0 \\ &= b_1y^3 + b_2y^2 + b_3y + b_4 \end{aligned}$$

Resulting in the byte vector (b1, b2, b3, b4). Left shift of 0 bytes.

<u>$i = 1$</u>

Proven in (i).

<u>$i = 2$</u>

$$\begin{aligned} y^2 * a(y) &= b_1y^{3+2} + b_2y^{2+2} + b_3y^{1+2} + b_4y^2 \\ &= b_1y^5 + b_2y^4 + b_3y^3 + b_4y^2 \end{aligned}$$

Since $y^4 = 1$ and $y^5 = y$,

$$y^2 * a(y) = b_3y^3 + b_4y^2 + b_1y + b_2$$

Resulting in the byte vector (b3, b4, b2, b1). Left shift of 2 bytes.

$\underline{i = 3}$

$$\begin{aligned} y^3 * a(y) &= b_1 y^{3+3} + b_2 y^{2+3} + b_3 y^{1+3} + b_4 y^3 \\ &= b_1 y^6 + b_2 y^5 + b_3 y^4 + b_4 y^3 \end{aligned}$$

Since $y^4 = 1$, $y^5 = y$ and $y^6 = y^2$,

$$y^3 * a(y) = b_4 y^3 + b_1 y^2 + b_2 y + b_3$$

Resulting in the byte vector (b4, b1, b2, b3). Left shift of 3 bytes.

With all cases of i covered, we can say that multiplication of any 4-byte vector by $0 \leq i \leq 3$ is a circular left shift of the vector by i bytes.

b. (i) $c_1(x) = 1, c_2(x) = x, c_3(x) = x + 1$

(ii) We start with the generic bit polynomial of the form,

$$b(x) = b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0$$

Now,

$$\begin{aligned} d(x) &= b(x) * c_2(x) \\ &= b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x \end{aligned}$$

In GF(2^8)

$$\begin{aligned} m(x) &= x^8 + x^4 + x^3 + x + 1 = 0 \\ x^8 &= x^4 + x^3 + x + 1 \end{aligned}$$

So,

$$d(x) = b_7 x^4 + b_7 x^3 + b_7 x + b_7 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$$

Collect like terms and we end up with,

$$\begin{aligned} d_7 &= b_6 \\ d_6 &= b_5 \\ d_5 &= b_4 \\ d_4 &= b_7 + b_3 \\ d_3 &= b_7 + b_2 \\ d_2 &= b_1 \\ d_1 &= b_7 + b_0 \\ d_0 &= b_7 \end{aligned}$$

(iii) We start with the generic bit polynomial of the form,

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

Now,

$$d(x) = b(x) * c_3(x)$$

$$= b_7x^8+b_6x^7+b_5x^6+b_4x^5+b_3x^4+b_2x^3+b_1x^2+b_0x+b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0$$

In GF(2^8)

$$\begin{aligned} m(x) &= x^8 + x^4 + x^3 + x + 1 = 0 \\ x^8 &= x^4 + x^3 + x + 1 \end{aligned}$$

So,

$$d(x) = b_7x^4 + b_7x^3 + b_7x + b_7 + b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x + b_7x^7 + b_6x^6 \\ + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Collect like terms and we end up with,

$$\begin{aligned} d_7 &= b_7 + b_6 \\ d_6 &= b_6 + b_5 \\ d_5 &= b_5 + b_4 \\ d_4 &= b_7 + b_4 + b_3 \\ d_3 &= b_7 + b_3 + b_2 \\ d_2 &= b_2 + b_1 \\ d_1 &= b_7 + b_1 + b_0 \\ d_0 &= b_7 + b_0 \end{aligned}$$

c. (i) We begin with the generic 4 byte polynomial,

$$\begin{aligned} s(y) =& s_3y^3 + s_2y^2 + s_1y + s_0 \\ c(y) =& (03)y^3 + (01)y^2 + (01)y + (02) \\ t(y) =& s(y) * c(y) \\ =& (03)s_3y^6 + (01)s_3y^5 + (01)s_3y^4 + (02)s_3y^3 + \\ & (03)s_2y^5 + (01)s_2y^4 + (01)s_2y^3 + (02)s_2y^2 + \\ & (03)s_1y^4 + (01)s_1y^3 + (01)s_1y^2 + (02)s_1y^1 + \\ & (03)s_0y^3 + (01)s_0y^2 + (01)s_0y^1 + (02)s_0 \end{aligned}$$

Collect y's,

$$t(y) = (03)s_3y^6 + ((01)s_3 + (03)s_2)y^5 + ((01)s_3 + (01)s_2 + (03)s_1)y^4 + ((02)s_3 + (01)s_2 + (01)s_1 + \\ (03)s_0)y^3 + ((02)s_2 + (01)s_1 + (01)s_0)y^2 + ((02)s_1 + (01)s_0)y + (02)s_0$$

