CPSC 418 / MATH 318 — Introduction to Cryptography
ASSIGNMENT 2 Problems 1-5

Name: Aifaz Dhuka
Student ID: 30069823

Problem 1 — Arithmetic in the AES MixColumns operation

(a) i. Consider a 4-byte vector $[b_3\ b_2\ b_1\ b_0]$. We can write this as the following polynomial- $p(y) = b_3y^3 + b_2y^2 + b_1y^1 + b_0y^0 = b_3y^3 + b_2y^2 + b_1y^1 + b_0$. So multiplying the polynomial by y results in the following:

$$\begin{aligned} p(y) * y &= (b_3y^3 + b_2y^2 + b_1y^1 + b_0)y \\ &= (b_3y^4 + b_2y^3 + b_1y^2 + b_0y^1) \\ \text{Since we know } y^4 = 1& \\ &= b_3 + b_2y^3 + b_1y^2 + b_0y^1 \end{aligned}$$

So the new vector after multiplication is $[b_2\ b_1\ b_0\ b_3]$, which is the left circular shift from the original vector.

ii. Consider a 4-byte vector $[b_3\ b_2\ b_1\ b_0]$. We can write this as the following polynomial- $p(y) = b_3y^3 + b_2y^2 + b_1y^1 + b_0y^0 = b_3y^3 + b_2y^2 + b_1y^1 + b_0$.

$$\begin{aligned} p(y) * y &= (b_3y^3 + b_2y^2 + b_1y^1 + b_0)y^i \\ &= (b_3y^{3+i} + b_2y^{2+i} + b_1y^{1+i} + b_0y^{0+i}) \end{aligned}$$

We know that there are four possible values i- 0, 1, 2 and 3. So we can divide it into 4 cases.
Case 1: When i=0, it means the vector is multiplied by 1 and thus there is no change which means a left circular shift by 0 bytes.

Case 2: When i=1, there will be a circular left shift by 1 byte which was shown in the previous part of this question (Q1ai).

Since we know that $y^4 = 1$, and using the property of exponents that - *if $a+b=c$, then $x^a * x^b = x^{a+b} = x^c$*, we know that whenever $a + i \geq 4$, there exists $b \geq 0$ such that $a + i = 4 + b$. So b here will be $a + i - 4$.
Case 3: In the above case when $i = 2$, we see that b_3y^{3+i} and b_2y^{2+i} will have an exponent greater than equal to 4. So the above could be written as $b_3y^{3+2} = b_3y^{4+1} = b_3y^4y^1$ and $b_2y^{2+2} = b_2y^{4+0} = b_2y^4y^0$ respectively. Since $y^4 = 1$, the new vector from this will be $[b_1\ b_0\ b_3\ b_2]$ since the polynomial turns into $b_3y^1 + b_2y^0 + b_1y^3 + b_0y^2$. So again we see a leftwards circular shift by two bytes.

Case 4: Similarly when $i = 3$, we see that b_3y^{3+i}, b_2y^{2+i} and b_1y^{1+i} will have an exponent greater than equal to 4. So this could be written as $b_3y^{3+3} = b_3y^{4+2} = b_3y^4y^2$, $b_2y^{2+3} = b_2y^{4+1} = b_2y^4y^1$ and $b_1y^{1+3} = b_1y^{4+0} = b_1y^4y^0$ respectively. Since $y^4 = 1$, the new vector from this will be $[b_0\ b_3\ b_2\ b_1]$ since the polynomial turns into $b_3y^2 + b_2y^1 + b_1y^2 + b_0y^3$. So again we see a leftwards circular shift by three bytes

in this case bytes.

In all the above cases, we see that whenever we multiply y^i to a 4 byte vector, we see i byte left circular shift where $0 \leq i \leq 3$.

