CPSC 418 / MATH 318 — Introduction to Cryptography
ASSIGNMENT 2

Name: Jeremy Stuart
Student ID: 00311644

Problem 1 — Arithmetic in the AES MixColumns operation (22 marks)

(a) i. Suppose x is an arbitrary 4-byte vector, and $x = y^3b_3 + y^2b_2 + yb_1 + b_0$. We must show that $y \cdot x$ is equivalent to a circular shift left which is to show that $y \cdot x = (y^3b_2 + y^2b_1 + yb_0 + b_3$.

$$\begin{aligned} y \cdot x &= y \cdot (y^3b_3 + y^2b_2 + yb_1 + b_0) \\ &= y^4b_3 + y^3b_2 + y^2b_1 + yb_0 \end{aligned}$$

Since $M(y) = y^4 + 1$ then $M(y) = 0$ means $y^4 = 1$
Then:

$$\begin{aligned} &= y^4b_3 + y^3b_2 + y^2b_1 + yb_0 \\ &= 1b_3 + y^3b_2 + y^2b_1 + yb_0 \\ &= y^3b_2 + y^2b_1 + yb_0 + 1b_3 \\ &= y^3b_2 + y^2b_1 + yb_0 + b_3 \end{aligned}$$

This is equivalent to the circular left shift we desire, as the MSB has shifted to the LSB position and all other bytes have shifted left. Thus multiplication of any 4-byte vector by y is equivalent to a circular left shift.

ii. To prove that $y^i \equiv y^j$ for any integer $i \geq 0$ we first recognize that doing arithmetic in which the polynomial $M(y) = y^4 + 1$ means that $y^4 = 1$. This means that any exponent of y will be reduced $\mod 4$. We can then rewrite i as $4x + j$ where x is an integer, and j is an integer such that $0 \leq r \leq 3$. Then there are four cases:
Case 1: r = 0
Then $i = 4x$, and $j \equiv 4x + r \mod 4$ which means $j = 0$. This is equivalent to $y^i = y^{4x}$ and since $4 \equiv 0 \mod 4$ then $y^i = y^j$ where $j \equiv i \mod 4$ and $j = 0$.

Case 2: r = 1
Then $i = 4x + 1$, and $j \equiv 4x + r \mod 4$ which means $j = 1$. This is equivalent to $y^i = y^{4x+r}$ and since $4x + 1 \equiv 1 \mod 4$ then $y^i = y^j$ where $j \equiv i \mod 4$ and $j = 1$.

Case 3: r = 2
Then $i = 4x + 2$, and $j \equiv 4x + r \mod 4$ which means $j = 2$. This is equivalent to $y^i = y^{4x+r}$ and since $4x + 2 \equiv 2 \mod 4$ then $y^i = y^j$ where $j \equiv i \mod 4$ and $j = 2$.

Case 4: r = 3

Then $i = 4x + 3$, and $j \equiv 4x + r \mod 4$ which means $j = 3$. This is equivalent to $y^i = y^{4x+r}$ and since $4x + 3 \equiv 3 \mod 4$ then $y^i = y^j$ where $j \equiv i \mod 4$ and $j = 3$. Then $y^i \equiv y^j$ for any integer $i \geq 0$ where $0 \leq j \leq 3$.

iii. To prove that multiplication of any 4-byte vector by $y^i (i \geq 0)$ is a circular left shift by j bytes where $j \equiv i(\mod 4)$ with $0 \leq j \leq 3$ we can use our previous questions to show how the modulo operation reduces the exponent to keep the expression as 4-bytes. We first recognize that any exponent of y can be expressed as $4x + j$, and since $4X$ always reduces to 0 mod 4, and we know that $0 \leq j \leq 3$ then the result of multiplying any 4-bytes of the form $y^3b_3 + y^2b_2 + yb_1 + b_0$ must result in an expression where the highest exponent is 3. We then use our proof for part i, which demonstrates that multiplication by y is a circular left shift of 1 place. Since j is equivalent to r in the last question, we know that y^j can be broken into $y \cdot y^{j-1}$ and this can expand out j times. Each of these multiplications is then a single left shift, which taken together form j circular left shifts.