Since $y^4 = 1, y^5 = y, y^6 = y^2$ we can replace terms and output our final formulas,

$$\begin{aligned}
t_0 &= (01)s_3 + (01)s_2 + (03)s_1 + (02)s_0 \\
t_1 &= (01)s_3 + (03)s_2 + (02)s_1 + (01)s_0 \\
t_2 &= (03)s_3 + (02)s_2 + (01)s_1 + (01)s_0 \\
t_3 &= (02)s_3 + (01)s_2 + (01)s_1 + (03)s_0
\end{aligned}$$

(ii)

$$C = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

That is,

$$\begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

Problem 2 — Error propagation in block cipher modes

a. (i) Since ECB encrypts/decrypts all blocks individually, only M_i would be affected.

   (ii) During decryption, the decrypted C_{i+1} would be XOR'ed with C_i in $D_k(C_{i+1}) \oplus C_i = M_{i+1}$. Therefore M_i and M_{i+1} would be affected.

   (iii) In OFB only M_i would be affected. Each block is XOR'ed with an i encrypted IV. The blocks do not interact with one another.

   (iv) During decryption, M_{i+1} is calculated using the encryption of C_i. $M_{i+1} = E_k(C_i) \oplus C_{i+1}$. Therefore M_i and M_{i+1} would be affected.

   (v) Only M_i will be affected. No connection between block encryption or decryption. $C_i = M_i \oplus E_k(ctr)$.

b. Only M_i will be affected It's as if a different message is being encrypted. Each block will be encrypted then decrypted normally, leaving a corrupted M_i.

Problem 3 — Binary exponentiation

a.

$$\begin{aligned} k = \lfloor log_2(n) \rfloor &= \lfloor log_2(11) \rfloor = 3 \\ n = 1 * 2^3 + 0 * 2^2 &+ 1 * 2^1 + 1 * 2^0 \\ r_0 &\equiv a \, (mod \, m) \\ r_0 &\equiv 17 \, (mod \, 77) \\ r_1 = r_0^2 \, (mod \, m) &= 17^2 \, (mod \, 77) = 58 \\ r_3 = r_1^2 \, (mod \, m) &= 58^2 \, (mod \, 77) = 54 \\ r_3 = r_2^2 \, (mod \, m) &= 14^2 \, (mod \, 77) = 61 \end{aligned}$$

b. (i) Base case

Let $i = 0$ that is,

$$s_0 = \sum_{j=0}^{i} b_j * 2^{i-j} = \sum_{j=0}^{0} b_j * 2^{0-j} = 1 * 2^{0-0} = 1$$

Induction proof

Assume,

$$s_i = \sum_{j=0}^{i} b_j * 2^{i-j} \quad \text{for } 0 \le i \le k. \tag{2}$$

$$\begin{aligned} s_{i+1} = 2s_i + b_{i+1} = 2 &\left(\sum_{j=0}^{i} b_j 2^{i-j} \right) + b_{i+1} \quad \text{from (2)} \\ = &\left(\sum_{j=0}^{i} 2b_j 2^{i-j} \right) + b_{i+1} * 2^0 \\ = &b_0 2^{i+1-0} + b_1 2^{i+1-1} + ... + b_i 2^{i+1-i} + b_{i+1} 2^{i+1-(i+1)} \\ = &\left(\sum_{j=0}^{i+1} b_j 2^{i-j} \right) \end{aligned}$$

(ii) Base case

$i = 0$ and $s_0 = 1$

$$r_0 = a \, (mod \, m) = a^1 \, (mod \, m) = a^{s_0} \, (mod \, m)$$

Induction proof

Assume,

$$r_i \equiv a^{s_i} \, (mod \, m) \quad \text{for } 0 \le i \le k. \tag{3}$$

Case 1

$$
\begin{aligned}
&r_{i+1} \equiv r_i^2 \ (mod\, m) \quad \text{if } b_{i+1} = 0 \\
&r_{i+1} \equiv (a^{s_i})^2 \ (mod\, m) \quad \text{by (3)} \\
&r_{i+1} \equiv a^{2s_i} \ (mod\, m) \\
&r_{i+1} \equiv a^{2s_i + b_{i+1}} \ (mod\, m) \quad \text{since } b_{i+1} = 0 \\
&r_{i+1} \equiv a^{s_{i+1}} \ (mod\, m)
\end{aligned}
$$

