Name: Joshua Dow

Student ID: 10150588

Problem 1 — Binary polynomial arithmetic, 20 marks

(a) i. $x^3, x^3+1, x^3+x, x^3+x+1, x^3+x^2, x^3+x^2+1, x^3+x^2+x, x^3+x^2+x+1$

ii. A. $f(x) = x^3 = x * x * x, f(0) = 0$

B. $f(x) = x^3+1, f(0) = 0+1 \neq 0$, but $f(1) = 1+1 = 0$ in GF(2)

C. $f(x) = x^3+x = x(x^2+1), f(0) = 0, f(1) = 1(1+1) = 0$

D. $f(x) = x^3+x^2 = x^2(x+1), f(0) = 0, f(1) = 1(1+1) = 0$

E. $f(x) = x^3+x^2+x = x(x^2+x+1), f(0) = 0$

F. $f(x) = x^3+x^2+x+1, f(0) = 0+0+0+1 \neq 0, f(1) = 1+1+1+1 = 0$

iii. A. $f(x) = x^3+x+1, f(0) = 0+0+1 \neq 0, f(1) = 1+1+1 = 1$ No roots exist in GF(2)

B. $f(x) = x^3+x^2+1, f(0) = 0+0+1 \neq 0, f(1) = 1+1+1 \neq 0$ No roots exist in GF(2)

(b) i. $f(x)g(x) = (x^2+1)(x^3+x^2+1) = x^5+x^4+x^2+x^3+x^2+1 = x^5+x^4+x^3+1$

$$
\begin{array}{r l}
 & \qquad\qquad\qquad\qquad x+1 \\
x^4+x+1 \,\big) & \overline{x^5+x^4+x^3+0x^2+0x+1} \\
 & \underline{x^5+0+0+x^2+x+0} \\
 & \underline{x^4+x^3+x^2-x+1} \\
 & \underline{x^4+0+0+x+1} \\
 & \underline{0x^4+x^3+x^2-2x+0} \\
 & \underline{x^3+x^2}
\end{array}
$$

Since -2x in GF(2) is zero. So $f(x)g(x) \equiv x^3+x^2 (\mod x^4+x+1)$

ii. We want $xg(x) \equiv 1 (\mod x^4+x+1$, since $p(x) = x^4+x+1 = 0$ in $GF(2^4)$, then $xg(x) + 1 * p(x) = 1$ from an extended definition of remainder of greatest common denominator and division algorithm then:
$xg(x) + (x^4+x+1) = 1$
$xg(x) + x^4 + x = 0$
$g(x) + x^3 + 1 = 0$
$g(x) = x^3+1$ We can ignore the negative here due to the cyclic nature of $GF(2^4)$
To verify:
$xg(x) = x(x^3+1) = x^4+x \equiv 1 (\mod x^4+x+1)$

(c) i. Proof that in this arithmetic, multiplication of any 4-byte vector by y is a circular left shift of the vector by one byte:
Suppose any 4-byte vector $abcd$ which we represent by: ay^3+by^2+cy+d. Then multiplication by y yields: $ay^4+by^3+cy^2+dy) = a+by^3+cy^2+dy = by^3+cy^2+dy+a$ since $y^4 = 1$ which is a circular shift left. We can continue this process as follows:
$y(by^3+cy^2+dy+a) = cy^3+dy^2+ay+b$
$y(cy^3+dy^2+ay+b) = dy^3+ay^2+by+c$

$y(dy^3 + ay^2 + by + a) = ay^3 + by^2 + cy + d$ Which is where we started. $\square$

ii. We need not use induction since we have a small finite set of elements. We can prove this on a case by case basis.

A. Case 1: $i = 4k, k \in Z$ then $j = 0$ so $y^0(\mod y^4 + 1) = 1(\mod y^4 + 1) \equiv 0$ and for any other multiple of 4 for $i = 4k$ then $y^{4k}(\mod y^4 + 1) \equiv 1^k(\mod y^4 + 1) \equiv 1(\mod y^4 + 1) \equiv 0$

B. Case 1: $i = 4k + 1, k \in Z$ then $j = 1$ so $y^1(\mod y^4 + 1) = y(\mod y^4 + 1) \equiv y$ and for any other multiple of 4 for $i = 4k + 1$ then $y^{4k+1}(\mod y^4 + 1) \equiv y * 1^k(\mod y^4 + 1) \equiv y(\mod y^4 + 1) \equiv y$

C. Case 1: $i = 4k + 2, k \in Z$ then $j = 2$ so $y^2(\mod y^4 + 1) = y^2(\mod y^4 + 1) \equiv y^2$ and for any other multiple of 4 for $i = 4k + 2$ then $y^{4k+2}(\mod y^4 + 1) \equiv y^2 * 1^k(\mod y^4 + 1) \equiv y^2(\mod y^4 + 1) \equiv y^2$