(b) i. $c_1(x) = 1x^0 = 1$
$c_2(x) = 1x^1 + 0x^0 = x$
$c_3(x) = 1x^1 + 1x^0 = x + 1$

ii. let $b = [b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0]$ which could be written in polynomial form as

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0$$

We know that $(02) = c_2(x) = x$ from the previous part. Let d(x) represent the polynomial form of the byte d. Since $d = (02)b$, it implies $d(x) = b(x) * c_2(x)$.

$$\begin{aligned} d(x) &= (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0) * x \\ &= b_7x^{7+1} + b_6x^{6+1} + b_5x^{5+1} + b_4x^{4+1} + b_3x^{3+1} + b_2x^{2+1} + b_1x^{1+1} + b_0x^{0+1} \\ &= b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x^1 \end{aligned}$$

We know that $m(x) = x^8 + x^4 + x^3 + x + 1$ and in this arithmetic $m(x) = 0$. So $0 = x^8 + x^4 + x^3 + x + 1$, which can be written as $x^8 = x^4 + x^3 + x + 1$ (no negative sign because subtraction is the same as addition here). So using the expression we get:

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

so $d = [b_6\ b_5\ b_6\ (b_3 + b_7)\ (b_7 + b_2)\ b_1\ (b_7 + b_0)\ b_7]$

iii. let $b = [b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0]$ which could be written in polynomial form as

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0$$

We know that $(03) = c_3(x) = x + 1$ from the previous part. Let e(x) represent the polynomial form of the byte e. Since $e = (03)b$, it implies $e(x) = b(x) * c_3(x)$.

$$\begin{aligned} e(x) &= (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0) * (x + 1) \\ &= (b_7x^{7+1} + b_6x^{6+1} + b_5x^{5+1} + b_4x^{4+1} + b_3x^{3+1} + b_2x^{2+1} + b_1x^{1+1} + b_0x^{0+1}) + \\ &(b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0) \\ &= b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x^1 + b_7x^7 + b_6x^6 + b_5x^5 \\ &+ b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 \end{aligned}$$

We know that $m(x) = x^8 + x^4 + x^3 + x + 1$ and in this arithmetic $m(x) = 0$. So $0 = x^8 + x^4 + x^3 + x + 1$, which can be written as $x^8 = x^4 + x^3 + x + 1$ (no negative

sign because subtraction is the same as addition here). So using the expression we get:

$$\begin{aligned}
e(x) &= b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x^1 + b_7x^7 + b_6x^6 + b_5x^5 \\
&+ b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 \\
&= b_7(x^4 + x^3 + x + 1) + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x^1 + b_7x^7 \\
&+ b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 \\
&= b_7x^4 + b_7x^3 + b_7x + b_7 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x^1 + b_7x^7 \\
&+ b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 \\
&= b_7x^7 + b_6x^7 + b_6x^6 + b_5x^6 + b_5x^5 + b_4x^5 + b_3x^4 + b_4x^4 + b_7x^4 \\
&+ b_2x^3 + b_3x^3 + b_7x^3 + b_1x^2 + b_2x^2 + b_0x^1 + b_1x^1 + b_7x + b_0 + b_7 \\
&= (b_7 + b_6)x^7 + (b_6 + b_5)x^6 + (b_5 + b_4)x^5 + (b_3 + b_4 + b_7)x^4 \\
&+ (b_2 + b_3 + b_7)x^3 + (b_1 + b_2)x^2 + (b_0 + b_1 + b_7)x + (b_0 + b_7)
\end{aligned}$$

So e =$[(b_7 + b_6)\ (b_6 + b_5)\ (b_5 + b_4)\ (b_3 + b_4 + b_7)\ (b_2 + b_3 + b_7)\ (b_1 + b_2)\ (b_0 + b_1 + b_7)\ (b_0 + b_7)]$

(c) i. we know the following:

$$c(y) = (03)y^3 + (01)y^2 + (01)y + (02)$$

$$s(y) = s_3y^3 + s_2y^2 + s_1y + s_0$$

Since $m(y) = y^4+1$ and $m(y) = 0$(in this arithmetic), the following is implied $y^4 = 1$
Calculating $t(y) = s(y)c(y)\ mod\ y^4 + 1$

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

So $t(y) = t_3y^3 + t_2y^2 + t_1y + t_0$ where-

- $t_3 = (((03)s_0) + ((01)s_1) + ((01)s_2) + ((02)s_3))$,
- $t_2 = (((01)s_0) + ((01)s_1) + ((02)s_2) + ((03)s_3))$,
- $t_1 = (((01)s_0) + ((02)s_1) + ((03)s_2) + ((01)s_3))$ and
- $t_0 = (((02)s_0) + ((03)s_1) + ((01)s_2) + ((01)s_3))$.

