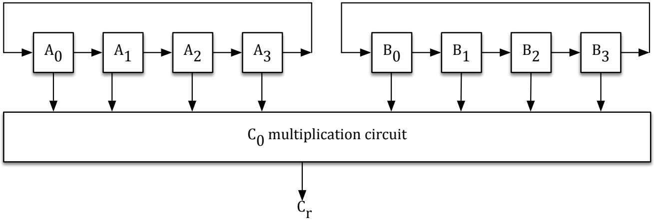
Arithmetic in Binary Fields



Representing Elements of $GF(2^k)$

- A Galois field of 2^k elements is denoted as $GF(2^k)$
- Such field are also called “binary fields” since the field elements can be represented using k-bit binary vectors
- For example, if $a \in GF(2^k)$, then $A_i \in \{0, 1\}$

$$a = (A_{k-1} A_{k-2} \cdots A_1 A_0)$$

- The 0 and 1 bits above are the coefficients of the basis elements
- There are two types of basis which are of interest in cryptography: the **polynomial basis** and the **normal basis**

Polynomial Basis Representation of GF(2^k)

- The polynomial basis is formed by taking the root α of a degree-k irreducible polynomial over GF(2), and representing every element of the field in a linear sum of the powers of α

$$\begin{aligned} A &= (A_{k-1}A_{k-2}\cdots A_1 A0) \\ &= A_{k-1}\alpha^{k-1} + A_{k-2}\alpha^{k-2} + \cdots + A_1\alpha + A_0 \\ &= \sum_{i=0}^{k-1} A_i\alpha^i \end{aligned}$$

- There are 2^k different binary vectors of length k, and thus every element of GF(2^k) is uniquely represented
- $\alpha \in \mathrm{GF}(2^k)$ is represented using $(000\cdots010)$

Normal Basis Representation of GF(2^k)

- The normal basis is formed by taking an element β of the field and representing every other elements of the field in a linear sum of the power of 2 powers of β
- The 0 and 1 bits in the vector are the coefficients of the powers β, for example, for $b_i \in \{0, 1\}$

$$
\begin{aligned}
B &= (B_{k-1} B_{k-2} \cdots B_1 B0) \\
&= B_{k-1}\beta^{2^{k-1}} + B_{k-2}\beta^{k-2} + \cdots + B_1\beta^{2^1} + B_0\beta^{2^0} \\
&= \sum_{i=0}^{k-1} B_i \beta^{2^i}
\end{aligned}
$$

- There are 2^k different binary vectors of length k, and therefore, every element of GF(2^k) is uniquely represented
- $\beta \in$ GF(2^k) is represented using $(000 \cdots 001)$

Addition in GF(2^k)

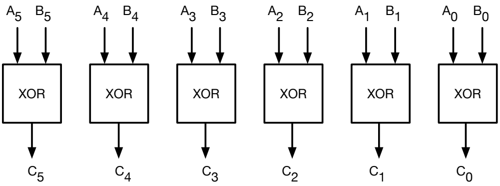
- The **addition** of two field elements represented in **polynomial basis** or **normal basis** is performed using exactly the **same algorithm**: GF(2) addition of the individual bits in the binary vectors
- However, both elements need to be in the same basis!
- Given a and b represented in polynomial basis or normal basis as vectors of length k, their sum $c = a + b \in \mathrm{GF}(2^k)$ is found as

$$\begin{array}{ccccccc} a & = & A_{k-1} & A_{k-2} & \cdots & A_1 & A_0 \\ b & = & B_{k-1} & B_{k-2} & \cdots & B_1 & B_0 \\ \hline c & = & C_{k-1} & C_{k-2} & \cdots & C_1 & C_0 \end{array}$$

- Each vector element C_i is computed using $C_i = A_i + B_i \pmod 2$
- GF(2) addition corresponds to the XOR operation in Boolean logic

Addition in GF(2^k)

- Here we have $C_i = A_i + B_i \pmod 2$ or $C_i = A_i$ XOR B_i



- GF(2^k) addition involves no carry generation or propagation
- Total delay = 1 XOR delay
- Total area = k XOR area
- Scales up easily for k
- Subtraction is the same as addition

