A Very Compact Rijndael S-box

Article · May 2005

CITATIONS
95

READS
3,669

1 author:

David Canright
Naval Postgraduate School
35 PUBLICATIONS **1,262** CITATIONS

SEE PROFILE

NPS-MA-05-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

A Very Compact Rijndael S-box

by

D. Canright

17 May 2005
(revised)

Prepared for: National Security Agency

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RDML Patrick W. Dunne, USN
President

Richard Elster
Provost

This report was prepared for the National Security Agency
and funded by the National Security Agency.

This report was prepared by:

_____
David Canright
Associate Professor of Mathematics

Reviewed by:

_____
Clyde Scandrett, Chairman
Department of Applied Mathematics

Released by:

_____
Leonard A. Ferrari, Ph.D.
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form approved

OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	17 May 2005	Technical Report 6 July - 24 September 2004

4. TITLE AND SUBTITLE

A Very Compact Rijndael S-box

5. FUNDING

6. AUTHOR(S)

D. Canright

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-MA-04-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Security Agency
9800 Savage Road, Ste. 6538
Fort Meade, MD 20755-6538

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words.)

One key step in the Advanced Encryption Standard (AES), or Rijndael, algorithm is called the "S-box", the only nonlinear step in each round of encryption/decryption. A wide variety of implementations of AES have been proposed, for various desiderata, that effect the S-box in various ways. In particular, the most compact implementation to date of Satoh et al. performs the 8-bit Galois field inversion of the S-box using subfields of 4 bits and of 2 bits. This work describes a refinement of this approach that minimizes the circuitry, and hence the chip area, required for the S-box. While Satoh used polynomial bases at each level, we consider also normal bases, with arithmetic optimizations; altogether, 432 different cases were considered. The isomorphism bit matrices are fully optimized, improving on the "greedy algorithm." The best case reduces the number of gates in the S-box by 20%. This decrease in chip area could be important for area-limited hardware implementations, e.g., smart cards. And for applications using larger chips, this approach could allow more copies of the S-box, for parallelism and/or pipelining in non-feedback modes of AES.

14. SUBJECT TERMS	15. NUMBER OF PAGES 71
cryptography, encryption, AES, Rijndael, Galois fields, FPGA, ASIC	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

A Very Compact Rijndael S-box

D. Canright
Applied Mathematics Dept., Code MA/Ca
Naval Postgraduate School
Monterey, CA 93943

(revised) May 17, 2005

Abstract

One key step in the Advanced Encryption Standard (AES), or Rijndael, algorithm is called the "S-box", the only nonlinear step in each round of encryption/decryption. A wide variety of implementations of AES have been proposed, for various desiderata, that effect the S-box in various ways. In particular, the most compact implementation to date of Satoh et al.[12] performs the 8-bit Galois field inversion of the S-box using subfields of 4 bits and of 2 bits. This work describes a refinement of this approach that minimizes the circuitry, and hence the chip area, required for the S-box. While Satoh[12] used polynomial bases at each level, we consider also normal bases, with arithmetic optimizations; altogether, 432 different cases were considered. The isomorphism bit matrices are fully optimized, improving on the "greedy algorithm." The best case reduces the number of gates in the S-box by 20%. This decrease in chip area could be important for area-limited hardware implementations, e.g., smart cards. And for applications using larger chips, this approach could allow more copies of the S-box, for parallelism and/or pipelining in non-feedback modes of AES.

1 Introduction

The Advanced Encryption Standard (AES) was specified in 2001 by the National Institute of Standards and Technology [9]. The purpose is to provide a standard algorithm for encryption, strong enough to keep U.S. government documents secure for at least the next 20 years. The earlier Data Encryption Standard (DES) had been rendered insecure by advances in computing power, and was effectively replaced by triple-DES. Now AES will largely replace triple-DES for government use, and will likely become widely adopted for a variety of encryption needs, such as secure transactions via the Internet. As Secretary of Commerce Norman Y. Mineta put it in announcing AES, "...this standard will serve as a critical computer security tool supporting the rapid growth of electronic commerce. This is a very significant step toward creating a more secure digital economy. It will allow e-commerce and e-government to flourish safely, creating new opportunities for all Americans."[7]

A wide variety of approaches to implementing AES have appeared, to satisfy the varying criteria of different applications. Some approaches seek to maximize throughput, e.g., [5], [14]

and [2]; others minimize power consumption, e.g., [6]; and yet others minimize circuitry, e.g., [11], [12], [15], and [1]. For the latter goal, Rijmen[10] suggested using subfield arithmetic in the crucial step of computing an inverse in the Galois Field of 256 elements—essentially expressing an 8-bit calculation in terms of 4-bit ones. This idea was further extended by Satoh et al.[12], breaking up the 4-bit calculations into 2-bit ones, which resulted in the smallest AES circuit to date.

The current work improves on the compact implementation of [12] in the following ways. Many (432) choices of representation (isomorphisms) were compared, and the most compact turns out to use a normal basis for each subfield ([12] uses a polynomial basis for each subfield). And while [12] used the "greedy algorithm" to reduce the number of gates in the bit matrices required in changing representations, here each bit matrix is fully optimized, resulting in the minimum number of gates. These various refinements result in an S-box circuit that is 20% smaller, a significant improvement.

The AES algorithm, also called the Rijndael algorithm, is a symmetric encryption algorithm, meaning encryption and decryption are performed by essentially the same steps. It is a block cipher, where the data is encrypted/decrypted in blocks of 128 bits. (The original Rijndael algorithm allows other block sizes, but the Standard only permits 128-bit blocks.) Each data block is modified by several "rounds" of processing, where each round involves four steps. Three different key sizes are allowed: 128 bits, 192 bits, or 256 bits, and the corresponding number of rounds for each is 10 rounds, 12 rounds, or 14 rounds, respectively. From the original key, a different "round key" is computed for each of these rounds. For simplicity, the discussion below will use a key length of 128 bits and hence 10 rounds.

There are several different modes in which AES can be used [8]. For some of these, such as Cipher Block Chaining (CBC), the result of encrypting one block is used in encrypting the next. These are called feedback modes, and the feedback effectively precludes pipelining (simultaneous processing of several blocks in the "pipeline"). Other modes, such as the "Electronic Code Book" mode or "Counter" modes, do not require feedback. These non-feedback modes may be pipelined for greater throughput.

The four steps in each round of encryption, in order, are called *SubBytes* (byte substitution), *ShiftRows*, *MixColumns*, and *AddRoundKey*. Before the first round, the input block is processed by AddRoundKey; one could consider this round number zero. Also, the last round, number ten, skips the MixColumns step. Otherwise, all rounds are the same, except each uses a different round key, and the output of one round becomes the input for the next. (For decryption, the mathematical inverse of each step is used, in reverse order; certain manipulations allow this to appear like the same steps as encryption with certain constants changed.)

Of these four steps, three of them (*ShiftRows*, *MixColumns*, and *AddRoundKey*) are *linear*, in the sense that the output 128-bit block for such steps is just the linear combination (bitwise, modulo 2) of the outputs for each separate input bit. These three steps are all easy to implement by direct calculation in software or hardware.

The single *nonlinear* step is the *SubBytes* (byte substitution) step, where each byte (8 bits) of the input is replaced by the result of applying the "S-box" function to that byte. This nonlinear function involves finding the inverse of the 8-bit number, considered as an element of the Galois field $GF(2^8)$. This is not a simple calculation, and so many current implementations use a table of the S-box function output; the input byte is an index into

the table to find the output. This table look-up method is fast and easy to implement.

But for hardware implementations of AES, there is one drawback of the table look-up approach to the S-box function: each copy of the table requires 256 bytes of storage, along with the circuitry to address the table and fetch the results. Each of the 16 bytes in a block can go through the S-box function independently, and so could be processed in parallel for the byte substitution step. This then effectively requires 16 copies of the S-box table for one round. To fully pipeline the encryption would entail "unrolling" the loop of 10 rounds into 10 sequential copies of the round calculation. This would require 160 copies of the S-box table, a significant allocation of hardware resources.

In contrast, this work describes a direct calculation of the S-box function using sub-field arithmetic, similar to [12]. While the calculation is complicated to describe, the advantage is that the circuitry required to implement this in hardware is relatively simple, in terms of the number of logic gates required. This type of S-box implementation is significantly smaller (less area) than the table it replaces, especially with the optimizations in this work. Furthermore, when chip area is limited, this compact implementation may allow parallelism in each round and/or unrolling of the round loop, for a significant gain in speed.

The rest of the paper describes the algorithm in detail. Section 2 describes some basics of Galois field arithmetic and representations, essential to the algorithm. The basic idea of the algorithm is explained in Section 3. Section 4 discusses ways to optimize the calculation, Section 5 describes the choices of representation, and Section 6 gives the detailed formulas of the algorithm. Finally, Section 7 summarizes the work.

2 Galois Fields $GF(2^n)$

Finite fields, or Galois fields, are important in many applications, such as error-correcting codes[4], and have been studied extensively (one good reference is [3]). Here we give only a brief, informal introduction to the properties necessary for the AES algorithm.