(b) i. (03) in hex is equivalent to 0000 0011 in binary. This translates to $x^1 + 1$ in the Rijndael field $GF(2^8)$.
(02) in hex is equivalent to 0000 0010 in binary. This translates to x^1 in the Rijndael field $GF(2^8)$.
(01) in hex is equivalent to 0000 0001 in binary. This translates to 1 in the Rijndael field $GF(2^8)$.

ii. From our previous answer we know that (02) is x^1 (or just x) in the Rijndael field $GF(2^8)$.
Then:

$$\begin{aligned} d &= (02)b \\ &= (x)(b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0) \\ &= (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \end{aligned}$$

But since we're working in modulo $m(x) = x^8 + x^4 + x^3 + x + 1$ then we also know that $x^8 = x^4 + x^3 + x + 1$. We substitute this in to get that:

$$\begin{aligned} &= (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \\ &= (b_7(x^4 + x^3 + x + 1) + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x \\ &= b_7x^4 + b_7x^3 + b_7x + b_7 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x \\ d_i &= b_6x^7 + b_5x^6 + b_4x^5 + (b_7 \oplus b_3)x^4 + (b_7 \oplus b_2)x^3 + b_1x^2 + (b_7 \oplus b_0)x + b_7 \end{aligned}$$

iii. From part i we know that (03) is $x + 1$ in the Rijndael field $GF(2^8)$. This gives us the equation:

$$\begin{aligned} d &= (03)b \\ &= (x + 1)(b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0) \end{aligned}$$

But we recognize that in this expansion, we start by multiplying $x \cdot b$ and then add $1 \cdot b$. In part ii (above) we arrived at the answer for $x \cdot b$ which is d_i, so we start

by taking that answer and adding the result of $1 \cdot b$ (which is trivially just b). We arrive at:

$$\begin{aligned}
e =& d_i + b \\
=& (b_6x^7 + b_5x^6 + b_4x^5 + (b_7 \oplus b_3)x^4 + (b_7 \oplus b_2)x^3 + b_1x^2 + (b_7 \oplus b_0)x + b_7) \\
&+ (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0) \\
=& (b_6 \oplus b_7)x^7 + (b_5 \oplus b_6)x^6 + (b_4 \oplus b_5)x^5 + ((b_5 \oplus b_6) \oplus b_4)x^4 + ((b_5 \oplus b_6) \oplus b_3)x^3 \\
&+ (b_1 \oplus b_2)x^2 + ((b_5 \oplus b_6) \oplus b_1)x + (b_7 \oplus b_0)
\end{aligned}$$

(c) i. Evaluate $t(y) = s(y) \cdot c(y)(\mod y^4 + 1)$.
$s(y) = s_3y^3 + s_2y^2 + s_1y^1 + s_0$
$c(y) = (03)y_3 + (01)y_2 + (01)y + (02)$

$$\begin{aligned}
t(y) =& s(y) \cdot c(y) \\
=& (s_3y^3 + s_2y^2 + s_1y^1 + s_0) \cdot ((03)y_3 + (01)y_2 + (01)y + (02)) \\
=& (s_3)(03)y^6 + (s_3)(01)y^5 + (s_3)(01)y^4 + (s_3)(02)y^3 + (s_2)(03)y^5 + (s_2)(01)y^4 + (s_2)(01)y^3 \\
&+ (s_2)(02)y^2 + (s_1)(03)y^4 + (s_1)(01)y^3 + (s_1)(01)y^2 + (s_1)(02)y + (s_0)(03)y^3 + (s_0)(01)y^2 \\
&+ (s_0)(01)y + (s_0)(02)
\end{aligned}$$

We now reduce by $\mod y^4 + 1$ meaning every instance of $y^4 = 1$.

$$\begin{aligned}
=& (s_3)(03)y^2 + (s_3)(01)y + (s_3)(01) + (s_3)(02)y^3 + (s_2)(03)y + (s_2)(01) + (s_2)(01)y^3 + (s_2)(02)y^2 \\
&+ (s_1)(03) + (s_1)(01)y^3 + (s_1)(01)y^2 + (s_1)(02)y + (s_0)(03)y^3 + (s_0)(01)y^2 + (s_0)(01)y \\
&+ (s_0)(02)
\end{aligned}$$

We now group like terms and XOR the terms appropriately:

$$\begin{aligned}
t(y) =& ((((s_3)(01) \oplus (s_2)(01)) \oplus (s_1)(03)) \oplus (s_0)(02)) \\
&+ ((((s_3)(01) \oplus (s_2)(03)) \oplus (s_1)(02)) \oplus (s_0)(01))y \\
&+ ((((s_3)(03) \oplus (s_2)(02)) \oplus (s_1)(01)) \oplus (s_0)(01))y^2 \\
&+ ((((s_3)(02) \oplus (s_2)(01)) \oplus (s_1)(01)) \oplus (s_0)(03))y^3
\end{aligned}$$

ii. The matrix C is:

$$\begin{pmatrix} t0 \\ t1 \\ t2 \\ t3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s0 \\ s1 \\ s2 \\ s3 \end{pmatrix} \tag{1}$$