Multiplication in GF(2^k)

- Multiplication of the elements of GF(2^k) using polynomial basis and normal basis is based on different algorithms
- Multiplication of the elements of GF(2^k) using polynomial basis is performed by multiplication of polynomials mod $p(\alpha)$
- $p(\alpha)$ is an irreducible polynomial of degree k over GF(2)
- On the other hand, multiplication of the elements of GF(2^k) using normal basis involves reduction of higher powers of the normal element β to lower powers
- Both bases may als be used simultaneously as they may offer efficiency, for example, by performing an operation in one basis and then converting to another

Polynomial Basis Multiplication in GF(2^k)

- The polynomial basis multiplication in GF(2^k) has two phases:
  - Polynomial Multiplication
  - Reduction with the degree-k irreducible polynomial $p(\alpha)$
- This is very similar to the **multiply-and-reduce** method of the **modular multiplication of integers**
- However, all additions are performed in GF(2^k), because vectors representing the field elements are not integers
- The degree-k irreducible polynomial $p(\alpha)$ is of the form

$$\alpha^k + p_{k-1}\alpha^k + p_{k-2}\alpha^{k-1} + \cdot + p_1\alpha + 1$$

  where $p_i \in \{0, 1\}$
- The first and last terms α^k and 1 must exist

Irreducible Polynomials Generating GF(2^k)

- To construct the Galois field GF(2^k), we need an irreducible polynomial $p(\alpha)$ of degree k over GF(2)
- Irreducible polynomials of any degree exist, in fact, usually there are more than one for a given k
- We choose just one of them, and keep it for our implementation
- For interoperability, the sender and receiver must choose the same irreducible polynomial
- All GF(2^k) fields generated by different irreducible polynomials (of degree k) are isomorphic to one another

Irreducible Polynomials over GF(2)

k	irreducible polynomials		
1	α	$\alpha + 1$	
2	$\alpha^2 + \alpha + 1$		
3	$\alpha^3 + \alpha + 1$	$\alpha^3 + \alpha^2 + 1$	
4	$\alpha^4 + \alpha + 1$	$\alpha^4 + \alpha^3 + 1$	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
5	$\alpha^5 + \alpha^2 + 1$	$\alpha^5 + \alpha^3 + 1$	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$
6	$\alpha^6 + \alpha + 1$	$\alpha^6 + \alpha^3 + 1$	$\alpha^6 + \alpha^5 + 1$
	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha + 1$	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha + 1$	$\alpha^6 + \alpha^5 + \alpha^2 + \alpha + 1$
	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + 1$	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + 1$	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha + 1$
7	$\alpha^7 + \alpha + 1$	$\alpha^7 + \alpha^3 + 1$	$\alpha^7 + \alpha^4 + 1$
	$\alpha^7 + \alpha^6 + 1$	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha + 1$	$\alpha^7 + \alpha^5 + \alpha^2 + \alpha + 1$
	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha + 1$	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha + 1$	$\alpha^7 + \alpha^4 + \alpha^4 + \alpha + 1$
	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + 1$	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + 1$	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + 1$
	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + 1$	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + 1$	

Irreducible Polynomials over GF(2)