A *field* is a set F of elements with two binary operations, say $\oplus$ and $\otimes$. We will call these addition and multiplication, and will sometimes use the standard notation $a + b$ and ab instead of $a \oplus b$ and $a \otimes b$, for simplicity. These operations must satisfy certain properties (here a, b, c represent arbitrary elements of F):

1. the set is *closed* with respect to both operations:
   (a) $a \oplus b \in F$
   (b) $a \otimes b \in F$
2. both operations are *associative*:
   (a) $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
   (b) $(a \otimes b) \otimes c = a \otimes (b \otimes c)$
3. both operations are *commutative*:
   (a) $a \oplus b = b \oplus a$

(b) $a \otimes b = b \otimes a$

4. the operations obey the *distributive* law: $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$

5. each operation has an *identity* (call the identities 0 and 1):

   (a) $a \oplus 0 = a$

   (b) $a \otimes 1 = a$

6. each element a has an *additive inverse* (say q): $a \oplus q = 0$ (this defines subtraction; the standard notation for the additive inverse of a is $-a$)

7. each nonzero element $a \neq 0$ has a *multiplicative inverse* (say r): $a \otimes r = 1$ (this defines division; the standard notation for the multiplicative inverse of a is a^{-1})

Familiar examples are the field of rational numbers, the field of real numbers, and the field of complex numbers. If a subset of a field is itself a field, using the same operations, then it is called a *subfield*. For example, the rational numbers is a subfield of the real numbers.

If a field has only a finite number of elements, it is a *finite field*. But given some finite set, it is not always possible to define two operations with the above properties; it is only possible if the number of elements in the set is of the form p^n where p is a prime number and n is a positive integer. Then p^n is called the *order* of the field and p is called the *characteristic* of the field. So there is no field of 6 elements, for example, but there is a field of 7 elements and a field of 8 ($= 2^3$) elements. Given a set of p^n elements there may be more than one way to define the operations to produce a field, but these different ways give fields that are *isomorphic*: by changing the names we can change one field into the other—the structure remains the same. So in this sense there is only one finite field for a given number of elements p^n; we call this the Galois Field $GF(p^n)$. (We will also use the notation of [3] for this field: $\mathbb{F}_k$, where $k = p^n$.) If a positive integer m is a factor of n, then $GF(p^m)$ is a subfield of $GF(p^n)$.

The simplest example is $GF(2) = \{0, 1\}$ with the usual addition and multiplication except $1 \oplus 1 = 0$; this is also called arithmetic *modulo 2*. Note that in this field, each element is its own additive inverse, so subtraction is the same as addition. This is true for all fields $GF(2^k)$ of characteristic 2.

Another example that will be important later is $GF(2^2)$, whose elements will be labeled $\{0, 1, \Omega, \Psi\}$. The operations are defined by the tables below:

$\oplus$	0	1	Ω	Ψ
0	0	1	Ω	Ψ
1	1	0	Ψ	Ω
Ω	Ω	Ψ	0	1
Ψ	Ψ	Ω	1	0

$\otimes$	0	1	Ω	Ψ
0	0	0	0	0
1	0	1	Ω	Ψ
Ω	0	Ω	Ψ	1
Ψ	0	Ψ	1	Ω

Note that if we swap the names Ω and Ψ everywhere, we get exactly the same operations, i.e., the same field. Also note that $GF(2^2)$ contains the subfield $GF(2) = \{0, 1\}$.

There are several different ways to look at a Galois field. An element a of $GF(p^n)$ is called *primitive* if all its powers are different: $a^0 \neq a^1 \neq a^2 \neq \cdots \neq a^{p^n-2}$. (For any nonzero

element b then $b^{p^n-1} = 1$; for any element b then $b^{p^n} = b$.) Hence the powers of a primitive element give all the nonzero elements of $GF(p^n)$. Every finite field has at least one primitive element, so one way to look at the field is in terms of powers of that element. For example, in $GF(2^2)$, Ω is a primitive element: $1 = \Omega^0, \Omega = \Omega^1, \Psi = \Omega^2$. This viewpoint makes multiplication easy: add the exponents modulo $p^n - 1$. But then addition is less obvious.

Another viewpoint involves polynomials, in some variable x, with coefficients in $GF(p)$; these are called polynomials *over* $GF(p)$. Each element of $GF(p^n)$ can be considered a polynomial over $GF(p)$, of degree less than n. Then addition just means adding the coefficients modulo p. Multiplication must be done modulo some specified polynomial $q(x)$, of degree n, with leading coefficient equal to 1; also $q(x)$ must be *irreducible*, which means it is not the product of two polynomials of lower order.

For example, in $GF(2^2)$ the only choice for $q(x)$ is $x^2 + x + 1$ (because the others factor: $x^2 = x*x$, $x^2+x = x*(x+1)$, $x^2+1 = (x+1)*(x+1)$; remember the coefficient arithmetic is modulo 2). Then we could think of $GF(2^2)$ as $\{0, 1, x, x+1\}$ where $x \otimes x = (x^2 \text{ modulo } q) = x^2 \oplus (x^2 + x + 1) = x + 1$, and similarly $x \otimes (x + 1) = (x^2 + x) \oplus (x^2 + x + 1) = 1$ and $(x + 1) \otimes (x + 1) = (x^2 + 1) \oplus (x^2 + x + 1) = x$.

This polynomial viewpoint makes more sense if we think of the variable x as being a root of the polynomial, so $q(x) = 0$. Then adding or subtracting multiples of $q(x)$ is just adding zero. In the first representation of $GF(2^2)$, note that $\Omega^2 \oplus (\Omega \oplus 1) = \Psi \oplus \Psi = 0$, so we could identify $x = \Omega$. Alternatively, we could identify $x = \Psi$ (switching the names as before), the other root.

Another viewpoint is that the field $GF(p^n)$ is a vector space of dimension n, with vector addition $\oplus$ and multiplication by scalars in $GF(p)$ (i.e., modulo p). (The vector viewpoint is convenient for choosing a representation, but does not fully reflect the multiplication operation $\otimes$.) Then any n linearly independent elements $\{b_1, b_2, \ldots, b_n\}$ of $GF(p^n)$ gives a *basis*, and we can indicate any element a by its list of coefficients with respect to this basis: if $a = c_1 \otimes b_1 \oplus c_2 \otimes b_2 \oplus \ldots \oplus c_n \otimes b_n$ (with each $c_i \in GF(p)$) then a is represented by the list of numbers $[c_1, c_2, \ldots, c_n]$. For small p this list commonly is written as digits in positional notation: $c_1 c_2 \ldots c_n$.

For example, the polynomial viewpoint for $GF(2^2)$, with $x = \Omega$, corresponds to using the ordered basis $[\Omega^1, \Omega^0]$; this is called a polynomial basis. Using this basis: $0 = 0\Omega^1 + 0\Omega^0 \equiv 00$, $1 = 0\Omega^1 + 1\Omega^0 \equiv 01$, $\Omega = 1\Omega^1 + 0\Omega^0 \equiv 10$, $\Psi = 1\Omega^1 + 1\Omega^0 \equiv 11$. This defines a field of 2-bit binary numbers (where $\oplus$ is bitwise exclusive-or), where for example $11 \otimes 11 = 10$.

But different choices of basis are also possible. Another type of basis with convenient properties is called a *normal* basis, of the form $\{b^{p^0}, b^{p^1}, \ldots, b^{p^{n-1}}\}$, where the element b of $GF(p^n)$ must be chosen to make that set of powers linearly independent. (One nice property is that an isomorphism [name change] on the field has the same effect as rotating this list of basis elements.)

Using the ordered normal basis $[\Omega^{2^1}, \Omega^{2^0}] = [\Psi, \Omega]$ for $GF(2^2)$ gives the correspondence $0 = 0\Psi + 0\Omega \equiv 00$, $1 = 1\Psi + 1\Omega \equiv 11$, $\Omega = 0\Psi + 1\Omega \equiv 01$, $\Psi = 1\Psi + 0\Omega \equiv 10$. This gives a different 2-bit representation of $GF(2^2)$; addition $\oplus$ is still bitwise exclusive-or, but now for example $11 \otimes 11 = 11$. So in one sense this is a different field, but it has exactly the same structure as the previous version, only the names have been changed to confuse the innocent.

The polynomial representation idea can be generalized. For any finite field F (of char-

acteristic p) containing a subfield S, where S is of order $r = p^j$ and F is of order $r^k = p^{jk}$, then the elements of F can be represented as polynomials of degree less than k, with coefficients in S (i.e., polynomials *over* S). We notate this view of the field as F/S (read as F "over" S). Again, addition just means adding the coefficients in S, and multiplication is done modulo some polynomial $q(x)$, of degree k. The coefficients of $q(x)$ also belong to S, with the leading coefficient equal to 1, and $q(x)$ must be irreducible over S (no element of S is a root). For example, the elements of $GF(5^6)$ can be represented as polynomials of the form $c_2x^2 + c_1x + c_0$, with all the $c_i \in GF(5^2)$, modulo the polynomial $q(x) = x^3 + x^2 + x + 3$, which is irreducible over $GF(5^2)$.