D. Case 1: $i = 4k + 3, k \in Z$ then $j = 3$ so $y^3(\mod y^4 + 1) = y^3(\mod y^4 + 1) \equiv y^3$ and for any other multiple of 4 for $i = 4k + 3$ then $y^{4k+3}(\mod y^4 + 1) \equiv y^3 * 1^k(\mod y^4 + 1) \equiv y^3(\mod y^4 + 1) \equiv y^3$

iii. Proof by induction on i:
Base Case: Let $i = 0$, then $y^i = y^0 = 1$ so for any 4-byte vector $ay^3 + by^2 + cy + d$ multiplication by 1 is itself.
Inductive Hypothesis: Suppose a 4-byte vector $qy^3 + ry^2 + sy + t$ and $i \geq 0$ we have a circular left shift such that $y^i(qy^3 + ry^2 + sy + t)$ is of the form: $qy^{3+i} + ry^{2+i} + sy^{1+i} + ty^i$.
Inductive Step: Let $j = i + 1 \geq 0$, then $y^j(qy^3 + ry^2 + sy + t)$
$= y^{i+1}(qy^3 + ry^2 + sy + t) = y^i y^1(qy^3 + ry^2 + sy + t) = y^i(qy^4 + ry^3 + sy^2 + ty)$
$= y^i(ry^3 + sy^2 + ty + r) = ry^{3+i} + sy^{2+i} + ry^{1+i} + ty^i$ By the inductive hypothesis we can see a singular left shift here, and corresponding to the value of i will continue to shift left. This concludes out induction on i $\square$

Problem 2 — Arithmetic with the constant polynomial of MixColumns in AES, 13 marks

(a) i. $c_1(x) = 0x01 \cdot x + 0x01 \cdot x^2$

ii. $c_2(x) = 0x02$

iii. $c_3(x) = 0x03 \cdot x^3$

(b) i. $d = (02)b = 0x02 \cdot b = (00000010)(b_7b_6b_5b_4b_3b_2b_1b_0)$
$= b_6b_5b_4b_3b_2b_1b_0b_7$, then for any d_i we have:
$d_i = b_{i-1}$ This represents a bitwise left shift

ii. $e = (03)b = 0x03 \cdot x^3 \cdot b = (00000011)x^3(b_7b_6b_5b_4b_3b_2b_1b_0)$
$= (b_6b_5b_4b_3b_2b_1b_0b_7 + b_7b_6b_5b_4b_3b_2b_1b_0)x^3$ This is known as a shift + add, we shift left by 1 ($3 = 2 + 1$) and then add the original bits

(c) i. $s(y)c(y) = ((03)y^3 + (01)y^2 + (01)y + (02))(s_3y^3 + s_2y^2 + s_1y + s_0)$

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

$$\begin{aligned}
&= (03)s_3y^2 + (03)s_2y + (03)s_1 + (03)s_0y^3 \\
&+(01)s_3y + (01)s_2 + (01)s_1y^3 + (01)s_0y^2 \\
&+(01)s_3 + (02)s_2y^3 + (01)s_1y^2 + (01)s_0y \\
&+(02)s_3y^3 + (02)s_2y^2 + (02)s_1y + (02)s_0
\end{aligned}$$

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

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

ii. **Problem 3** — Error propagation in block cipher modes, 12 marks

(a) i. In ECB, an error in C_i only affects M_i since each block is individually encrypted

ii. In CBC, an error in C_i affects every encryption after since each previous encryption is used as input to the next encryption. So on decryption each M after and including M_i is effected.

iii. In OFB, an error in C_i only affects M_i because C_i is not used in the calculations that follow

iv. In CFB with one register, an error in C_i affects every calculation afterwards. This is because with only one register the previous encryption is used in the calculation for the following calculations.

v. In CTR, an error in C_i affects only M_i on decryption because CTR is independent of previous calculations

(b) Only that one block will be affected, in regular CFB the error will not propagate due to how CFB synchronizes when a familiar state is reached. So once we leave the corrupted block, the remaining calculations will be fine.

Problem 4 — Flawed MAC designs, 24 marks

(a) i. Supposing that M_1 consists of L blocks, then going through $ITHASH(K||M_1)$ gives us $PHMAC_K(M_1)$. But notice that the first L+1 round for both $ITHASH(K||M_1), ITHASH(K||$ are the same regardless of the choice of K, so we can calculate $PHMAC_K(M_2)$ by applying the compression algorithm f to $PHMAC_K(M_1)$ and X.

ii. Since $AHMAC$ is not weak collision resistant the there is a message $M_2 \neq M_1$ such that $AHMAC_K(M_1) = AHMAC_K(M_2) \iff ITHASH(M_1||K) = ITHASH(M_2||K)$. Only the last round of the computation depends on K. By the L^{th} round of computation the output is the same so on the $L+1^{st}$ round is $H \longleftarrow f(H, K)$ so we have now generated an additional message AHMAC pair.

(b) i. $CBC - MAC(M_3) = e_k(e_k(M_1) \oplus e_k(0^n)) = e_k(CBC - MAC(M_1) \oplus e_k(0^n)$
$= e_k(M_2 \oplus 0^n) = e_k(M_2) = CBC - MAC(M_2)$
This violates computational resistance because we end up with a strong collision since $CBC - MAC(M_3) = CBC - MAC(M_2)$

ii. $CBC - MAC(M_4) = e_k(e_k(M_2) \oplus CBC - MAC(M_1) \oplus CBC - MAC(M_2) \oplus X)$
$= e_k(CBC - MAC(M_2) \oplus CBC - MAC(M_1) \oplus CBC - MAC(M_2) \oplus X)$
$= e_k(CBC - MAC(M_1) \oplus X)$
$= CBC - MAC(M_3)$
This violates computational resistance by having two seperate messages encrypted to the same MAC.