.

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

This is found using the equation above. Since calculating the dot product is just multiplying the the column (with s_i) with the each row of C. So s_i multiplies with all the elements in the column i, where $0 \leq i < 4$. And then the sum of all elements in a row are added together. Then we get a matrix with one column and four rows that is the resultant vector. This is the same thing we had in the previous question, just in the vector form. So just fetching the coefficients from the equations gave the C.

Problem 2 — Error propagation in block cipher modes

(a) i. For ECB mode, since it does not rely on the previous cipher text for encryption, the error in the transmission of a particular block C_i affects the decryption of that that particular block only.

ii. For CBC mode, each plain text is XORed with the previous block's cipher text before encrypting to avoid frequency analysis. So in this case when there is an error with a particular block C_i, then this affects the blocks i and i+1. Since without having the correct C_i, the decryption part would fail for block i. While the blocks i+1 needs to XOR the C_i after decryption to get the correct plain text and since C_i is erroneous, it cannot achieve that. All the blocks after i+1 could be decrypted correctly since that only need the cipher text of the previous block to get the correct plain text.

iii. For OFB mode, each plaintext is XORed with the encryption of IV. So, When decrypting, we use the same method as encryption and just XOR the ciphertext and the encrypted IV to get the plain text. So an error in C_i only affects the decryption of the same block since the cipher text is not part of the encryption process nor the decryption process. The only thing used for encryption and decryption process is just the IV. So $E(IV_i)$ is the IV of block i+1. And that is the only information from the previous block that is required and also note that IV is not dependent on cipher texts.

iv. Similar to the CFB mode, since this block cipher uses the cipher text of the previous block for its decryption process and since C_i is erroneous, the decryption of the blocks i and i+1 are only affected. Also since this CFB mode has only one register, the error can be rectified by changing the bits and it is possible to get the correct plain text.

v. In CTR mode, we use a nonce and a counter and encrypt it and XOR it with plaintext while encryption. Similarly for decryption, the same noonce and counter is used and encrypted and then XORed with the cipher text. So if cipher text C_i is erroneous, only that single block is corrupted and no other blocks are affected since no other block uses another block's cipher text for decryption.

(b) So when a particular message block, M_i, is erroneous, then while decryption, only block i would be erroneous. It would not affect any other blocks as a result of wrong plain text as long as the cipher text of the wrong plain text is used for encryption and that cipher is being sent to Bob. So since the cipher text of the wrong message is sent, all the other blocks will work in the correct fashion and will yield the correct plain text. The C_i would be decrypted and would output the same erroneous plain text that was used while encryption.

Problem 3 — Binary exponentiation

(a) Computing $17^{11} \pmod{77}$ using the Binary exponentiation algorithm. So $n = 11$ which in binary form is 1011. So binary representation of n is

$$n = (b_0 * 2^3) + (b_1 * 2^2) + (b_2 * 2^1) + (b_3 * 2^0) = (1 * 2^3) + (0 * 2^2) + (1 * 2) + 1$$

And we know the following:

$$r_0 \equiv 17 \pmod{77}$$

Using this information we can get the following:

$$r_1 = r_0^2 \pmod{77} = 17^2 \pmod{77} = 58$$

$$r_2 = r_1^2 * 17 \pmod{77} = 58^2 * 17 \pmod{77} = 54$$

$$r_3 = r_2^2 * 17 \pmod{77} = 54^2 * 17 \pmod{77} = 61$$

So $r_k = r_3 = 61$ that is $61 \equiv 17^{11} \pmod{77}$

(b) i. Proving that the expression is true using induction.
Base Case (i=0): We know from the question that $s_0 = 1$. Proving that the expression is true for the base case i=0.

$$\begin{aligned} s_i &= \sum_{j=0}^{i} b_j * 2^{i-j} \\ &= \sum_{j=0}^{0} b_j * 2^{0-j} \\ &= b_0 * 2^{0-0} = b_0 * 1 \\ s_i &= b_0 \end{aligned}$$

Since $b_0 = 1$, we can conclude that that expression is true for the base case as $s_0 = 1 = b_0$.