Since the names of the elements of $GF(p^n)$ change with choice of representation, we might wonder if the elements have certain properties that are independent of representation, a sort of identification. One such property is the *minimal polynomial* (over $GF(p)$) of a given element a. This is the irreducible polynomial of smallest degree, with coefficients in $GF(p)$ and leading coefficient $= 1$, having a as a root. The degree m of the minimal polynomial is always $\leq n$, and that minimal polynomial has m distinct roots in $GF(p^n)$. Elements with the same minimal polynomial are called *conjugates*; if one of them is a then the m conjugates are $\{a, a^p, a^{p^2}, \ldots, a^{p^{m-1}}\}$. Each isomorphism of $GF(p^n)$ corresponds to replacing each element b by b^{p^k} (for some integer k), and so in effect rotates each set of conjugates. For any primitive element, the minimal polynomial is called a primitive polynomial and has degree n. (Note that a normal basis is a set of n distinct conjugates.) In $GF(2^2)$ for example, the minimal polynomial for 0 is x, that for 1 is $x + 1$, and the one for Ω and Ψ is $x^2 + x + 1$ (they are conjugate primitive elements).

Again, these ideas can be extended to elements of $F = GF(p^n)$ as polynomials over any subfield S of order $r = p^j$, where $n = jk$ for some k, so F is of order r^k. Then each element a of F has a minimal polynomial over S, of degree $m \leq k$, with m distinct roots in F, and the m conjugates of a over S are $\{a, a^r, a^{r^2}, \ldots, a^{r^{m-1}}\}$. Also F/S is a vector space of dimension k over S, and a normal basis is a set of k distinct conjugates.

The *trace* of a over S is then defined as

$$\mathrm{Tr}_{F/S}(a) \equiv a + a^r + a^{r^2} + \ldots + a^{r^{k-1}}$$

and the *norm* is defined as

$$\mathrm{N}_{F/S}(a) \equiv a \cdot a^r \cdot a^{r^2} \cdot \ldots \cdot a^{r^{k-1}}$$

(If the minimal polynomial of a is of degree k, then the trace is the sum of the conjugates and the norm is the product of the conjugates.) It turns out that both the trace and the norm are always elements of the subfield S. For example, in $GF(2^2)/GF(2)$, both the trace and the norm of Ω are 1.

This brief introduction to Galois fields only covers the points relevant to the algorithm below. A nice, succinct introduction is given in [4]; for more depth and rigor, see [3].

3 S-box Algorithm

The S-box function of an input byte a is defined by two substeps:

1. *Inverse*: Let $c = a^{-1}$, the multiplicative inverse in $GF(2^8)$ (except if $a = 0$ then $c = 0$).

2. *Affine Transformation*: Then the output is $s = M\,c \oplus b$, where M is a specified 8×8 matrix of bits, b is a specified byte, and the bytes c, b, s are treated as vectors of bits. More explicitly:

$$\begin{pmatrix} s_7 \\ s_6 \\ s_5 \\ s_4 \\ s_3 \\ s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

   where bit #7 is the most significant and all bit operations are modulo 2.

The second substep is affine (linear plus a constant) and easy to implement; the algorithm for the first substep, finding the inverse, is described below.

The AES algorithm uses the particular Galois field of 8-bit bytes where the bits are coefficients of a polynomial (i.e., a polynomial basis), and multiplication is modulo the irreducible polynomial $q(x) = x^8 + x^4 + x^3 + x + 1$. (A 9-bit binary representation is $q(x) = 100011011$; this is the "smallest" irreducible polynomial of degree 8 over $GF(2)$, in the sense of comparing the binary number representations.) Let A be one root of $q(x)$; we will think of the polynomial basis as $[A^7, A^6, A^5, A^4, A^3, A^2, A, 1]$. It turns out that $A = 00000010$ is not a primitive element, but $A + 1 = 00000011$ is; we call it B. (B is a root of the second smallest irreducible polynomial: 100011101; see Table D.3 for more details.) Some implementations of AES use logarithm and antilogarithm tables, base B (as shown in Appendix D), for finding inverses and products in $GF(2^8)$. In particular, $A = B^{25}$. (Note: we will use Roman letters for specific elements of $GF(2^8)$, lowercase Greek letters for elements of $GF(2^4)$, and uppercase Greek letters for $GF(2^2)$; the naming scheme is summarized in Table D.3.)

Direct calculation of the inverse (modulo an eighth-degree polynomial) of a seventh-degree polynomial is not easy. But calculation of the inverse (modulo a second-degree polynomial) of a first-degree polynomial is relatively easy, as pointed out by Rijmen [10]. This suggests the following changes of representation.

First, we use the isomorphism between $GF(2^8)$ and $GF(2^8)/GF(2^4)$ to represent a general element g of $GF(2^8)$ as a polynomial (in y) over $GF(2^4)$, of degree 1 or less, as $g = \gamma_1 y + \gamma_0$, with multiplication modulo an irreducible polynomial $r(y) = y^2 + \tau y + \nu$. Here, all the coefficients are in $GF(2^4)$. Then the pair $[\gamma_1, \gamma_0]$ represents g in terms of a polynomial basis $[Y, 1]$ where Y is one root of $r(y)$. Of course, we are free to use any basis for this representation, for example the normal basis $[Y^{16}, Y]$. Note that

$$r(y) = y^2 + \tau y + \nu = (y + Y)(y + Y^{16})$$

so $\tau = \mathrm{Tr}_{\mathbb{F}_{256}/\mathbb{F}_{16}}(Y)$ is the trace and $\nu = \mathrm{N}_{\mathbb{F}_{256}/\mathbb{F}_{16}}(Y)$ is the norm of Y.

Second, using $GF(2^4)/GF(2^2)$ we can similarly represent $GF(2^4)$ as linear polynomials (in z) over $GF(2^2)$, as $\gamma = \Gamma_1 z + \Gamma_0$, with multiplication modulo an irreducible polynomial

$s(z) = z^2 + Tz + N$, with all the coefficients in $GF(2^2)$. Again, this uses a polynomial basis $[Z, 1]$ for $GF(2^4)/GF(2^2)$, where Z is one root of $s(z)$. We could use any basis, such as the normal basis $[Z^4, Z]$. And for the same reasons above, $T = \mathrm{Tr}_{\mathbb{F}_{16}/\mathbb{F}_4}(Z)$ is the trace and $N = \mathrm{N}_{\mathbb{F}_{16}/\mathbb{F}_4}(Z)$ is the norm of Z (considering T and N as uppercase Greek for τ and ν).

Third we use $GF(2^2)/GF(2)$ to represent $GF(2^2)$ as linear polynomials (in w) over $GF(2)$, as $\Gamma = g_1 w + g_0$, with multiplication modulo $t(w) = w^2 + w + 1$, where $g_1, g_0 \in \{0, 1\}$. This uses a polynomial basis $[W, 1]$, where W is either Ω or Ψ; a normal basis would be $[W^2, W]$. (Note that the trace and norm of Ω and Ψ are 1.)

This allows operations in $GF(2^8)$ to be expressed in terms of simpler operations in $GF(2^4)$, which in turn are expressed in the simple operations of $GF(2^2)$. In particular, we want to find the inverse in $GF(2^8)$. Say the inverse of $g = \gamma_1 y + \gamma_0$ is $d = \delta_1 y + \delta_0$. Then (recalling subtraction is the same as addition in $GF(2^n)$)

$$\begin{aligned} gd &= (\gamma_1 y + \gamma_0)(\delta_1 y + \delta_0) \bmod (y^2 + \tau y + \nu) \\ &= [(\gamma_1\delta_1)y^2 + (\gamma_1\delta_0 + \gamma_0\delta_1)y + (\gamma_0\delta_0)] \bmod (y^2 + \tau y + \nu) \\ &= [(\gamma_1\delta_1)y^2 + (\gamma_1\delta_0 + \gamma_0\delta_1)y + (\gamma_0\delta_0)] + (\gamma_1\delta_1)(y^2 + \tau y + \nu) \\ &= (\gamma_1\delta_0 + \gamma_0\delta_1 + \gamma_1\delta_1\tau)y + (\gamma_0\delta_0 + \gamma_1\delta_1\nu) \\ = 1 &= 0y + 1 \end{aligned}$$

Solving the two equations

$$\begin{aligned} 0 &= \gamma_1\delta_0 + (\gamma_0 + \gamma_1\tau)\delta_1 \\ 1 &= \gamma_0\delta_0 + (\gamma_1\nu)\delta_1 \end{aligned}$$

by

$$\begin{aligned} 0 &= \gamma_1\gamma_0\delta_0 + (\gamma_0^2 + \gamma_1\gamma_0\tau)\delta_1 \\ \gamma_1 &= \gamma_1\gamma_0\delta_0 + (\gamma_1^2\nu)\delta_1 \end{aligned}$$

gives

$$\begin{aligned} \gamma_1 &= (\gamma_1^2\nu + \gamma_1\gamma_0\tau + \gamma_0^2)\delta_1 \\ \gamma_1\delta_0 &= (\gamma_0 + \gamma_1\tau)\delta_1 \end{aligned}$$

so that

$$\begin{aligned} \delta_1 &= (\gamma_1^2\nu + \gamma_1\gamma_0\tau + \gamma_0^2)^{-1}\,\gamma_1 \\ \delta_0 &= (\gamma_1^2\nu + \gamma_1\gamma_0\tau + \gamma_0^2)^{-1}\,(\gamma_0 + \gamma_1\tau) \end{aligned}$$

So finding an inverse in $GF(2^8)$ involves an inverse and several multiplications in $GF(2^4)$. (Addition in $GF(2^4)$ as 4-bit elements, using any basis, is just bitwise exclusive-or.)