k	irreducible polynomials		
8	$\alpha^8+\alpha^4+\alpha^3+\alpha+1$	$\alpha^8+\alpha^7+\alpha^2+\alpha+1$	$\alpha^8+\alpha^5+\alpha^3+\alpha+1$
	$\alpha^8+\alpha^7+\alpha^2+\alpha+1$	$\alpha^8+\alpha^6+\alpha^5+\alpha+1$	$\alpha^8+\alpha^7+\alpha^5+\alpha+1$
	$\alpha^8+\alpha^7+\alpha^6+\alpha+1$	$\alpha^8+\alpha^4+\alpha^3+\alpha^2+1$	$\alpha^8+\alpha^5+\alpha^3+\alpha^2+1$
	$\alpha^8+\alpha^6+\alpha^3+\alpha^2+1$	$\alpha^8+\alpha^7+\alpha^3+\alpha^2+1$	$\alpha^8+\alpha^6+\alpha^5+\alpha^2+1$
	$\alpha^8+\alpha^5+\alpha^4+\alpha^3+1$	$\alpha^8+\alpha^6+\alpha^5+\alpha^3+1$	$\alpha^8+\alpha^7+\alpha^5+\alpha^3+1$
	$\alpha^8+\alpha^6+\alpha^5+\alpha^4+1$	$\alpha^8+\alpha^7+\alpha^5+\alpha^4+1$	
257	$\alpha^{257}+\alpha^{12}+1$	$\alpha^{257}+\alpha^{41}+1$	$\alpha^{257}+\alpha^{48}+1$
	$\alpha^{257}+\alpha^{51}+1$	$\alpha^{257}+\alpha^{65}+1$	$\alpha^{257}+\alpha^{192}+1$
	$\alpha^{257}+\alpha^{206}+1$	$\alpha^{257}+\alpha^{209}+1$	$\alpha^{257}+\alpha^{216}+1$
	$\alpha^{257}+\alpha^{245}+1$		

Sparse Irreducible Polynomials Generating GF(2^k)

- Since any irreducible polynomial of degree k can be used to construct the field GF(2^k), it is a good idea to select one that will offer maximum arithmetic efficiency
- Due to the complexity of the reduction phase of the polynomial basis multiplication, a **sparse** or **short** irreducible polynomial is preferred
- A sparse or short irreducible polynomial of degree k has as few terms as possible
- For example, $\alpha^7 + \alpha + 1$ is irreducible over GF(2) and has just three terms, and it is the shortest irreducible polynomial of degree 7

Irreducible Trinomials and Pentanomials over GF(2)

- The shortest irreducible polynomial for any k has at least 3 terms:

$$\alpha^k + \alpha^j + 1 \text{ for some } j \in [1, k-1]$$

- Such polynomials are called **trinomials**
- The next shortest irreducible polynomial for any k has 5 terms:

$$\alpha^k + \alpha^{j_1} + \alpha^{j_2} + \alpha^{j_3} + 1 \text{ for some unequal } j_1, j_2, j_3 \in [1, k-1]$$

- Such polynomials are called **pentanomials**
- Binomials and quadrinomials are reducible over GF(2)

Irreducible Polynomials Generating GF(2^k)

- **Question 1**: Does there exist an irreducible trinomial for every k?
- Answer: No
- For example, there are no irreducible trinomials for $k = 8, 13, 16, 19$ and many others, however, there are irreducible pentanomials for these k values

- **Question 2**: Does there exist an irreducible trinomial or irreducible pentanomial for every k?
- Answer: This is an open question.
- However, the research indicates that up to $k = 10,000$ there is either an irreducible trinomial or an irreducible pentanomial for every k `http://www.hpl.hp.com/techreports/98/HPL-98-135.pdf`

Polynomial Basis Multiplication in $GF(2^7)$

- Given $a, b \in GF(2^k)$ expressed in polynomial basis, the field multiplication is performed in two phases
  - Polynomial multiplication of $a(\alpha)$ and $b(\alpha)$
    $c'(\alpha) = a(\alpha) \cdot b(\alpha)$
  - Polynomial reduction using the irreducible polynomial $p(\alpha)$
    $c(\alpha) = c'(\alpha) \bmod p(\alpha)$
- Consider $GF(2^7)$ and the irreducible trinomial

$$p(\alpha) = \alpha^7 + \alpha + 1 = (10000011)$$

- Let $a, b \in GF(2^7)$ such that

$$\begin{aligned} a &= (0100110) = \alpha^5 + \alpha^2 + \alpha \\ b &= (1001001) = \alpha^6 + \alpha^3 + 1 \end{aligned}$$

Polynomial Basis Multiplication in GF(2^7)

