Some Observations on Arithmetic Addition in $GF(2^n)$

- We know that the polynomial coefficients in $GF(2^n)$ must obey the arithmetic rules that apply to $GF(2)$ (which is the same as Z_2, the set of remainders modulo 2).
- And we know that the operation of addition in $GF(2)$ is like the logical XOR operation.
- Therefore, adding the bit patterns in $GF(2^n)$ simply amounts to taking the bitwise XOR of the bit patterns.
- The examples in the next page hold in $GF(2^8)$. The last two examples of them illustrate that **subtracting is the same as adding in $GF(2^8)$**. That is because each "number" is its own additive inverse in $GF(2^8)$. In other words, for every $x \in GF(2^8)$, we have $-x = x$. Yet another way of saying the same thing is that for every $x \in GF(2^8)$, we have $x + x = 0$.

Examples on Arithmetic Addition in $GF(2^n)$

$$5 + 13 = 0000\ 0101 + 0000\ 1101 = 0000\ 1000 = 8$$

$$76 + 22 = 0100\ 1100 + 0001\ 0110 = 0101\ 1010 = 90$$

$$7 - 3 = 0000\ 0111 - 0000\ 0011 = 0000\ 0100 = 4$$

$$7 + 3 = 0000\ 0111 + 0000\ 0011 = 0000\ 0100 = 4$$

Some Observations on Arithmetic Multiplication in GF (2^n) (1)

- Just as it is convenient to use the simple binary arithmetic (in the form of XOR operations) for additions in *GF* (2^n), could we do the same for multiplications?
- Recall, that we can of course multiply the bit patterns of *GF* (2^n) by going back to the modulo polynomial arithmetic and using the multiplications operations defined in *GF* (2) for the coefficients. (Recall that in *GF* (2), multiplication is the same as logical AND.)
- But it would be nice if we could directly multiply the bit patterns of *GF* (2^n) without having to think about the underlying polynomials directly.

Some Observations on Arithmetic Multiplication in GF (2^n) (2)

- It turns out that we can indeed do so, but the technique is specific to the order of the finite field being used. The order of a finite field refers to the number of elements in the field. So the order of *GF* (2^n) is 2^n .
- More particularly, the bitwise operations needed for directly multiplying two bit patters in *GF* (2^n) are specific to the irreducible polynomial that defines a given *GF* (2^n).
- On the next slide, we will focus specifically on the *GF* (2^8) finite field that is used in AES and show multiplications can be carried out directly in this field by using bitwise operations.

Direct Bitwise Operations for Multiplications in $GF(2^8)$ (1)

- Let's consider the finite field $GF(2^8)$ that is used in AES. This field is derived using the following irreducible polynomial of degree 8:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

- Now let's see how we can carry out multiplications with direct bitwise operations in this $GF(2^8)$.
- We first take note of the following equality in $GF(2^8)$:

$$x^8 \bmod m(x) = x^4 + x^3 + x + 1$$

The result follows immediately by a long division of x^8 by $x^8 + x^4 + x^3 + x + 1$. Obviously, the first term of the quotient will be 1. Multiplying the divisor by the quotient yields $x^8 + x^4 + x^3 + x + 1$. When this is subtracted from the dividend x^8, we get $-x^4 - x^3 - x - 1$, which is the same as the result shown above.

Direct Bitwise Operations for Multiplications in $GF(2^8)$ (2)

- Now let's consider the general problem of multiplying a general polynomial $f(x)$ in $GF(2^8)$ by just x. Let's represent $f(x)$ by

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

  Therefore, this $f(x)$ stands for the bit pattern $b_7b_6b_5b_4b_3b_2b_1b_0$.

- Obviously,

$$f(x) \times x = b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$$

  But now recall that we must take the modulo of this polynomial with respect to $m(x) = x^8 + x^4 + x^3 + x + 1$. What that yields depends on whether or not the bit b_7 is set.

- If the bit b_7 of $f(x)$ is equals 0, then the right hand above is already in the set of polynomials in $GF(2^8)$ and nothing further needs to be done. In this case, the output bit pattern is $b_6b_5b_4b_3b_2b_1b_00$.