Similarly, to find the inverse in $GF(2^4)$ of $\gamma = \Gamma_1 z + \Gamma_0$ as $\delta = \Delta_1 z + \Delta_0$, then

$$\gamma\delta = (\Gamma_1\Delta_0 + \Gamma_0\Delta_1 + \Gamma_1\Delta_1 T)z + (\Gamma_0\Delta_0 + \Gamma_1\Delta_1 N)$$

so

$$\begin{aligned}\Delta_1 &= (\Gamma_1^2 N + \Gamma_1\Gamma_0 T + \Gamma_0^2)^{-1}\,\Gamma_1 \\ \Delta_0 &= (\Gamma_1^2 N + \Gamma_1\Gamma_0 T + \Gamma_0^2)^{-1}\,(\Gamma_0 + \Gamma_1 T)\end{aligned}$$

And to find the inverse in $GF(2^2)$ of $\Gamma = g_1 w + g_0$ as $\Delta = d_1 w + d_0$, then

$$\Gamma\Delta = (g_1 d_0 + g_0 d_1 + g_1 d_1)w + (g_0 d_0 + g_1 d_1)$$

so

$$\begin{aligned}d_1 &= (g_1^2 + g_1 g_0 + g_0^2)^{-1}\,g_1 \\ d_0 &= (g_1^2 + g_1 g_0 + g_0^2)^{-1}\,(g_0 + g_1)\end{aligned}$$

since both coefficients (trace and norm) in the polynomial $t(w)$ are 1. This can be further simplified because for $g \in GF(2)$, $g^2 = g^{-1} = g$, so

$$\begin{aligned}d_1 &= (g_1 + g_1 g_0 + g_0)\,g_1 \\ &= (g_1 + g_1 g_0 + g_1 g_0) \\ &= g_1 \\ d_0 &= (g_1 + g_1 g_0 + g_0)\,(g_0 + g_1) \\ &= (g_1 g_0 + g_1 + g_1 g_0 + g_1 g_0 + g_0 + g_1 g_0) \\ &= g_1 + g_0\end{aligned}$$

Note that if the above inversion formulas are applied to a zero input then the output will also be zero, so that special case is handled automatically.

How do these calculations change if we use *normal* bases at each level? In $GF(2^8)$, to find the inverse of $g = \gamma_1 Y^{16} + \gamma_0 Y$ as $d = \delta_1 Y^{16} + \delta_0 Y$, we use the fact that both Y and Y^{16} satisfy $y^2 + \tau y + \nu = 0$ where $\tau = Y^{16} + Y$ and $\nu = (Y^{16})Y$. Then $1 = \tau^{-1}(Y^{16} + Y)$, so:

$$\begin{aligned}gd &= (\gamma_1 Y^{16} + \gamma_0 Y)(\delta_1 Y^{16} + \delta_0 Y) \\ &= (\gamma_1\delta_1)(Y^{16})^2 + (\gamma_1\delta_0 + \gamma_0\delta_1)(Y^{16})Y + (\gamma_0\delta_0)Y^2 \\ &= (\gamma_1\delta_1)(\tau Y^{16} + \nu) + (\gamma_1\delta_0 + \gamma_0\delta_1)\nu + (\gamma_0\delta_0)(\tau Y + \nu) \\ &= (\gamma_1\delta_1\tau)Y^{16} + (\gamma_0\delta_0\tau)Y + [(\gamma_1\delta_1)\nu + (\gamma_1\delta_0 + \gamma_0\delta_1)\nu + (\gamma_0\delta_0)\nu)] \\ &= (\gamma_1\delta_1\tau)Y^{16} + (\gamma_0\delta_0\tau)Y + [(\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu]\tau^{-1}(Y^{16} + Y) \\ &= [\gamma_1\delta_1\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}]Y^{16} + [\gamma_0\delta_0\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}]Y \\ = 1 &= \tau^{-1}(Y^{16} + Y)\end{aligned}$$

Solving the two equations

$$\begin{aligned}\tau^{-1} &= \gamma_1\delta_1\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1} \\ \tau^{-1} &= \gamma_0\delta_0\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}\end{aligned}$$

gives

$$\begin{aligned} 0 &= \gamma_1\delta_1 + \gamma_0\delta_0 \\ 1 &= \gamma_1\delta_1\tau^2 + (\gamma_1\delta_0 + \gamma_0\delta_1)\nu \\ \gamma_0 &= \gamma_1\gamma_0\delta_1\tau^2 + (\gamma_1\gamma_0\delta_0 + \gamma_0^2\delta_1)\nu \\ &= \gamma_1\gamma_0\delta_1\tau^2 + (\gamma_1^2\delta_1 + \gamma_0^2\delta_1)\nu \\ &= [\gamma_1\gamma_0\tau^2 + (\gamma_1^2 + \gamma_0^2)\nu]\delta_1 \end{aligned}$$

so that

$$\begin{aligned} \delta_1 &= [\gamma_1\gamma_0\tau^2 + (\gamma_1^2 + \gamma_0^2)\nu]^{-1}\ \gamma_0 \\ \delta_0 &= [\gamma_1\gamma_0\tau^2 + (\gamma_1^2 + \gamma_0^2)\nu]^{-1}\ \gamma_1 \end{aligned}$$

Again, finding an inverse in $GF(2^8)$ involves an inverse and several multiplications in $GF(2^4)$.

Analogously, to find the inverse in $GF(2^4)$ of $\gamma = \Gamma_1 Z^4 + \Gamma_0 Z$ as $\delta = \Delta_1 Z^4 + \Delta_0 Z$, then

$$\gamma\delta = [\Gamma_1\Delta_1 T + (\Gamma_1 + \Gamma_0)(\Delta_1 + \Delta_0)NT^{-1}]Z^4 + [\Gamma_0\Delta_0 T + (\Gamma_1 + \Gamma_0)(\Delta_1 + \Delta_0)NT^{-1}]Z$$

so

$$\begin{aligned} \Delta_1 &= [\Gamma_1\Gamma_0 T^2 + (\Gamma_1^2 + \Gamma_0^2)N]^{-1}\ \Gamma_0 \\ \Delta_0 &= [\Gamma_1\Gamma_0 T^2 + (\Gamma_1^2 + \Gamma_0^2)N]^{-1}\ \Gamma_1 \end{aligned}$$

And to find the inverse in $GF(2^2)$ of $\Gamma = g_1 W^2 + g_0 W$ as $\Delta = d_1 W^2 + d_0 W$, then

$$\Gamma\Delta = [g_1 d_1 + (g_1 + g_0)(d_1 + d_0)]W^2 + [g_0 d_0 + (g_1 + g_0)(d_1 + d_0)]W$$

so

$$\begin{aligned} d_1 &= [g_1 g_0 + g_1 + g_0]\ g_0 \\ &= g_0 \\ d_0 &= [g_1 g_0 + g_1 + g_0]\ g_1 \\ &= g_1 \end{aligned}$$

using the same simplifications as before in $GF(2)$.

This shows how we break one problem (the 8-bit inverse in $GF(2^8)$) down into simpler problems (4-bit operations in $GF(2^4)$), which can further be broken down to still simpler problems (2-bit operations in $GF(2^2)$ and bit operations in $GF(2)$).

4 Optimizations

There are several ways to reorganize the calculations above in order to reduce the total operation count and hence minimize the circuitry required. Additionally, there is some freedom in the choice of the coefficients in the minimal polynomials $r(y)$ and $s(z)$ to give convenient multipliers.

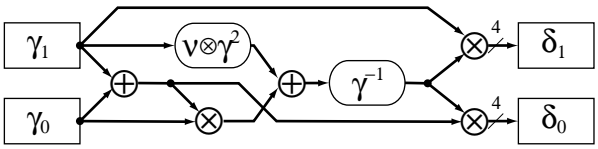

Figure 1: Polynomial $GF(2^8)$ inverter: $(\gamma_1 y + \gamma_0)^{-1} = (\delta_1 y + \delta_0)$ Notes: the datapaths all have the same bit width, shown at the output (4 bits here); addition is bitwise exclusive-OR; and sub-field multipliers appear below.