- Since the elements of GF(2^7) are polynomials up to the degree 6, the polynomial multiplication produces a polynomial of degree up to 12

$$\begin{aligned} c'(\alpha) = a(\alpha) \cdot b(\alpha) &= (\alpha^5 + \alpha^2 + \alpha)(\alpha^6 + \alpha^3 + 1) \\ &= \alpha^{11} + \alpha^7 + \alpha^4 + \alpha^2 + \alpha \\ &= (0100010010110) \end{aligned}$$

- There are various algorithms for the polynomial multiplication
- All additions are in GF(2)
- The add-shift algorithm produces c' as

$$\begin{array}{ccccccccccccc} & & & & & & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ & & & & & & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ \hline & & & & & & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ & & & 0 & 1 & 0 & 0 & 1 & 1 & 0 & & & \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & & & & & & \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Polynomial Basis Multiplication in GF(2^7)

- The irreducible polynomial $p(\alpha) = \alpha^7 + \alpha + 1 = (10000011)$
- We perform polynomial reduction by first left adjusting the binary vector for $p(\alpha)$ with the product vector $c'(\alpha)$
- We then perform XOR and shift-right operations, until all top (beyond α^6) terms of the product $c'(\alpha)$ are zero

$$
\begin{array}{ccccccccccccccc}
c' & = & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
p & = & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & & & & \\
\hline
c & = & & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
p & = & & & & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
\hline
c & = & & & & & & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
\end{array}
$$

- Therefore, we find $c = (0100101) = \alpha^5 + \alpha^2 + 1$

Irreducible All-One Polynomials

- Alternatively irreducible polynomials with more 1s may also be useful for efficiency purposes
- A particular set of irreducible polynomials over GF(2) is called all-one polynomials (AOPs) which are of the form

$$(11\cdots 11) = \alpha^k + \alpha^{k-1} + \cdot + \alpha + 1$$

- A degree k AOP is irreducible if and only if $p = k + 1$ is prime and 2 is a primitive mod p
- For $k \leq 100$, the AOP is irreducible for the following k values $\{2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82, 100\}$

Irreducible All-One Polynomials

- For example, for $k = 4$ we have $p = 5$ prime and 2 is primitive mod 5, therefore the AOP $\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$ is irreducible, which also happens to be a pentanomial
- Similarly, for $k = 10$ we have $p = 11$ prime and 2 is primitive mod 11, therefore the AOP

$$\alpha^{10} + \alpha^9 + \alpha^8 + \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$$

  is irreducible
- Reduction with an AOP requires XOR of the all-one $p(\alpha)$ vector with the product vector, and this can be implemented by noting that

$$C_i \text{ XOR } P_i = C_i \text{ XOR } 1 = \bar{C}_i$$

- Here $\bar{C}_i$ is the Boolean complement of C_i

Reduction with an AOP

- Consider the field $GF(2^4)$ and its irreducible AOP $p(\alpha) = \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1 = (11111)$
- Let $a = (1011)$ and $b = (1001)$ be in $GF(2^4)$, in other words, $a = \alpha^3 + \alpha + 1$ and $b = \alpha^3 + 1$
- The polynomial multiplication phase produces $c = a \cdot b$

$$c = (\alpha^3 + \alpha + 1)(\alpha^3 + 1) = \alpha^6 + \alpha^4 + \alpha + 1 = (01010011)$$

- The reduction phase produces

$$\begin{array}{ccrrrrrrrr}
c & = & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
p & = & & 1 & 1 & 1 & 1 & 1 & & \\
\hline
c & = & & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
p & = & & & 1 & 1 & 1 & 1 & 1 & \\
\hline
c & = & & & 0 & 1 & 0 & 0 & 0 & 1 \\
p & = & & & & 1 & 1 & 1 & 1 & 1 \\
\hline
c & = & & & & 0 & 1 & 1 & 1 & 0
\end{array}$$

- Therefore, we find $c = (1110) = \alpha^3 + \alpha^2 + \alpha$

Normal Basis Squaring in GF(2^k)