Direct Bitwise Operations for Multiplications in $GF(2^8)$ (3)

- However, if b_7 equals 1, we need to divide the polynomial we have for $f(x) \times x$ by the modulus polynomial $m(x)$ and keep just the remainder. Therefore, when $b_7 = 1$, we can write

$$
\begin{aligned}
&(f(x) \times x) \bmod m(x) \\
&= (b_7x^8 + b_6x^7 + b_5x^6 + x_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x) \\
&= (b_6x^7 + b_5x^6 + x_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \quad + \quad (x^8 \bmod m(x)) \\
&= (b_6x^7 + b_5x^6 + x_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1) \\
&= (b_6b_5b_4b_3b_2b_1b_00) \quad \otimes \quad (00011011)
\end{aligned}
$$

where, in the last expression shown, we have used the fact that the addition in $GF(2^8)$ corresponds to the logical XOR {$\otimes$} operation for the bit patterns involved.

Summary of How a Multiplication is Carried Out in $GF(2^8)$ (1)

- Let's say you want to multiply two bit patterns B_1 and B_2, each 8 bits long.
- If B_2 is the bit pattern 00000001, then obviously nothing needs to be done. The result is B_1 itself.
- If B_2 is the bit pattern 00000010, then we are multiplying B_1 by x. Now the answer depends on the value of the most significant bit in B_1. If B_1's MSB is 0, the result is obtained by shifting the B_1 bit pattern to the left by one bit and inserting a 0 bit from the right.
- If B_1's MSB is 1, first we again shift the B_1 bit pattern to the left as above. Next, we take the XOR of the shifted pattern with the bit pattern 00011011 for the final answer.

Summary of How a Multiplication is Carried Out in $GF(2^8)$ (2)

- If B_2 is the bit pattern 00000100, then we are multiplying B_1 by x^2. This amounts to first multiplying B_1 by x, and then multiplying the result again by x. So it amounts to two applications of the logic in the previous two steps.
- In general, if B_2 consists of a single bit in the j^{th} position from the right (using the 0 index for the right-most position), we need j applications of the logic laid out above for multiplying with x.
- Even more generally, when B_2 consists of an arbitrary bit pattern, we consider the bit pattern to be a sum of bit patterns each containing only single bit.

Summary of How a Multiplication is Carried Out in *GF* (2^8) (3)

- For example, if B_2 is 10000011, we can write

$$
\begin{aligned}
& B_1 \times 10000011 \\
&= B_1 \times (00000001 + 00000010 + 10000000) \\
&= (B_1 \times 00000001) + (B_1 \times 00000010) + (B_1 \times 10000000) \\
&= (B_1 \times 00000001) \otimes (B_1 \times 00000010) \otimes (B_1 \times 10000000)
\end{aligned}
$$

Each of the three multiplications shown in the final expression involves multiplying B_1 with a single power of x. That we can easily do with the logic already explained.

Finding Multiplicative Inverses in $GF(2^n)$ (1)

- So far we have talked about efficient bitwise operations for implementing the addition, the subtraction, and the multiplication operations for the bit patterns corresponding to the elements of $GF(2^n)$.
- But what about **division**? Can division be carried out directly on the bit patterns?
- In general, you can use the **Extended Euclid Algorithm** for finding the multiplicative inverse (MI) of a polynomial in $GF(2^n)$.
- If you have fixed the value of n for a particular $GF(2^n)$ field (and if n is not too large), you can pre-compute the multiplicative inverses for all the elements of $GF(2^n)$ and store them away. (Recall that the MI of a bit pattern A in $GF(2^n)$ is a bit pattern B so that $A \times B = 1$.
- For example, in $GF(2^8)$, the MI of a bit pattern A is the bit pattern B so that $A \times B = 000000001$.

Finding Multiplicative Inverses in $GF(2^n)$ (2)

- The table below shows the multiplicative inverses for the bit patterns of $GF(2^3)$. Also shown are the additive inverses. But note that every element x is its own additive inverse. Also note that the additive identity element is not expected to possess a multiplicative inverse.

	Additive Inverse	Multiplicative Inverse
000	000	-----
001	001	001
010	010	101
011	011	110
100	100	111
101	101	010
110	110	011
111	111	100

Finally . . .

- Acknowledgment: These lecture notes are based on the textbook by William Stallings and notes prepared by Avinash Kak, Purdue University. My sincere thanks are devoted to them and to all other people who offered the material on the web.

- Students are advised to study and solve the problems and answer the questions in **Assignment-A3**.