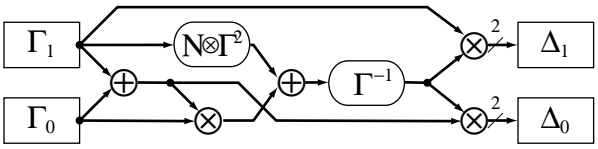

Figure 2: Polynomial $GF(2^4)$ inverter: $(\Gamma_1 z + \Gamma_0)^{-1} = (\Delta_1 z + \Delta_0)$

The inverse formulas in $GF(2^8)/GF(2^4)$ would simplify considerably if we could choose $\tau = 0$ or $\nu = 0$, but neither choice gives an irreducible polynomial. We *can* find irreducible polynomials with $\tau = 1$, which is also convenient. This is better than choosing $\nu = 1$, since τ appears in two products in the inverse (in the polynomial basis, but even for the normal basis $\tau = 1$ turns out to be preferable). We can't choose both $\nu = \tau = 1$ since then we get the minimal polynomial of Ω and Ψ in $GF(2^2)$, a subfield of $GF(2^4)$. So from here on we let $\tau = 1$ and similarly let $T = 1$.

4.1 Polynomial Basis Optimizations

First we consider optimizations using polynomial bases. In $GF(2^8)/GF(2^4)$ the only operation required is the inverse. Satoh et al.[12] indicate the following steps in inverting $g = \gamma_1 y + \gamma_0$, where we return to the $\oplus, \otimes$ notation, and give names to intermediate results, to clarify the subfield operations needed:

$$\begin{aligned} \phi &= \gamma_1 \oplus \gamma_0 \\ \theta &= [(\nu \otimes \gamma_1^2) \oplus (\phi \otimes \gamma_0)]^{-1} \\ g^{-1} &= [\theta \otimes \gamma_1] y + [\theta \otimes \phi] \end{aligned}$$

(Note: in the notation of [12], our ν becomes λ and our N becomes ϕ.) The operations required in the subfield $GF(2^4)/GF(2^2)$ include an inverter, multipliers, and adders (bitwise XOR); see Figure 1.

The subfield inversions can be performed similarly, as suggested by [12]. So to invert $\gamma = \Gamma_1 z + \Gamma_0$ in $GF(2^4)$:

$$\begin{aligned} \Phi &= \Gamma_1 \oplus \Gamma_0 \\ \Theta &= [(N \otimes \Gamma_1^2) \oplus (\Phi \otimes \Gamma_0)]^{-1} \\ \gamma^{-1} &= [\Theta \otimes \Gamma_1] z + [\Theta \otimes \Phi] \end{aligned}$$

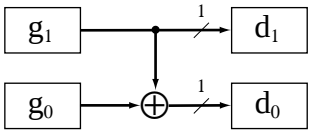

Figure 3: Polynomial $GF(2^2)$ inverter: $(g_1 w + g_0)^{-1} = (d_1 w + d_0)$ Note: in $GF(2^2)$ inverting is the same as squaring.

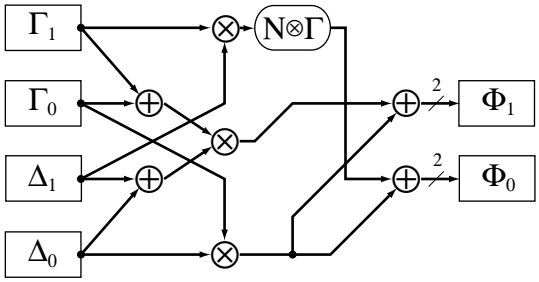

Figure 4: Polynomial $GF(2^4)$ multiplier: $(\Gamma_1 z + \Gamma_0) \otimes (\Delta_1 z + \Delta_0) = (\Phi_1 z + \Phi_0)$

(see Figure 2). And in $GF(2^2)$ the inverse of $\Gamma = g_1 w + g_0$ is simply:

$$\Gamma^{-1} \;=\; [g_1]w + [g_1 \oplus g_0]$$

(see Figure 3).

The multiplier in $GF(2^4)$ given by [12] finds the product $\gamma\delta = (\Gamma_1 z + \Gamma_0)(\Delta_1 z + \Delta_0)$ by the steps

$$\begin{aligned} \Phi &= \Gamma_0 \otimes \Delta_0 \\ \gamma\delta &= [\Phi \oplus (\Gamma_1 \oplus \Gamma_0) \otimes (\Delta_1 \oplus \Delta_0)]z + [\Phi \oplus (N \otimes \Gamma_1 \otimes \Delta_1)] \end{aligned}$$

(see Figure 4.) Similarly in $GF(2^2)$, the product $\Gamma\Delta = (g_1 w + g_0)(d_1 w + d_0)$ can be found by

$$\begin{aligned} f &= g_0 \otimes d_0 \\ \Gamma\Delta &= [f \oplus (g_1 \oplus g_0) \otimes (d_1 \oplus d_0)]w + [f \oplus (g_1 \otimes d_1)] \end{aligned}$$

(where in $GF(2)$, $\otimes$ means AND; see Figure 5).

For further efficiency, multiplication by a known constant (e.g. ν above), which we will call "scaling," should use a specialized circuit instead of a generic multiplier, and the same is true for squaring.

Scaling $\gamma = \Gamma_1 z + \Gamma_0$ in $GF(2^4)$ by $\nu = \Delta_1 z + \Delta_0$ becomes simpler for special choices of ν, for example, if $\Delta_0 = 0$. (It is not possible to choose $\Delta_1 = 0$, because then $r(y)$ is reducible.) Then

$$\nu\gamma \;=\; [\Delta_1 \otimes (\Gamma_1 \oplus \Gamma_0)]z + [(N\Delta_1) \otimes \Gamma_1]$$

And choosing $N = \Delta_1^{-1}$ makes scaling by ν even simpler:

$$\nu\gamma \;=\; [(N^{-1}) \otimes (\Gamma_1 \oplus \Gamma_0)]z + [\Gamma_1]$$

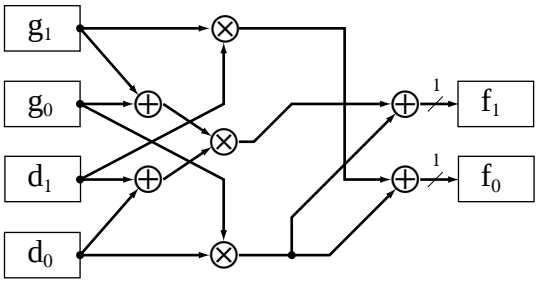

Figure 5: Polynomial $GF(2^2)$ multiplier: $(g_1w + g_0) \otimes (d_1w + d_0) = (f_1w + f_0)$ Note: in $GF(2)$, multiplication is bitwise AND.

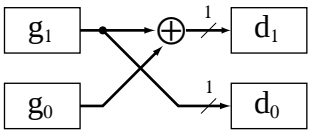

Figure 6: Polynomial $GF(2^2)$ w-scaler: $(w)\otimes(g_1w+g_0) = (d_1w+d_0)$

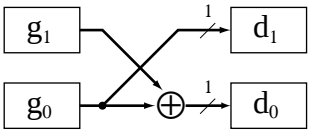

Figure 7: Polynomial $GF(2^2)$ w^2-scaler: $(w^2)\otimes(g_1w+g_0) = (d_1w+d_0)$

In $GF(2^2)$, since $N \neq 0, 1$ (so that $s(z) = z^2 + z + N$ is irreducible over $GF(2^2)$), then both N and $N + 1$ are roots of $t(w) = w^2 + w + 1$, and $N^{-1} = N^2 = N + 1$. Depending on which root we choose for the polynomial basis $[w, 1]$, then either $N = w$ or $N^2 = w$. In either case, since we need scalers for both N and N^2, this corresponds to scalers for both w and w^2, and scaling becomes

$$\begin{aligned}(w) \otimes (g_1w + g_0) &= [g_1 \oplus g_0]w + [g_1] \\ (w^2) \otimes (g_1w + g_0) &= [g_0]w + [g_0 \oplus g_1]\end{aligned}$$

(see Figures 6–7).

Squaring $\gamma = \Gamma_1 z \oplus \Gamma_0$ in $GF(2^4)$ corresponds to

$$\begin{aligned}\Phi &= \Gamma_1^2 \\ \gamma^2 &= [\Phi]z + [\Gamma_0^2 \oplus N \otimes \Phi]\end{aligned}$$

Of course, squaring $\Gamma = g_1w + g_0$ in the subfield $GF(2^2)$ can be done similarly, using further simplifications in $GF(2)$:

$$\Gamma^2 = [g_1]w + [g_0 \oplus g_1]$$

Note that, in $GF(2^2)$, every nonzero element Γ satisfies $\Gamma^3 = 1$, so $\Gamma^{-1} = \Gamma^2$, i.e., the $GF(2^2)$ inverter is the same as the squarer (see Figure 3).