Case 2

$$
\begin{aligned}
&r_{i+1} \equiv r_i^2 a \ (mod\, m) \quad \text{if } b_{i+1} = 1 \\
&r_{i+1} \equiv a^{2s_i + 1} \ (mod\, m) \quad \text{by (3)} \\
&r_{i+1} \equiv a^{2s_i + b_{i+1}} \ (mod\, m) \quad \text{since } b_{i+1} = 1 \\
&r_{i+1} \equiv a^{s_{i+1}} \ (mod\, m)
\end{aligned}
$$

(iii) Previously shown,

$$r_k \equiv a^{s_k} \ (mod\, m) \tag{4}$$

$$s_k = \sum_{j=0}^{k} b_j * 2^{k-j} \tag{5}$$

$$n = \sum_{j=0}^{k} b_j * 2^{k-j} \tag{6}$$

Now,

$$
\begin{aligned}
&r_k \equiv a^{s_k} \ (mod\, m) \quad \text{by (4)} \\
&r_k \equiv a^{\sum_{j=0}^{k} b_j * 2^{k-j}} \ (mod\, m) \quad \text{by (5)} \\
&r_k \equiv a^n \ (mod\, m) \quad \text{by (6)} \\
&a^n \equiv r_k \ (mod\, m)
\end{aligned}
$$

Problem 4 — A modified man-in-the-middle attack on Diffie-Hellman

a. We will start with Bob.
Bob receives

$$(g^a)^q = g^{aq}$$

Bob raises this to his own b value...

$$(g^{aq})^b = g^{aqb} = K_b$$

Now Alice,
Alice receives

$$(g^b)^q = g^{bq}$$

Alice raises this to his own a value...

$$(g^{bq})^a = g^{bqa} = K_a$$

We see

$$K_a = g^{bqa} = g^{aqb} = K_b$$

b. We start with the following,

$$K \equiv g^{abq}\ (mod\ P)$$
$$\text{let } t = ab$$
$$K \equiv g^{tq}\ (mod\ P)$$

Now,

$$q = (P-1)/m$$
$$P - 1 = mq \tag{7}$$
$$tq = s(P-1) + r \quad \text{where } s \text{ and } r \text{ are some integers.} \tag{8}$$

$$K \equiv g^{s(P-1)} * g^r\ (mod\ P)$$
$$\text{FLT: } g^(P-1) \equiv 1\ (mod\ P)$$
$$K \equiv (g^{(P-1)})^s * g^r\ (mod\ P)$$
$$K \equiv g^r\ (mod\ P)$$

From (7) and (8)

$$tq = smq + r$$
$$r = tq - smq$$
$$r = q(t - sm)$$

We can see that r is a multiple of q as $(t - sm) \epsilon \mathbb{Z}$
Now,
$q = (P-1)/m$ we see that the possibilities for r increases as m increases. This is evident as

m splits $P-1$ into smaller pieces that r will be a multiple of. Thus $0 \leq r \leq P-2$ has more potential values directly proportional to the size of m.
$K \equiv g^r \,(mod\, P)$, Mallory knows P, m and g. She can plug in n values into $r = n * (P-1)/m$ and keep the bound of r in mind, then insert r into the K equation. This will allow her to calculate all possible keys.

c. This is useful because Mallory doesn't have to do anything after Bob and Alice have the key K and once Mallory knows the key as well. She can simply tap the line and decrypt the messages later, without needing to act as a man in the middle, decrypting and encrypting messages to send to each communicator.

Problem 5 — A simplified password-based key agreement protocol

a. We know,

$$B \equiv g^b \ (mod\, N)$$
$$A \equiv g^a \ (mod\, N)$$
$$v \equiv g^p \ (mod\, N)$$

Therefore,

$$K_{client} \equiv B^{a+p} \equiv B^a * B^p \equiv g^{ba} * g^{bp} \equiv A^b * v^b \equiv (Av)^b \equiv K_{server} \ (mod\, N)$$

b. Mallory could calculate an A value such that $Av \equiv 1 \ (mod\, N)$ and $A \,\epsilon\, \mathbb{Z}$. She sends (I,A) to the server. Mallory recieves B from the server and uses 1 as the key. This works because...

$$K_{server} \equiv (Av)^b \ (mod\, N)$$
$$K_{server} \equiv 1^b \ (mod\, N)$$
$$K_{server} \equiv 1 \ (mod\, N)$$

c. Assume Mallory can solve the Key recovery problem. That is Mallory can solve $K = (Av)^b$ for K, when exponents a and b are unknown to her. If Mallory can solve the key recovery problem for any v, then we assume she can solve this problem when $v = 1$.
That is Mallory can solve $K = (A * 1)^b$...

$$K = (A * 1)^b$$
$$K = A^b$$

$K = A^b$ is the Diffie-Hellman problem, therefore if Mallory can solve the key recovery problem without knowing a or b, then she can solve the Diffie-Hellman problem as well.