Problem 2 — Error propagation in block cipher modes (12 marks)

(a) i. **ECB Mode**: Only the block with the error in it is affected, since ECB simply XOR's each block in the message with the key (essentially just a substitution cipher), when the decrypt process takes place the cipher block with the error will be XOR'd with the key and return something other than the original message block. Every other block is decyrpted independent of the cipher block with the error however, so nothing else in the message is affected.

ii. **CBC Mode**: In CBC mode, the corrupted cipher block C_i will cause errors in message block M_i, and M_{i+1}. This is because decryption in CBC mode is done by XORing the the current cipher block and the previous block to obtain the current message block . So M_i is obtained by XORing C_i and C_{i-1}, and M_{i+1} is obtained by XORing C_{i+1} and C_i. In both bases we know that C_i is corrupted, so it will not produce the correct message blocks when XORing to obtain M_i and M_{i+1}.

iii. **OFB Mode**: In OFB mode, keystream blocks are generated initially using the key and some IV. The result of this becomes the input for the next block of the keystream, but is also XORed with the message to generate the ciphertext of the current block. When decrypting, the same protocol is carried out except that now the ciphertext is XORed with the keystream to generate the message. Since the ciphertext plays no part in the generation of the keystream then the only message block affected by our corrupted cipherblock C_i is the corresponding plaintext block M_i.

iv. **CFB mode with one register**: In CFB mode our keystream is generated from a register that holds the previous k ciphertexts. This means that our corrupted C_i block will be used to generate the keystream until it is completely cycled through the register, which should take k encryption steps. This means that K message blocks following the corrupted block will also not be properly decrypted to their appropriate messages, but the error will not propagate any further. If only one ciphertext is held in the register, then only M_i and M_{i+1} will be affected since C_i is used in the decryption of those two blocks.

v. **CTR Mode**: CTR mode is essentially the same as CFB mode, except a counter is used instead of a register. Once they are combined/encrypted to generate the keystream the same process of XORing is followed as CFB mode, however the resulting cipher is not fed back into the keystream of the next block, so the corrupted block C_i will only affect the resulting message block M_i, and the rest of the message maintains it's integrity.

(b) None of the other message blocks would be affected, only the originally corrupted message block M_i. While it might seem like the corrupted message block M_i would corrupt every block after it, it in fact would not. We know that the resulting cipher block C_i would not decrypt to what Alice intended M_1 to be. Intuition would say that because C_i is used in the encryption of M_{i+1} that it would spread, except that C_i could be ANY block or information and that so long as it was used in the same way to decrypt M_{i+1} then it doesn't matter what it is. Every encrypted block following M_i will be different than it would have been if M_i was not corrupted, but the chain of encryption and decryption would still maintain integrity so that the message will remain the same with only M_i being affected.

Problem 3 — Binary exponentiation (13 marks)

(a) To start, calculate $n = 11$ and find it's binary representation: 1011. These bits will correspond to b_0, b_1, b_2, b_3 respectively. We also need to calculate $k = \lfloor \log_2 n \rfloor = 3$. We now follow step 2 and initialize $r_0 \equiv 17 \mod 77$. Following step 3, we find that since $b_{0+1} = 0$ then:

$$\begin{aligned} r_0 + 1 \equiv r_i^2 = r_0^2 &= 17^2 \mod 77 \\ &\equiv 289 \mod 77 \\ &\equiv 58 \mod 77 \\ r_1 &\equiv 58 \end{aligned}$$

Continuing, we increment i to 1 and repeat. Thus since $b_2 = 1$ then $r_{i+1}^2 = r_1^2 \cdot a$, then:

$$\begin{aligned} r_{1+1} \equiv r_i^2 \cdot a = r_1^2 \cdot a &= 58^2 \cdot 17 \mod 77 \\ &\equiv 54 \mod 77 \\ r_2 &\equiv 54 \end{aligned}$$

Continuing, we increment i to 1 and repeat. Thus since $b_3 = 1$ then $r_{i+1}^2 = r_2^2 \cdot a$, then:

$$\begin{aligned} r_{2+1} \equiv r_i^2 \cdot a = r_2^2 \cdot a &= 54^2 \cdot 17 \mod 77 \\ &\equiv 61 \mod 77 \\ r_3 &\equiv 61 \end{aligned}$$

Thus $61 \equiv 17^{11} \mod 77$ by binary exponentiation.