Another improvement comes from combining the square in $GF(2^4)$ with the scaling by ν, since it is only this combination that is required in the $GF(2^8)$ inverter. Then for the choice

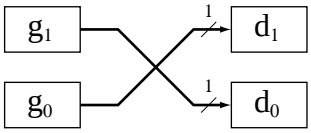

Figure 8: Polynomial $GF(2^2)$ square-scaler: $(w)\otimes(g_1w+g_0)^2 = (d_1w+d_0)$ Note: no gates required.

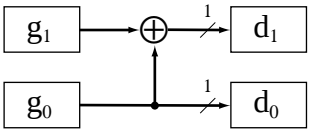

Figure 9: Polynomial $GF(2^2)$ square-scaler: $(w^2) \otimes (g_1w + g_0)^2 = (d_1w + d_0)$

of ν above

$$\begin{aligned}
\nu \otimes \gamma^2 &= \nu \otimes (\Gamma_1 z + \Gamma_0)^2 \\
&= \nu \otimes ([\Gamma_1^2]z + [\Gamma_0^2 \oplus N \otimes \Gamma_1^2]) \\
&= [N^2 \otimes (\Gamma_1^2 \oplus (\Gamma_0^2 \oplus N \otimes \Gamma_1^2))]z + [\Gamma_1^2] \\
&= [(N^2+1) \otimes \Gamma_1^2 \oplus N^2 \otimes \Gamma_0^2]z + [\Gamma_1^2] \\
&= [N \otimes \Gamma_1^2 \oplus N^2 \otimes \Gamma_0^2]z + [\Gamma_1^2]
\end{aligned}$$

In the subfield $GF(2^2)$, combining squaring with scaling by w gives

$$\begin{aligned}
(w) \otimes \Gamma^2 &= (w) \otimes (g_1w + g_0)^2 \\
&= (w) \otimes ([g_1]w + [g_0 \oplus g_1]) \\
&= [g_1 \oplus (g_0 \oplus g_1)]w + [g_1] \\
&= [g_0]w + [g_1]
\end{aligned}$$

(see Figure 8) so this combination is free (being just a swap of two bits)! This suggests that if we choose $w = N$, then

$$\nu \otimes \gamma^2 = [\{N \otimes \Gamma_1^2\} \oplus N \otimes \{N \otimes \Gamma_0^2\}]z + [N^2 \otimes \{N \otimes \Gamma_1^2\}]$$

performs this combined operation with one addition and two scalings in the subfield, since the operations in $\{\}$ are free. Or, if instead we choose $w = N^2$ (see Figure 9) then

$$\nu \otimes \gamma^2 = [N^2 \otimes \{N^2 \otimes \Gamma_1^2\} \oplus \{N^2 \otimes \Gamma_0^2\}]z + [N \otimes \{N^2 \otimes \Gamma_1^2\}]$$

again requiring only one addition and two scalings.

Also, combining the multiplication in $GF(2^2)$ with scaling by N gives a small improvement; this combination appears in the $GF(2^4)$ multiplier. If $N = w$, for example, the scaled product $N\Gamma\Delta = w(g_1w + g_0)(d_1w + d_0)$ becomes

$$\begin{aligned}
f &= (g_1 \oplus g_0) \otimes (d_1 \oplus d_0) \\
N\Gamma\Delta &= [f \oplus (g_1 \otimes d_1)]w + [f \oplus (g_0 \otimes d_0)]
\end{aligned}$$

so the scaling is "free."

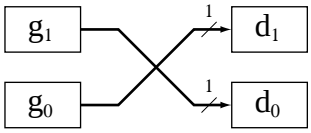

Figure 10: Normal $GF(2^2)$ inverter: $(g_1W^2 + g_0W)^2 = (d_1W^2 + d_0W)$ Note: no gates required; again, in $GF(2^2)$ inverting is the same as squaring.

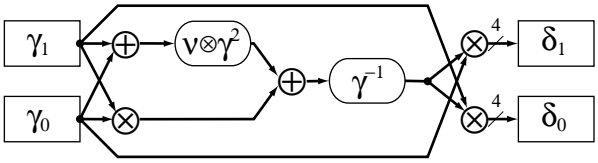

Figure 11: Normal $GF(2^8)$ inverter: $(\gamma_1 Y^{16} + \gamma_0 Y)^{-1} = (\delta_1 Y^{16} + \delta_0 Y)$

4.2 Normal Basis Optimizations

Analogous optimizations are available using normal bases, although the details change. For instance, in $GF(2^2)$ with a normal basis $[W^2, W]$ the squaring operation is free:

$$(g_1W^2 + g_0W)^2 \;=\; g_0W^2 + g_1W$$

(see Figure 10) And while it is still convenient to choose $\tau = 1$ and $T = 1$, different choices for ν and N can make the combination of squaring and scaling in $GF(2^4)$ efficient. Here scaling the square of $\gamma = \Gamma_1 Z^4 + \Gamma_0 Z$ by $\nu = \Delta_1 Z^4 + \Delta_0 Z$ gives

$$\begin{aligned}
\nu \otimes \gamma^2 &= \nu \otimes \{[\Gamma_1^2 \oplus N \otimes (\Gamma_1^2 \oplus \Gamma_0^2)]Z^4 + [\Gamma_0^2 \oplus N \otimes (\Gamma_1^2 + \Gamma_0^2)]Z\} \\
&= [\Delta_1 \otimes (\Gamma_1^2 \oplus N \otimes (\Gamma_1^2 \oplus \Gamma_0^2)) + N(\Delta_1 + \Delta_0) \otimes (\Gamma_1^2 \oplus \Gamma_0^2)]Z^4 \\
&\quad + [\Delta_0 \otimes (\Gamma_0^2 \oplus N \otimes (\Gamma_1^2 \oplus \Gamma_0^2)) + N(\Delta_1 + \Delta_0) \otimes (\Gamma_1^2 \oplus \Gamma_0^2)]Z \\
&= [(\Delta_1 + N\Delta_0) \otimes \Gamma_1^2 \oplus (N\Delta_0) \otimes \Gamma_0^2]Z^4 + [(N\Delta_1) \otimes \Gamma_1^2 \oplus (\Delta_0 + N\Delta_1) \otimes \Gamma_0^2]Z
\end{aligned}$$

This can be made more efficient by choosing, for example, $\Delta_1 = N\Delta_0$, giving

$$\begin{aligned}
\nu \otimes \gamma^2 &= [(N\Delta_0) \otimes \Gamma_0^2]Z^4 + [(N\Delta_1) \otimes \Gamma_1^2 \oplus (\Delta_0 + N\Delta_1) \otimes \Gamma_0^2]Z \\
&= [(N\Delta_0) \otimes \Gamma_0^2]Z^4 + [(N^2\Delta_0) \otimes \Gamma_1^2 \oplus ((N^2+1)\Delta_0) \otimes \Gamma_0^2]Z \\
&= [(N\Delta_0) \otimes \Gamma_0^2]Z^4 + [(N^2\Delta_0) \otimes \Gamma_1^2 \oplus (N\Delta_0) \otimes \Gamma_0^2]Z
\end{aligned}$$

which again requires only two scalings and an addition (note the common sub-expression), since squaring is free. Also, it is possible to choose $\Delta_0 = N^{-1}$ to save one scaling.

The top level inversion, of $g = \gamma_1 Y^{16} + \gamma_0 Y$ in $GF(2^8)$, can be done by

$$\begin{aligned}
\theta &= [\{\nu \otimes (\gamma_1 \oplus \gamma_0)^2\} \oplus (\gamma_1 \otimes \gamma_0)]^{-1} \\
g^{-1} &= [\theta \otimes \gamma_0]Y^{16} + [\theta \otimes \gamma_1]Y
\end{aligned}$$

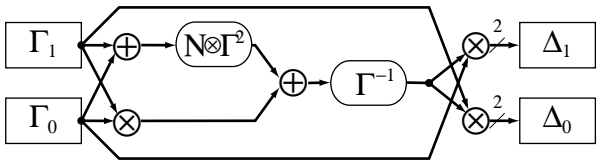

Figure 12: Normal $GF(2^4)$ inverter: $(\Gamma_1 Z^4 + \Gamma_0 Z)^{-1} = (\Delta_1 Z^4 + \Delta_0 Z)$

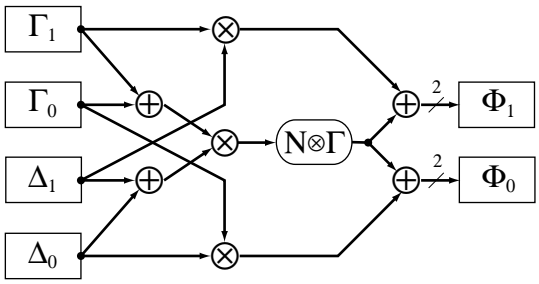

Figure 13: Normal $GF(2^4)$ multiplier: $(\Gamma_1 Z^4 + \Gamma_0 Z) \otimes (\Delta_1 Z^4 + \Delta_0 Z) = (\Phi_1 Z^4 + \Phi_0 Z)$

(see Figure 11). Similarly, $\gamma = \Gamma_1 Z^4 + \Gamma_0 Z$ in $GF(2^4)$ is inverted by

$$\begin{aligned} \Theta &= [N \otimes (\Gamma_1 \oplus \Gamma_0)^2 \oplus (\Gamma_1 \otimes \Gamma_0)]^{-1} \\ \gamma^{-1} &= [\Theta \otimes \Gamma_0] Z^4 + [\Theta \otimes \Gamma_1] Z \end{aligned}$$

(see Figure 12) where in $GF(2^2)$ inversion is the same as squaring, which is free.