- The normal basis squaring in GF(2^k) is simply a left rotation of the bits of the field element for any k
- This property of the normal basis for GF(2^k) makes it very attractive for coding and cryptography applications
- Consider the element $a \in \text{GF}(2^k)$ expressed in normal basis as

$$a = \sum_{i=0}^{k-1} A_i \beta^{2^i} = A_0\beta + A_1\beta^2 + \cdots + A_{k-2}\beta^{2^{k-2}} + A_{k-1}\beta^{2^{k-1}}$$

- As a vector, we can write it as $a = (A_{k-1}A_{k-2}\cdots A_1A_0)$

Normal Basis Squaring in GF(2^k)

- We then calculate the expression for a^2 using the sum formulas
- All cross terms in the expression for a^2 disappear, leaving only

$$a^2 = \sum_{i=0}^{k-1} A_i \beta^{2^{i+1}} = A_0\beta^2 + A_1\beta^4 + \cdots + A_{k-2}\beta^{2^{k-1}} + A_{k-1}\beta^{2^k}$$

- Since $\beta^{2^k} = \beta$, we rearrange the above sum as

$$A_{k-1}\beta + A_0\beta^2 + A_1\beta^4 + \cdots + A_{k-2}\beta^{2^{k-1}}$$

- This gives the vector representation as $a^2 = (A_{k-2} \cdots A_1 A_0 A_{k-1})$
- Therefore, the squaring of a is obtained by left rotating its vector

Normal Basis Multiplication in GF(2^k)

- Given two elements $a, b \in \mathrm{GF}(2^k)$ expressed in normal basis, the normal basis multiplication algorithm will produce the product $c = a \cdot b$ in the the normal basis
- Since the power of 2 powers of the normal element β are in the expressions for a and b, the expression for c will have non-power of 2 powers of β
- For example, the product of $A_i \beta^{2^i}$ and $B_j \beta^{2^j}$ will be $A_i B_j \beta^{2^i+2^j}$
- In order to obtain an expression for c containing only the power of 2 powers of β, we need to "reduce" $\beta^{2^i+2^j}$ terms to β^{2^n}
- The irreducible polynomial $p(\alpha)$ is **implicitly** involved in this reduction, since the conversion tables from the $2^i + 2^j$ powers to the 2^n powers are obtained using $p(\alpha)$

Normal Basis Multiplication in $GF(2^4)$

- Consider the field $GF(2^4)$ and the irreducible trinomial $p(\alpha) = \alpha^4 + \alpha + 1$
- There exists a normal element β for $k = 3$, in fact, there always exists a normal element for any k
- The polynomial representation of β is found as $\beta = \alpha^3$
- We need the polynomial representation of β in order to create the conversion table from the powers $2^i + 2^j$ to the powers 2^n using the irreducible polynomial $p(\alpha)$

All Powers of β in Normal Basis

- Using $\beta = \alpha^3$ and $p(\alpha) = \alpha^4 + \alpha + 1$, we can find the polynomial representations of all power of 2 powers of β

$$
\begin{aligned}
\beta &= \alpha^3 \\
\beta^2 &= \alpha^3 + \alpha^2 \\
\beta^4 &= \alpha^3 + \alpha^2 + \alpha + 1 \\
\beta^8 &= \alpha^3 + \alpha
\end{aligned}
$$

- Now we need to find **normal expressions** for all powers of β
- These computations can be performed using computer algebra, and need to be done only once during the algorithm development
- Once they are obtained, a Boolean circuit is built that uses AND and XOR gates and rewiring to compute the bits c_i of the product

All Powers of β in Normal Basis

- In order to obtain the normal expressions for other powers of β, we can use the ones we already know
- For example, to compute β^3 we use

$$\beta^3 = \beta \cdot \beta^2 = \alpha^3 \cdot (\alpha^3 + \alpha^2) = \alpha^6 + \alpha^5 = \alpha^3 + \alpha \pmod{p(\alpha)}$$

- This gives $\beta^3 = \beta^8$
- Proceeding, we find the normal representation of all powers of β
- Furthermore, we have $\beta^0 = \beta^{15}$ and $\beta^{16} = \beta$

