1 SubBytes using Composite Field Arithmetic: an alternative description

The aim of this section is to bridge the gap between composite field mathematics and the corresponding hardware implementation. To date the authors believe this has not been covered in sufficient detail to permit easy reproduction or improvement of reported results.

There are a number of existing works [Paar, Satoh, Ward, Matrovito] on using the equivalent composite field arithmetic, $GF(2^n)^m$, to reduce the complexity of the required logic for a given operation in a binary extension field, $GF(2^k)$.

Rudra et al [Rudra] used composite field arithmetic in $GF(2^4)^2$ based on the theory in Paar [Paar]. However, in Satoh et al [Satoh] the composite fields used were $GF((2^2)^2)^2$ for which the construction was not given. This was later pipelined by Zhang et al [Zhang] to yield a design with a throughput of 21.56 Gbps.

1.1 Representation

In such a set of composite fields the logic will be at a minimum when the simplest primitive polynomials are used; these are known to be trinomials. Each field is defined in terms of a sub field. To avoid confusion a different letter (x, y or z) is used for each field's polynomial $P_x(x)$, $P_y(y)$ and $P_z(z)$. In electronics, we will use a set of binary values ($a_0..a_N$) to represent field values. Thus the construction can be started with the information contained in Table 1.

Field	Bit Representation	Polynomial
$GF(2)$	a_0	n/a
$GF(2^2)$	$a_1x + a_0$	$P_x(x)=x^2+x+1$
$GF(2^2)^2$	$a_3xy+a_2y + a_1x+a_0$	$P_y(y)=y^2+y+\phi \quad \phi \in GF(2^2)$
$GF((2^2)^2)^2$	$a_7xyz+a_6yz+a_5xz+a_4z + a_3xy+a_2y+a_1x+a_0$	$P_z(z)=z^2+z+\lambda \quad \lambda \in GF(2^2)^2$

Table 1. Polynomial representation in composite fields

As an alternative, the column headings for the binary representation of the composite fields may be thought of as in Table 2. It should be remembered that in Galois Field arithmetic any higher powers of the irreducible variable (eg y^2) will be reduced according to the primitive polynomial (eg $y^2=y+\phi$).

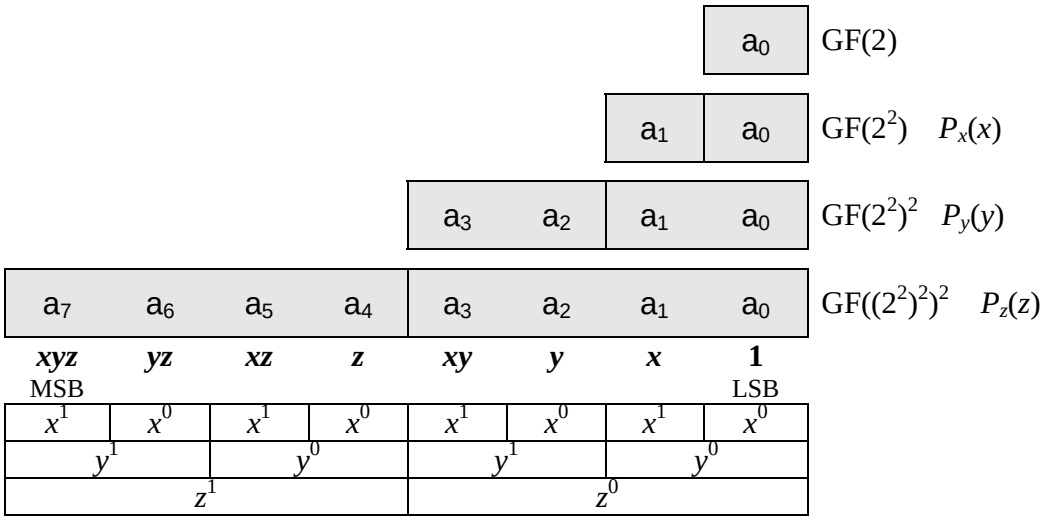

Table 2. Binary representation in composite fields

1.2 Multiplicative Inverse

First multiplicative inversion in composite field $GF(2^n)^m$ can be defined recursively in terms of a subfield $GF(2^n)$ and its extension $GF(2^n)^m$ with primitive polynomial $P_{nm}(x)$. Such a proof is given in Zhang [Zhang]. Briefly for a value in the binary extension field, $T(x)$, the inverse $T^{-1}(x)$ is sought.