In $GF(2^4)$ the product $\gamma\delta = (\Gamma_1 Z^4 + \Gamma_0 Z)(\Delta_1 Z^4 + \Delta_0 Z)$ is found by

$$\begin{aligned} \Phi &= N \otimes (\Gamma_1 \oplus \Gamma_0) \otimes (\Delta_1 \oplus \Delta_0) \\ \gamma\delta &= [\Phi \oplus (\Gamma_1 \otimes \Delta_1)] Z^4 + [\Phi \oplus (\Gamma_0 \otimes \Delta_0)] Z \end{aligned}$$

(see Figure 13) And in $GF(2^2)$, the product $\Gamma\Delta = (g_1 W^2 + g_0 W)(d_1 W^2 + d_0 W)$ corresponds to

$$\begin{aligned} f &= (g_1 \oplus g_0) \otimes (d_1 \oplus d_0) \\ \Gamma\Delta &= [f \oplus (g_1 \otimes d_1)] W^2 + [f \oplus (g_0 \otimes d_0)] W \end{aligned}$$

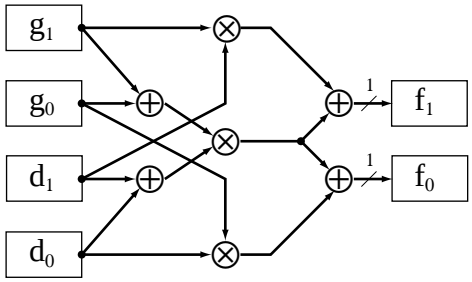

Figure 14: Normal $GF(2^2)$ multiplier: $(g_1 W^2 + g_0 W) \otimes (d_1 W^2 + d_0 W) = (f_1 W^2 + f_0 W)$

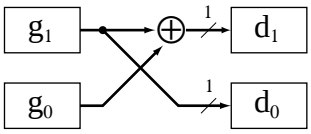

Figure 15: Normal $GF(2^2)$ w-scaler: $(W) \otimes (g_1W^2 + g_0W) = (d_1W^2 + d_0W)$

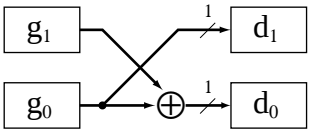

Figure 16: Normal $GF(2^2)$ w^2-scaler: $(W^2) \otimes (g_1W^2 + g_0W) = (d_1W^2 + d_0W)$

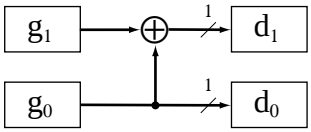

Figure 17: Normal $GF(2^2)$ square-scaler: $(W) \otimes (g_1W^2 + g_0W)^2 = (d_1W^2 + d_0W)$

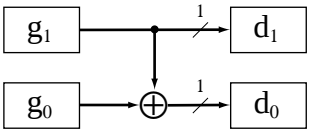

Figure 18: Normal $GF(2^2)$ square-scaler: $(W^2) \otimes (g_1W^2 + g_0W)^2 = (d_1W^2 + d_0W)$

(see Figure 14) Scaling in $GF(2^2)$ is accomplished by

$$\begin{aligned}(W) \otimes (g_1W^2 + g_0W) &= [g_1 \oplus g_0]W^2 + [g_1]W \\ (W^2) \otimes (g_1W^2 + g_0W) &= [g_0]W^2 + [g_0 \oplus g_1]W\end{aligned}$$

(see Figures 15–18)

At this level of optimization, the smallest $GF(2^8)$ inverter using normal bases turns out to use exactly the same number of gates as the smallest polynomial version. However, this does not account for further optimizations from common subexpressions (discussed below), nor for the change in representation (basis) required on entering and leaving the S-box.

4.3 Mixing Basis Types

There is no reason why the three bases, for $GF(2^8)$, $GF(2^4)$, and $GF(2^2)$, should all be polynomial bases or all be normal bases; one is free to choose either type of basis at each level. (Of course, one could choose other types of basis at each level, but both polynomial and normal bases have structure that leads to efficient calculation, which is lacking in other bases.) We have seen that the inverters in $GF(2^8)$ for both types of basis require the same number and type of operations in $GF(2^4)$, and similarly for the inverters in $GF(2^4)$. The multipliers also use the same operations for both types of bases; the same is true for the scalers in $GF(2^2)$.

In $GF(2^2)$, squaring is free with a normal basis, while the combination $w \otimes \Gamma^2$ is free with a polynomial basis. Since the $GF(2^4)$ inverter needs one $GF(2^2)$ inverter (same as squaring) and one combo $N \otimes \Gamma^2$, then as long as $N = w$ this gives no preference for either type of basis.

The main differences then are in the combined squaring-scaling operation required by the $GF(2^8)$ inverters: $\nu \otimes \gamma^2$. The details vary for the calculations this operation requires in

$GF(2^2)$, depending on the basis types and the relations between ν, N, z, and w. The tables below summarize all the different cases.

Coefficients: Polynomial $GF(2^4)$ Basis				XOR Gates		
$\nu =$		$\nu \otimes (Az+B)^2 =$		poly. $GF(2^2)$		norm.
$Cz+D$		$[(CN^2+D)A^2+CB^2]z+[(C+D)NA^2+DB^2]$		$w=N$	$w=N^2$	$GF(2^2)$
N	0	$A^2 \oplus N \otimes B^2$	$N^2 \otimes A^2$	4	5	4
N^2	0	$N \otimes A^2 \oplus N^2 \otimes B^2$	A^2	4	5	4
N	N	$N^2 \otimes A^2 \oplus N \otimes B^2$	$N \otimes B^2$	3	4	4
N^2	N^2	$A^2 \oplus N^2 \otimes B^2$	$N^2 \otimes B^2$	4	4	3
N	1	$N \otimes B^2$	$(A \oplus B)^2$	3	5	3
N^2	N	$N^2 \otimes B^2$	$N \otimes (A \oplus B)^2$	3	4	4
N	N^2	$N \otimes (A \oplus B)^2$	$N \otimes (A \oplus B)^2 \oplus B^2$	5	7	5
N^2	1	$N^2 \otimes (A \oplus B)^2$	$N^2 \otimes (A \oplus B)^2 \oplus N \otimes B^2$	5	6	6

Coefficients: Normal $GF(2^4)$ Basis				XOR Gates		
$\nu =$		$\nu \otimes (Az^4+Bz)^2 =$		poly. $GF(2^2)$		norm.
Cz^4+Dz		$[CA^2+DN(A^2+B^2)]z^4+[CN(A^2+B^2)+DB^2]z$		$w=N$	$w=N^2$	$GF(2^2)$
N	0	$N \otimes A^2$	$N^2 \otimes (A \oplus B)^2$	3	4	4
0	N	$N^2 \otimes (A \oplus B)^2$	$N \otimes B^2$	3	4	4
N^2	0	$N^2 \otimes A^2$	$(A \oplus B)^2$	4	4	3
0	N^2	$(A \oplus B)^2$	$N^2 \otimes B^2$	4	4	3
N	1	$N \otimes B^2$	$N^2 \otimes A^2 \oplus N \otimes B^2$	3	4	4
1	N	$N \otimes A^2 \oplus N^2 \otimes B^2$	$N \otimes A^2$	3	4	4
N^2	1	$A^2 \oplus N \otimes B^2$	A^2	3	5	3
1	N^2	B^2	$N \otimes A^2 \oplus B^2$	3	5	3

The first table is for a polynomial basis in $GF(2^4)$; the second is for a normal basis. The first two columns show the coefficients of ν in terms of N, which depends on the bases for $GF(2^4)$ and $GF(2^2)$. (All eight possibilities are shown for both tables, although, due to the symmetry of normal bases, the second table essentially has only four cases, each shown two ways.) The next two columns show the coefficients of $\nu \otimes \gamma^2$ that need to be calculated; each is expressed in a form to suggest a compact calculation. The last three columns show the total number of XOR gates required for: a polynomial basis for $GF(2^2)$ with $w = N$; a polynomial basis for $GF(2^2)$ with $w = N^2$; or a normal basis for $GF(2^2)$. Note that addition in $GF(2^2)$ uses two XOR's while scaling uses one. These numbers incorporate taking advantage of whichever calculation is free in the particular $GF(2^2)$ basis, and *include* this adjustment: for a polynomial basis in $GF(2^2)$ with $w = N^2$, add one since the $N \otimes \Gamma^2$ in the inverter requires a scaling.