All Powers of β in Normal Basis

β^i	Normal Expansion	β^i	Normal Expansion
β^0	$\beta^8 + \beta^4 + \beta^2 + \beta$	β^8	β^8
β^1	β	β^9	β^4
β^2	β^2	β^{10}	$\beta^8 + \beta^4 + \beta^2 + \beta$
β^3	β^8	β^{11}	β
β^4	β^4	β^{12}	β^2
β^5	$\beta^8 + \beta^4 + \beta^2 + \beta$	β^{13}	β^8
β^6	β	β^{14}	β^4
β^7	β^2	β^{15}	$\beta^8 + \beta^4 + \beta^2 + \beta$

An Example Normal Basis Multiplication in $GF(2^4)$

- Consider two elements of $a, b \in GF(2^4)$ given in normal basis as

$$\begin{aligned} a = (1011) &= \beta^8 + \beta^2 + \beta \\ b = (1001) &= \beta^8 + \beta \end{aligned}$$

- Their product is obtained as

$$\begin{aligned} c &= (\beta^8 + \beta^2 + \beta) \cdot (\beta^8 + \beta) \\ &= \beta^{16} + \beta^{10} + \beta^3 + \beta^2 \end{aligned}$$

- Using the representations of $\beta^{16} = \beta$, $\beta^{10} = \beta^8 + \beta^4 + \beta^2 + \beta$, and $\beta^3 = \beta^8$ from the conversion table, we obtain

$$\begin{aligned} c &= \beta^{16} + \beta^{10} + \beta^3 + \beta^2 \\ &= \beta + (\beta^8 + \beta^4 + \beta^2 + \beta) + \beta^8 + \beta^2 \\ &= \beta^4 \end{aligned}$$

Normal Basis Multiplication in GF(2^4)

- The multiplication of two arbitrary elements of in normal basis

$$
\begin{aligned}
a &= A_0\beta + A_1\beta^2 + A_2\beta^4 + A_3\beta^8 \\
b &= B_0\beta + B_1\beta^2 + B_2\beta^4 + B_3\beta^8
\end{aligned}
$$

- The product c would be

$$
\begin{aligned}
c = \; & A_0B_0\beta^2 + A_0B_1\beta^3 + A_0B_2\beta^5 + A_0B_3\beta^9 \\
& A_1B_0\beta^3 + A_1B_1\beta^4 + A_1B_2\beta^6 + A_1B_3\beta^{10} \\
& A_2B_0\beta^5 + A_2B_1\beta^6 + A_2B_2\beta^8 + A_2B_3\beta^{12} \\
& A_3B_0\beta^9 + A_3B_1\beta^{10} + A_3B_2\beta^{12} + A_3B_3\beta^{16}
\end{aligned}
$$

Normal Basis Multiplication in $GF(2^4)$

- Using the representations of all powers of β in normal basis, we obtain

$$\begin{aligned} c \;=\;& A_0B_0\beta^2 + A_0B_1\beta^8 + A_0B_2(\beta^8+\beta^4+\beta^2+\beta) + A_0B_3\beta^4 \\ & A_1B_0\beta^8 + A_1B_1\beta^4 + A_1B_2\beta + A_1B_3(\beta^8+\beta^4+\beta^2+\beta) \\ & A_2B_0(\beta^8+\beta^4+\beta^2+\beta) + A_2B_1\beta + A_2B_2\beta^8 + A_2B_3\beta^2 \\ & A_3B_0\beta^4 + A_3B_1(\beta^8+\beta^4+\beta^2+\beta) + A_3B_2\beta^2 + A_3B_3\beta \end{aligned}$$

- By grouping the powers of β, we obtain

$$\begin{aligned} c \;&=\; (A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3)\beta + \\ &=\; (A_0B_0 + A_0B_2 + A_1B_3 + A_2B_0 + A_2B_3 + A_3B_1 + A_3B_2)\beta^2 + \\ &=\; (A_0B_2 + A_0B_3 + A_1B_1 + A_1B_3 + A_2B_0 + A_3B_0 + A_3B_1)\beta^4 + \\ &=\; (A_0B_1 + A_0B_2 + A_1B_0 + A_1B_3 + A_2B_0 + A_2B_2 + A_3B_1)\beta^8 \\ &=\; C_0\beta + C_1\beta^2 + C_2\beta^4 + C_3\beta^8 \end{aligned}$$