(b) i. Proof by induction

Base Case:

$$\begin{aligned} s_0 &= \sum_{j=0}^{0} b_0 2^{0-0} \\ &= 1 \\ s_0 &= 1 \quad \text{as required} \end{aligned}$$

Will show that:

$$S_{i+1} = \sum_{j=0}^{i+1} b_j 2^{(i+1-j)}$$

Inductive Hypothesis: Assume for all $k \in \mathbb{Z}, k \geq 0$ and assume

$$\sum_{j=0}^{k} b_j 2^{(k-j)} = s_k$$

.

Inductive Step: show for s_{k+1}

$$\begin{aligned} s_{k+1} &= 2(s_k) + b_{k+1} \\ &= 2(\sum_{j=0}^{k} b_j 2^{(k-j)}) + b_{k+1} \quad \text{by inductive hypothesis} \\ &= (\sum_{j=0}^{k} b_j 2^{(k-j+1)}) + b_{k+1} \\ &= (\sum_{j=0}^{k} b_j 2^{(k-j+1)}) + b_{k+1}(2^0) \\ &= (\sum_{j=0}^{k+1} b_j 2^{(k-j+1)}) \end{aligned}$$

By induction,

$$S_{i+1} = \sum_{j=0}^{i+1} b_j 2^{(i+1-j)}$$

ii. Proof by induction:
Base Case: $i = 0$

$$\begin{aligned} & r_0 \equiv a^{s_0} \mod m \quad (s_0 = 1 \quad \text{is given}) \\ \text{then} \quad & r_0 \equiv a^1 \mod m \quad \text{as required in step 2 (initialization)} \end{aligned}$$

Will show that: $r_{i+1} = a^{s_{i+1}} \mod m$

Induction Hypothesis: Assume for all $k \in \mathbb{Z}, k \geq 0$ and assume $r_k \equiv a^{s_k} \mod m$

Induction Step: Show for $r_{k+1} = \begin{cases} \mathrm{r}_i^2 \mod m & \text{if} \quad b_{i+1} = 0 \\ r_i^2 \cdot a \mod m & \text{if} \quad b_{i+1} = 0 \end{cases}$

2 cases:

case $1 : b_{i+1} = 0$

$$\begin{aligned} r_{k+1} &= r_k^2 \\ &= a^{2(s_k)} \quad \text{by induction hypothesis} \\ r_{k+1} &= a^{S_{k+1}} \quad \text{as required} \end{aligned}$$

case $2 : b_{i+1} = 1$

$$\begin{aligned} r_{k+1} &= r_k^2 \cdot a \\ &= r_k \cdot r_k \cdot a \quad \text{by induction hypothesis} \\ &= a^{s_k} \cdot a^{s_k} \cdot a \\ &= a^{2s_k} \cdot a \\ &= a^{2s_k+1} \\ &= a^{2s_k+b_{1+1}} \\ &= a^{s_{k+1}} \quad \text{as required (question specifies } a^{s_{k+1}} = 2s_{k+1} + b_{1+1}) \end{aligned}$$

Then $r_{i+1} = a^s_{i+1} \mod m$ for all $0 \leq 1 \leq k$.

iii. From part ii we know that $r_i \equiv a^{s_i} \mod m$ for $0 \leq i \leq k$. From part i we know that s_i is the sum which expresses the value of n using it's binary form. We therefore know that $s_i = n$. This means that:

$$\begin{aligned} a^n &\equiv r_k \mod m \\ a^n &\equiv a^{s_i} \mod ma^{s_i} \qquad \equiv a^{s_i} \mod m \end{aligned}$$

Therefore $a^n \equiv r_k \mod m$.

Problem 4 — A modified man-in-the-middle attack on Diffie-Hellman (10 marks)

(a) Once Mallory has added her own values during the exchange, Alice receives $(g^b)^q$ and Bob receives $(g^a)^q$. Alice and Bob then raise the numbers they received to their own values a and b, respectively. Alice arrives at $(((g^b)^q)^a)$ and Bob arrives at $(((g^a)^q)^b)$. We find that:

$$\begin{aligned}(((g^b)^q)^a) &= (((g^a)^q)^b) \\ (g^{bq})^a &= (g^{aq})^b \\ g^{bqa} &= g^{aqb} \\ g^{abq} &= g^{abq}\end{aligned}$$

Therefore both Alice and Bob arrive at the same values for their shared key.