Inductive hypothesis: Suppose the expression is true for some integer i=x, where $0 < x < k$ that is: $s_x = \sum_{j=0}^{x} b_j * 2^{x-j}$.

Inductive step: We want to prove that the expression is true for x+1. We know

that $s_{i+1} = 2s_i + b_{i+1}$. So,

$$s_{x+1} = 2s_x + b_{x+1}$$

using inductive hypothesis we get:

$$\begin{aligned}
&= 2(\sum_{j=0}^{x} b_j * 2^{x-j}) + b_{x+1} \\
&= \sum_{j=0}^{x} b_j * 2^{x-j} * 2 + b_{x+1} \\
&= \sum_{j=0}^{x} b_j * 2^{x+1-j} + b_{x+1} \\
&= (b_0 * 2^{x+1} + ... + b_x * 2^{x+1-x}) + (b_{x+1} * 2^0) \\
&= \sum_{j=0}^{x+1} b_j * 2^{(x+1)-j}
\end{aligned}$$

Hence it is true for k+1. Thus by principle of mathematical induction, the expression $s_i = \sum_{j=0}^{i} b_j * 2^{i-j}$ is true for all $0 \leq i \leq k$.

ii. Proving the expression is true using induction.
Base case (i=0): We already know that $r_0 \equiv a \mod m$. Proving that the expression is true for this case.

$$\begin{aligned}
r_i &\equiv a^{s_i} \mod m \\
&\equiv a^{s_0} \mod m \\
&\equiv a^1 \mod m \\
&\equiv a \mod m
\end{aligned}$$

Thus the expression is true for the base case.
Inductive hypothesis: Let the expression be true for $i = y$,where $0 < y < k$, such that $r_y \equiv a^{s_y} \mod m$.
Inductive Step: We want to prove that the expression is true for y+1. We know that

$$r_{i+1} = \begin{cases} r_i^2 \mod m & \text{if } b_{i+1} = 0 \\ r_i^2 * a \mod m & \text{if } b_{i+1} = 1 \end{cases}$$

We can divide this into two cases:

Case 1 ($b_{y+1} = 0$): So $r_{y+1} = r_y^2 \mod m$. Using the inductive hypothesis we get the following:

$$\begin{aligned}
r_{y+1} &= (a^{s_y} \mod m)^2 \mod m \\
&= (a^{s_y})^2 \mod m \\
&= a^{2s_y} \mod m \\
&= a^{2s_y+0} \mod m \\
&= a^{2s_y+b_{y+1}} \mod m
\end{aligned}$$

using the expression $s_{i+1} = 2s_i + b_{i+1}$,

$$= a^{s_{y+1}} \mod m$$

Case 2 ($b_{y+1} = 1$): So $r_{y+1} = r_y^2 * a \mod m$. Using the inductive hypothesis we get the following:

$$\begin{aligned} r_{y+1} &= (a^{s_y} \mod m)^2 * a \mod m \\ r_{y+1} &= (a^{s_y})^2 * a \mod m \\ r_{y+1} &= a^{2s_y} * a \mod m \\ r_{y+1} &= a^{2s_y+1} \mod m \\ r_{y+1} &= a^{2s_y+b_{y+1}} \mod m \end{aligned}$$

using the expression $s_{i+1} = 2s_i + b_{i+1}$,

$$= a^{s_{y+1}} \mod m$$

Since in both the cases, the expression is true, by the principle of mathematical induction, the expression $r_i \equiv a^{s_i} \mod m$ is true for $0 \leq i \leq k$.

iii. Proving that $a^n \equiv r_k \mod m$

$$\begin{aligned} r_k \mod m &= (a^{s_k} \mod m) \mod m \\ &= a^{s_k} \mod m \\ &= a^{\sum_{j=0}^{i} b_j 2^{i-j}} \mod m \end{aligned}$$

Using binary representation of n we get,

$$= a^n \mod m$$

Thus $r_k \mod m = a^n \mod m$. Thus the algorithm computes $= a^n \mod m$ as claimed.