Normal Basis Multiplication in GF(2^4)

- This expression gives the bits of the product c in terms of the bits of a and b, expressed in the normal basis

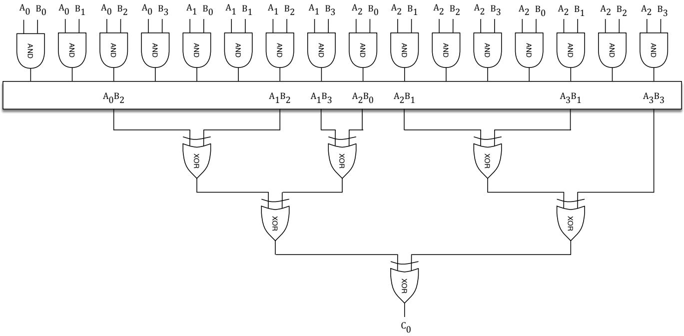
$$
\begin{aligned}
C_0 &= A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3 \\
C_1 &= A_0B_0 + A_0B_2 + A_1B_3 + A_2B_0 + A_2B_3 + A_3B_1 + A_3B_2 \\
C_2 &= A_0B_2 + A_0B_3 + A_1B_1 + A_1B_3 + A_2B_0 + A_3B_0 + A_3B_1 \\
C_3 &= A_0B_1 + A_0B_2 + A_1B_0 + A_1B_3 + A_2B_0 + A_2B_2 + A_3B_1
\end{aligned}
$$

- The above formulas imply we need 16 2-input AND gates to obtain the terms A_iB_j for $i, j = 0, 1, 2, 3$
- We then need 24 XOR gates to compute the product bits C_0, C_1, C_2, C_3, in other words, 6 XOR gates for each C_i
- The normal basis multiplication operation is more complicated than the squaring, which was just a left rotation of the bits

Normal Basis Multiplication in GF(2^4)

- Interestingly there is more structure in the normal basis multiplication than this formulation makes it obvious
- First we design a circuit consisting of AND and XOR gates for computing the first bit of the product C_0

$$C_0 = A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3$$



Normal Basis Multiplication in GF(2^4)

- Consider the normal basis expressions for C_0 and C_1 given as

$$\begin{aligned} C_0 &= A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3 \\ C_1 &= A_0B_0 + A_0B_2 + A_1B_3 + A_2B_0 + A_2B_3 + A_3B_1 + A_3B_2 \end{aligned}$$

- Now we rearrange the terms in C_1 so that the term A_iB_j in C_0 is aligned with the term $A_{i+1 \pmod 4}B_{j+1 \pmod 4}$ in C_1
- For example, the below the term A_0B_2 in C_0, we place the term A_1B_3
- Similarly, the below the term A_3B_3 in C_0, we place the term A_0B_0

$$\begin{aligned} C_0 &= A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3 \\ C_1 &= A_1B_3 + A_2B_3 + A_2B_0 + A_3B_1 + A_3B_2 + A_0B_2 + A_0B_0 \end{aligned}$$

- **Miraculously** this alignment works for C_0 and C_1
- All terms in C_1 are placed below their corresponding terms in C_0

Normal Basis Multiplication in GF(2^4)

- It also works for C_1 and C_2

$$\begin{aligned} C_1 &= A_1B_3 + A_2B_3 + A_2B_0 + A_3B_1 + A_3B_2 + A_0B_2 + A_0B_0 \\ C_2 &= A_2B_0 + A_3B_0 + A_3B_1 + A_0B_2 + A_0B_3 + A_1B_3 + A_1B_1 \end{aligned}$$

- It also works for C_2 and C_3

$$\begin{aligned} C_2 &= A_2B_0 + A_3B_0 + A_3B_1 + A_0B_2 + A_0B_3 + A_1B_3 + A_1B_1 \\ C_3 &= A_3B_1 + A_0B_1 + A_0B_2 + A_1B_3 + A_1B_0 + A_2B_0 + A_2B_2 \end{aligned}$$