$$T(x) = t_H x + t_L \quad t_H, t_L \in GF2^n$$

Let $P_{nm}(x)$ be a primitive trinomial,

$$P_{nm}(x) = x^2 + x + \lambda \qquad \lambda \in GF2^n$$

From Euclid, for finite field modulo arithmetic, the product of a polynomial $A(x)$ and a primitive polynomial P_{nm} is zero and the product of a polynomial $T(x)$ and its inverse $B(x)$ is unity.

$$A(x)P_{nm}(x) + B(x)T(x) = 1 \quad where\ B(x) \equiv T^{-1}(x) \tag{1}$$

This can be rearranged into the form

$$P_{nm}(x) = Q(x)T(x) + R(x) \tag{2}$$

and the quotient, $Q(x)$ and remainder $R(x)$ are found by long division of $P_{\mathrm{nm}}(x)$ by $T(x)$.

$$Q(x) = t_H{}^{-1}x \quad + \quad (1 + t_L t_H{}^{-1})t_H{}^{-1}$$

$$R(x) = \lambda + (1 + t_L t_H{}^{-1})t_L t_H{}^{-1}$$

These may be substituted back into equation (2) to give

$$t_H{}^2 P_{nm}(x) = (t_H x + t_L + t_H)T(x) + t_H{}^2\lambda + t_L t_H + t_L{}^2$$

Let $\Phi = \left(t_H{}^2\lambda + t_L(t_L + t_H)\right)^{-1}$

then $\left[\Phi t_H{}^2\right]P(x) + \left[\Phi(t_H x + t_L + t_H)\right]T(x) = 1$

Comparing this with equation (1) we have

$$T^{-1}(x) = t_H \Phi x + (t_L + t_H)\Phi \qquad where \quad t_{H,}t_L, \lambda, \Phi \in GF2^n \tag{3}$$

Equation (3) may be used to define composite field multiplicative inversion in terms of arithmetic in the sub field. This definition includes inversion (in the Φ term) thus is applied recursively down to the base field. Figure 1 depicts the mathematics in circuit terms.

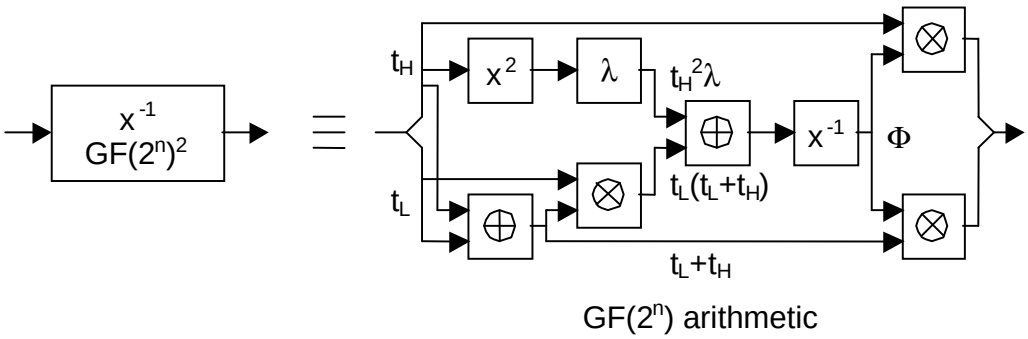

Figure 1. Composite field multiplicative inverse

The circuit requires a number of operations all in GF(2^n) arithmetic using the polynomial $P_{nm}(x)=x^2+x+\lambda$. These are addition, $\oplus$, squaring, x^2, multiplication, $\otimes$, constant multiplication by λ and inversion, x^{-1}. In this case addition is simply the *XOR* of the corresponding bits. The inversion may be implemented using the same basic circuit as above with reduced 'n' and 'm' until GF(2) is reached. The remaining operations (x^2, λ and $\otimes$) are all variations of composite field multiplication. The theory is addressed below.