(b) We know that the key generated by Alice and Bob is $K = (g^q)^{ab}$. We also know that $q = \frac{p-1}{m}$ which means that $mq = p - 1$. We can think of ab as being some variable that could be equivalent to m. If we do this, we would find that:

$$\begin{aligned}K &\equiv (g^q)^{ab} \mod p \\ &\equiv (g^q)^m \mod p \\ &\equiv g^{mq} \mod p \\ &\equiv q^{p-1} \mod p \qquad \text{(by the properties provided above)} \\ &\equiv 1\end{aligned}$$

It is possible for us to do this because we know that ab is a large number and m is very small (as provided in the problem description). This means we can know that as long as $ab \geq m$ then $ab = xm + r$ where x is some integer (a quotient) and r is a remainder. We also know that $0 \leq r < m$ which tells us that there are m possible values for r. Then we know that there are m possible values for K.

(c) The advantage of this attack where Mallory multiplies by an additional chosen exponent to the values Bob and Alice send (hereafter "value added attack") over Mallory's attack where she suppresses Alice and Bob's values and replaces them with her own (hereafter "suppression attack") is that in the value added attack Mallory can become a passive attacker, while she needs to remain active in the suppression attack. This is because in the suppression attack, both Alice and Bob will receive g^e from Mallory. Alice will then compute g^{ea} but thinks it is g^{ba}, while Bob will calculate g^{eb} thinking it is actually g^{ab}. If the protocol were successful, both parties would have the same value, but since Mallory has suppressed their values then g^{ea} should not equal g^{ea}. When Alice sends a message to Bob it will be encrypted with g^{ea} and Bob would try to decrypt it with g^{eb} but since we know these values are not equal he will quickly discover something is wrong. To stop this, Mallory would have to intercept Alice's message, decrypt it with $g^e a$ and encrypt it again with g^{eb} so that Bob would be able to decrypt the message without discovering anything is wrong. Mallory would need to do this for EVERY message exchanged by Alice and Bob and if she commits and mistakes with decrypting and encrypting, or takes too long, would risk Alice and Bob discovering her attack. Using the value added attack, Mallory can simply read every encrypted message that is sent back and forth because

she can decrypt the messages based on the proof we provided in part ii of this question. In this value added attack Mallory is not required to do anything after adding her own exponent, and so essentially becomes a passive attacker. The advantage of the value added attack over the suppression attack is Mallory is not required to compute anything extra, and Alice and Bob can exchange messages without Mallory in the middle. Mallory can stop monitoring without Alice and Bob ever finding out, and can return again to keep listening at any time.

Problem 5 — A simplified password-based key agreement protocol (8 marks)

(a) Show $K_{client} = K_{server}$:

$$\begin{aligned} K_{client} &= K_{server} \\ B^{a+p} &= (Av)^b \\ (g^b)^{a+p} &= (g^a g^p)^b \\ (g^{ab+bp}) &= (g^{a+p})^b \\ (g^{ab+bp}) &= (g^{ab+bp}) \end{aligned}$$

Thus both arrive at the same key.

(b) In order to be able to generate a valid K_{client} Mallory needs to send a value A such that $A = v^{-1}$, that is to say that A is a multiplicative inverse of v. This means that when A and v are multiplied together, they would be equivalent to 1 ($\mod p$), and the result would be that:

$$\begin{aligned} K_{server} &= (Av)^b \mod N \\ &= (v^{-1}v)^b \mod N \\ &= (1)^b \\ &= 1 \end{aligned}$$

Mallory can easily find this value for A since we know that v and N are prime, and we therefore know that a unique inverse exists. Mallory knows the value of v and N and can therefore calculate the inverse. This means that Mallory can select A as the multiplicative inverse modulo N, and this will result in a key that is equivalent to 1 modulo N. This means that the Server will calculate a key value equivalent to 1, which Mallory knows and so she needs only send that value back to prove that she knows the value of B^{a+p} without knowing the password.

(c) We begin by assuming that Mallory can solve any instance of the key recovery problem. In order to do this, Mallory needs to be able to compute discrete logs. This is because K_{server} and K_{client} both calculate to g^{ab+bp} which is shown in part i of this question. If we assume that Mallory can find any key K for any values of a, b, v then we need to assume that Mallory has found some way to extract the exponent of g (thereby solving the discrete log problem). This would mean Mallory can extract the exponents for g^{ab}. If Mallory is able to extract this, then she can also extract the same value for K from the Diffie-Hellman problem since $K = g^{ab}$.