- In fact this is a property of the normal basis

Normal Basis Multiplication in GF(2^4)

- The rearranged set of equations for the product terms are

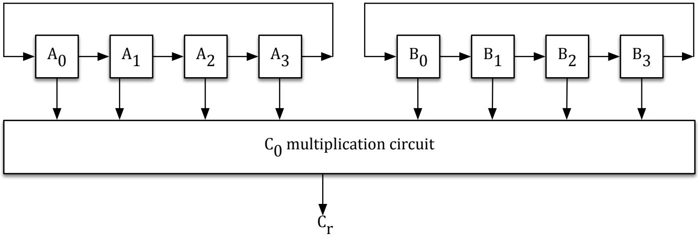
$$\begin{aligned}
C_0 &= A_0B_2 + A_1B_2 + A_1B_3 + A_2B_0 + A_2B_1 + A_3B_1 + A_3B_3 \\
C_1 &= A_1B_3 + A_2B_3 + A_2B_0 + A_3B_1 + A_3B_2 + A_0B_2 + A_0B_0 \\
C_2 &= A_2B_0 + A_3B_0 + A_3B_1 + A_0B_2 + A_0B_3 + A_1B_3 + A_1B_1 \\
C_3 &= A_3B_1 + A_0B_1 + A_0B_2 + A_1B_3 + A_1B_0 + A_2B_0 + A_2B_2
\end{aligned}$$

- The implication of this rearrangement is that the circuit for computing C_0 can be used for computing C_r for $r = 1, 2, 3$
- The rearrangement and realignment algorithm is determined by the property that, for $r = 1, 2, 3$

A_iB_j in C_0 **is aligned with** $A_{i+r \bmod 4}\ B_{j+r \bmod 4}$ in C_r

Normal Basis Multiplication in GF(2^4)

- Suppose the input bits are arranged as $(A_0A_1A_2A_3\ B_0B_1B_2B_3)$ and applied to the C_0 multiplication circuit in order to compute C_0
- If we now apply the input bits as $(A_3A_0A_1A_2\ B_3B_0B_1B_2)$, we will be computing C_1 using the same circuit
- This represents a right rotation of the input bits applied to the circuit
- By right shifting and applying the input vectors 4 times, all product bits in increasing index are computed using the same C_0 circuit



Optimal Normal Basis Multiplication

- There is another remarkable property of the normal bases
- For a given k value there may be a basis for which the multiplication requires minimum number of XOR gates
- The number of XOR gates for computing the first product term C_0 for GF(2^4) was 6, which is one less than the number of terms in the normal representation of C_0
- Since we are using the same circuit (whether sequentially or in parallel), the number of XOR gates for computing any product bit is the same

Optimal Normal Basis Multiplication in GF(2^4)

Theorem

The minimum number of terms in the normal representation of the product C_0 for GF(2^k) is given as $2k-1$, and the bases with this property are called optimal normal bases.

- The normal basis $\beta = \alpha^3$ for GF(2^4) had $2 \cdot 4 - 1 = 7$ terms in the expression for C_0 is an optimal normal basis
- The fundamental construction method of optimal normal bases was given by Mullin, Onyszchuk, Vanstone and Wilson in 1988
- They proved the existence of 2 types optimal normal bases
- The uniqueness of these bases was proven by Gao and Lenstra in 1991

Optimal Normal Bases for GF(2^k)

- While there is a normal basis for GF(2^k) for every k, an optimal normal basis exists for only some values of k
- For example, for $k \in [2, 2000]$ there are a total of 430 values of k for which an optimal normal basis Type 1 or Type 2 exists

k	2	3	4	5	6	9	10	11	12	14	18	23
Type	1, 2	2	1	2	2	2	1	2	1	2	1, 2	2
k	251	254	261	268	270	273	278	281	292	293	299	303
Type	2	2	2	1	2	2	2	2	1	2	2	2
k	508	509	515	519	522	530	531	540	543	545	546	554
Type	1	2	2	2	1	2	2	1	2	2	1	2