1.3 Multiplication

Composite field multiplication in the field GF(2^n)^m, with primitive trinomial, $P_{nm}(x)=x^2+x+\lambda$ may be defined in terms of GF(2^n) arithmetic.

Let the two values to be multiplied be $A(x)$ and $B(x)$ then in terms of the subfield we can write:

$$A(x) = A_H x + A_L \quad and \quad B(x) = B_H x + B_L$$

Then the product, $AB(x)$ is

$$AB(x) = x(A_H B_H + A_L B_H + A_H B_L) + \lambda A_H B_H + A_L B_L$$

where $A_H, A_L, B_H, B_L, \lambda \in GF2^n$

This equation permits the construction of GF(2^n)^m multipliers in terms of only multiplication and addition in GF(2^n), as with the previous inversion example, for n>1 this may be repeated for successively smaller subfields until the base field of GF(2) is reached. In GF(2) multiplication this is simply *AND* and addition *XOR*.

A further optimisation was proposed by Mastrovito [Mastrovito] which is applicable where the complexity of the multiplication operation is more than the addition (i.e. composite field above the base field). The number of multiplications is reduced by one at the expense of an addition. The Mastrovito composite field multiplier may be represented by the following equation:

$$AB(x) = x\big((A_L + A_H)(B_L + B_H) + A_L B_L\big) + \lambda A_H B_H + A_L B_L$$

The composite field Mastrovito multiplier (for a primitive trinomial) may be expressed in circuit terms as in Figure 3. The circuit only includes operations in the sub field GF(2^n) and may be applied recursively using sub fields of lesser degree to define the multipliers until the base field GF(2) is reached. For any given value of the constant λ, the constant multiplier shown in the circuit may be simplified to a circuit only containing *XOR* gates.

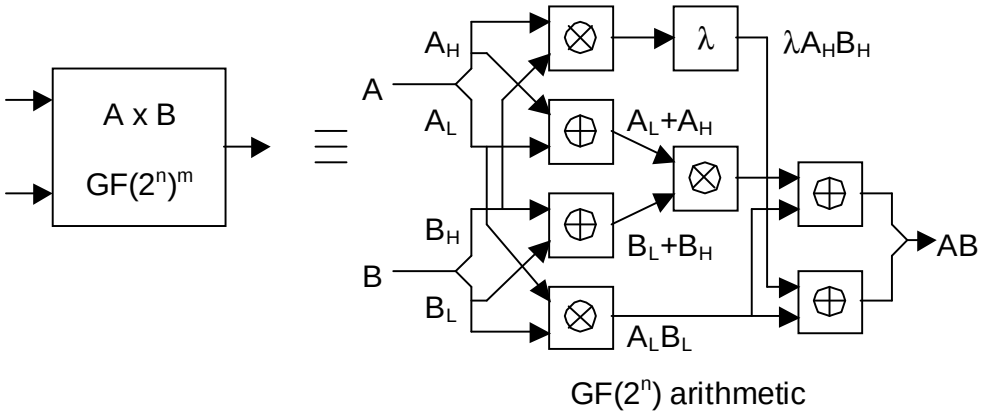

Figure 2. Composite field multiplication

1.4 Field conversion

In order to apply the reduced complexity, in terms of number of gates, offered by composite field arithmetic the respective field polynomials need to be determined together with a suitable isomorphism such that the composite field inversion in GF($((2^2)^2)^2$ modulo $P(x)$, $P(y)$, $P(z)$ and extended field inversion in GF(2^8) modulo $P(\omega)$ are equivalent (Figure 4).