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

(a) Alice receives the $(g^b)^q \mod p$ and she knows a, so the key Alice will use is $((g^b)^q)^a \mod p$, which is the same as $g^{abq} \mod p$.
Bob receives the $(g^a)^q \mod p$ and he knows b, so the key Bob will use is $((g^a)^q)^b \mod p$, which is the same as $g^{abq} \mod p$.
Hence they both compute the same shared key.

(b) Since the $K = (g^q)^{ab}$. We know using Fermat's little theorem that $a^{p-1} \equiv 1 \mod p$. We also know that $p = mq + 1$ which entails $mq = p - 1$. Consider ab to be equivalent to m. So,

$$\begin{aligned} g^{(ab)q} \mod p &= g^{mq} \mod p \\ &= (g^{p-1} \mod p) \\ &= (1 \mod p) \\ &= 1 \end{aligned}$$

Since we know ab is a big number and m is a small number, so $ab \geq m$ and so ab could be represented as $(ab) = l(m) + r$ where $l, r \in \mathbb{Z}$ and $0 \leq r \leq m - 1$. We see there are only m possible r values and hence there are m possible values for K.

(c) So in the usual man in the middle attack that was discussed in the class, Alice has the key g^{aq} and thinks it is g^{ab} and similarly Bob has the key g^{bq} and thinks it is g^{ab}. We see that in this case, during their communication, Mallory needs to decrypt the message first with one key, and then encrypt with another key and send it. So Mallory needs be a part of their communication since if Mallory does not decrypt and then encrypt with the other key, when the message reaches the receiver and upon decryption attempt, they may fail to do so and could get suspicious or may get to know that they were a man in the middle attack victim. While in the modified one, since all the parties that is ALice, Bob and Mallory all thave the same key, Mallory will not have to receive, then decrypt and then encrypt and send. She may just let the communication go as usual. She will only have to decrypt it when she wants to actively attack that is modify, add or delete something to the messages. So whenever Mallory decides to leave or rejoin, Alice and Bob would not be aware that they were or are under attack. So the main advantage for Mallory is her involvement in the communication. In the modified one, she can decide whenever she wants to be involved in the attack while in the usual man in the middle attack, she will have to be involved in the communication to stay undetected.

Problem 5 — A simplified password-based key agreement protocol

(a) Showing that $K_{server} = K_{client}$.

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

(b) Since Mallory know the pair (I, v), and since g and N are public, Mallory can masquerade as Ian. This is because the server uses the expression, $(Av)^b \mod N$ to derive K_{server} and since Mallory has to send $A = g^a \mod N$, she could choose A in such a way she can show that she knows $B^{a+p} = K_{client}$. So if $A = v^{-1}$ that is a multiplicative inverse of $v \mod N$, then

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

Since finding multiplicative inverse is a feasible and a simple task, Mallory can calculate A such that it is a multiplicative inverse. Now, to show that Mallory has knowledge about K_{client}, she can just send back 1 since that is the only possible key that could be generated using the A that Mallory sent. This is how Mallory generates a valid key that the server believes is a shared key with Ian.

(c) Suppose Mallory knows A and B. Suppose she can solve the Key recovery problem for any possible v. So she can compute $K_{server} = K_{client}$. This means she is able to compute $K_{server} = (Av)^b \mod N$. Since

$$\begin{aligned} (Av)^b \mod N &= (A^b \mod N * v^b \mod N) \mod N \\ &= (((g^a)^b \mod N) * (v^b \mod N)) \mod N \end{aligned}$$

If Mallory is able to make $v^b \mod N = 1$, she will be able to solve the Diffie Hellman problem as well. Since N is public, if Mallory chooses $p = N - 1$, then by fermet's little theorem, $v = g^p \mod N = g^{N-1} \mod N = 1$. So now continuing the above equation, we get:

$$\begin{aligned} (((g^a)^b \mod N) * (v^b \mod N)) \mod N &= (((g^a)^b \mod N) * (1^b \mod N)) \mod N \\ &= (g^a)^b \mod N \end{aligned}$$

Thus this gives us the $g^{ab} \mod N$ which was key needed. Hence solving the diffie hellman problem is as hard as solving the key recovery problem.