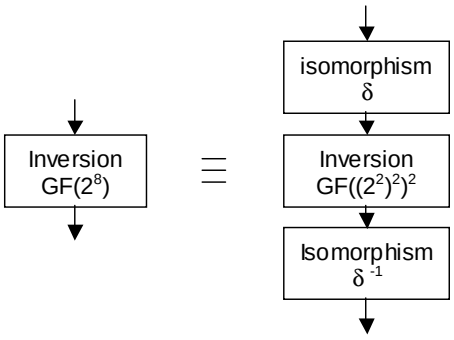

Figure 4. Extended field and composite field equivalence

The procedure for the construction of such a set of fields and corresponding transformation is documented in Paar [Paar]. The isomorphism may be considered as the matrix multiplication of the 8-by-8 binary matrix, δ, with the 8-bit value in GF(2^8) the result is an 8-bit value in GF(($(2^2)^2$)^2) which will have equivalent inverse.

From Paar [Paar] the columns of the conversion matrix are powers of a primitive element, ε in GF($(2^2)^2$)^2) starting with ε^0={01h}. The inverse isomorphism, δ^{-1}, can be found from matrix inversion of δ. The values of ε, ϕ and λ together with any primitive polynomial in GF(2^8) (used during the search process) can be conveniently determined using a computer based exhaustive search technique. The AES polynomial is only irreducible (not primitive) thus is not guaranteed to generate all possible elements so a primitive conversion polynomial, $P_\delta(\omega)$ is used.

Table 3 summarises the values required for the field construction. These values produce the circuits described by Satoh [Satoh] and Zhang [Zhang] unfortunately some crucial details were omitted from both the original descriptions which hinder understanding. Such information is important when considering whether further improvements can be made.

Field	Polynomial	Primitive Element Used
GF(2^2)	$P_x(x)=x^2+x+1$	$\alpha = x = \{10\}_2$
GF(2^2)^2	$P_y(y)=y^2+y+\phi$, $\phi=x=\{10\}_2$	$\beta = x+y = \{0110\}_2$
GF(($(2^2)^2$)^2)	$P_z(z)=z^2+z+\lambda$, $\lambda=xy+y=\{1100\}_2$	$\gamma = \{16,26,96,a6 \text{ or } f6\}_{16}$
GF(2^8) (conversion)	$P_\delta(\omega) = \omega^8+\omega^4+\omega^3+\omega^2+1$	$\varepsilon = xy+z+xy+y+x+1$ $= \omega^6+\omega^4+\omega^3+\omega^2+\omega+1$ $= \{0101\ 1111\}_2$
GF(2^8)	$P(\omega) = \omega^8+\omega^4+\omega^3+\omega+1$ (AES)	

Table 3. Composite field construction

The resulting isomorphism transform matrix and its inverse are:

$$\delta = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad \delta^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The *SubBytes* process also required the application of a given affine transform (defined in FIPS-197) for both its forward and reverse forms. The circuit in figure 5 shows a combined design for forward and reverse SubBytes, the multiplexers selecting between forward and reverse transforms.

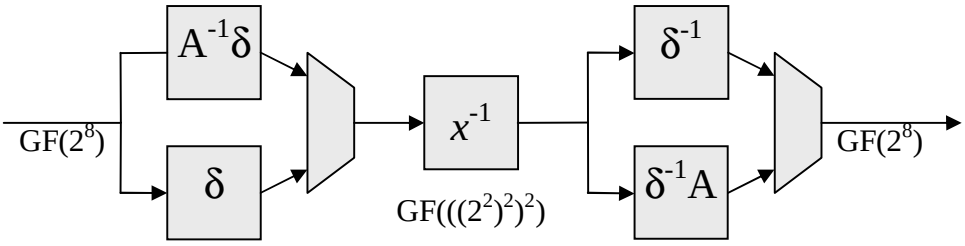

Figure 5. Combined SubBytes circuit using composite field arithmetic.

The isomorphism and affine transforms may be combined into a single transform [Satoh]. This results in the following matrices:

$$\delta^{-1}A(x) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad A^{-1}\delta(x) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$