Altogether, 85 XOR's and 36 AND's are needed for the rest of the calculation, so the inverter could include from 88 to 92 XOR's (excluding common subexpression optimizations below), depending on basis choice. This does *not* account for the gates needed to change

between representations (bases) on entering and exiting the S-box. Since there is only a difference of 4 XOR's between the smallest and largest inverter that incorporate the above optimizations, the change of basis can play an important role.

4.4 Common Subexpressions

A further level of optimization comes from finding subexpressions that appear more than once in the above hierarchical view of the inverter. Each of these common subexpressions need only be computed once, thus reducing the size of the inverter.

As [12] mentions, one place this occurs is when the same factor is input to two different multipliers. Each multiplier needs the sum of the high and low halves of each factor, so a shared factor saves one addition in the subfield. For example, a 2-bit factor shared by two $GF(2^2)$ multipliers saves one XOR. Moreover, since each $GF(2^4)$ multiplier includes three $GF(2^2)$ multipliers, then a shared 4-bit factor implies three corresponding shared 2-bit factors. So each shared 4-bit factor saves five XOR's (one 2-bit addition and three 1-bit additions).

The polynomial-basis inverters for $GF(2^8)$ and $GF(2^4)$ each have two different factors that are each shared between two multipliers (which appeared as ϕ and θ in $GF(2^4)$, Φ and Θ in $GF(2^2)$). However, each of the corresponding normal-basis inverters share all three factors among the three multipliers (called θ, γ_1 and γ_0 in $GF(2^4)$, and Θ, Γ_1 and Γ_0 in $GF(2^2)$). This gives a significant advantage to using a normal basis in $GF(2^8)$, since the additional shared factor in the $GF(2^8)$ inverter saves five more XOR's.

Another place to look is in the $GF(2^4)$ square-scale combination. It turns out that, of the 36 variations in the tables (page 18), a repeated sum of two bits can be found in 10 cases (all with polynomial $GF(2^4)$ bases), saving one XOR.

A more subtle saving occurs in the $GF(2^4)$ inverter. There are essentially 6 versions, depending on the types of basis for $GF(2^4)$ and $GF(2^2)$, and for a polynomial $GF(2^2)$ basis whether $N = w$ or $N = w^2$. Each case can be improved by at least one XOR, and in two cases, by two XOR's. These improvements all involve bit sums computed for common factors being combined with some other operations, but the details vary from case to case. For example, with both bases polynomial, combining the $GF(2^2)$ inverter with finding the sum of its output bits (it's a shared factor) saves one XOR. Or for both normal bases, combining the sum of the high and low inputs and the following square-scale operation with the bit sums of the high and low inputs (shared factors) again saves one XOR.

The last optimization occurs in the $GF(2^8)$ inverter, combining the bit sums for shared input factors with parts of the square-scale operation. Again the details vary with the specifics of the basis choices. All 36 versions with a normal $GF(2^8)$ basis were examined (the others have a 5 XOR handicap), and also the all-polynomial version corresponding to the bases in [12], for comparison. The resulting improvement ranges from three to five XOR's: for most cases (23) it was three, for a dozen cases it was four, and it was five in only two cases.

While all these additional optimizations apply differently to the various basis choices, they tend to make the various versions more similar in size, with one exception: the extra shared factor in the normal $GF(2^8)$ inverter gives an advantage of five XOR's. Hence those cases using a polynomial basis for $GF(2^8)$ are effectively uncompetitive. The smallest (prior

to these optimizations) inverter saves $15 + 3$ XOR's in shared factors, 1 more in the $GF(2^4)$ inverter, and 3 more in the $GF(2^8)$ inverter, giving a total size of 66 XOR's and 36 AND's. (The bases of [12] give an inverter with 73 XOR's.)

The following tables show the size of the inverter when all of these optimizations have been applied; in addition to the number of XOR's shown, each inverter includes 36 AND's.

Poly.		XOR Gates		
$\nu =$		poly. $GF(2^2)$		norm.
$Cz + D$		$w = N$	$w = N^2$	$GF(2^2)$
N	0	67	67	67
N^2	0	67	67	67
N	N	67	67	67
N^2	N^2	67	67	67
N	1	67	67	67
N^2	N	67	67	67
N	N^2	68	68	67
N^2	1	67	68	67

Norm.		XOR Gates		
$\nu =$		poly. $GF(2^2)$		norm.
$Cz^4 + Dz$		$w = N$	$w = N^2$	$GF(2^2)$
N	0	66	66	66
0	N	66	66	66
N^2	0	66	66	66
0	N^2	66	66	66
N	1	66	66	66
1	N	66	66	66
N^2	1	66	66	66
1	N^2	66	66	66

The first table is for a polynomial $GF(2^4)$ basis, the second for a normal $GF(2^4)$ basis; both tables assume a normal basis for $GF(2^8)$, for the extra shared 4-bit factor. It is apparent that these low-level optimizations tend to even out the differences expected from the square-scale operation (compare with the tables on page 18). Using a polynomial $GF(2^4)$ basis costs at least one XOR (one less shared 2-bit factor), and a few cases cost one more. Because the variation in the inverter size is so small, the cost of changing between the standard representation and the S-box basis will be decisive.

4.5 Logic Gate Optimizations

Mathematically, computing the Galois inverse in $GF(2^8)$ breaks down into operations in $GF(2)$, i.e., the bitwise operations XOR and AND. However, it can be advantageous to consider other logical operations that give equivalent results.

For example, for the 0.13-μ CMOS standard cell library considered[13], a NAND gate is smaller than an AND gate. Since the AND output bits in the $GF(2^2)$ multiplier are always combined by pairs in a following XOR, then the AND gates can be replaced by NAND gates. That is, $[\,(a \otimes b) \oplus (c \otimes d)\,]$ is equivalent to $[\,(a$ NAND $b)$ XOR $(c$ NAND $d)\,]$. This gives a slight size saving.

Also, in this library an XNOR (not-exclusive-or, which really should be called NXOR) gate is the same size as an XOR gate. This is useful in the affine transformation of the S-box, where the addition of the constant $b =$ 0x63 requires applying a NOT to some of the output bits. In most cases, this can be done by replacing an XOR by an XNOR in the bit-matrix multiply, so is "free." But in some cases, such as when an output bit is given by a single input bit, the negation must be done explicitly with a NOT gate.

Note that the combination $[\,a \oplus b \oplus (a \otimes b)\,]$ is equivalent to $[\,a$ OR $b\,]$. In the few places in the inverter where this combination occurs, we can replace 2 XOR's and an AND by a single OR, a worthwhile substitution. But since we use NAND's, as mentioned above, then

the replacement would be a NOR, which is smaller than an OR. In fact, the NOR gate is smaller than an XOR gate, which means that even when a little rearrangement is required to get that combination, it is worthwhile even if the NOR ends up replacing only a single XOR.

These gate-level optimizations apply more or less equally to the different bases considered, so play only a minor role in the selection of a particular basis.

5 Choices of Representation

This algorithm involves several related representations, or isomorphisms, of Galois Fields. First, $GF(2^8)$ is considered as the set of bytes with the polynomial basis implied by the irreducible polynomial $q(x) = x^8+x^4+x^3+x+1$. Then $GF(2^8)/GF(2^4)$ is also considered as polynomials with coefficients in $GF(2^4)$, based on the irreducible polynomial $r(y) = y^2+y+\nu$. Similarly, $GF(2^4)/GF(2^2)$ uses a basis implied by the irreducible polynomial $s(z) = z^2+z+N$, and $GF(2^2)/GF(2)$ uses a root of $t(w) = w^2 + w + 1$. So each byte of information has two forms: the standard AES form (polynomial basis in 8 powers of A), and the subfield form in $GF(2^8)/GF(2^4)$ as a pair of 4-bit coefficients, each being (in $GF(2^4)/GF(2^2)$) a pair of two-bit coefficients, which in turn are coefficients in the basis for $GF(2^2)$.

One approach to using these two forms, as suggested by [11], is to convert each byte of the input block once, and do all of the AES algorithm in the new form, only converting back at the end of all the rounds. Since all the arithmetic in the AES algorithm is Galois arithmetic, this would work fine, provided the key was appropriately converted as well. However, the *MixColumns* step involves multiplying by constants that are simple in the standard basis (2 and 3, or A and $A+1$), but this simplicity is lost in the subfield basis. For example, scaling by 2 in the standard basis takes only 3 XOR's; the most efficient normal-basis version of this scaling requires 18 XOR's. Similar concerns arise in the inverse of *MixColumns*, used in decryption. This extra complication more than offsets the savings from delaying the basis change back to standard. Then, as in [12], the affine transformation can be combined with the basis change (see below). For these reasons, it is most efficient to change into the subfield basis on entering the S-box and to change back again on leaving it.

Each change of basis is in effect multiplication by an 8×8 bit matrix. Letting X refer to the matrix that converts from the subfield basis to the standard basis, then to compute the S-box function of a given byte, first we do a bit-matrix multiply by X^{-1} to change into the subfield basis, then calculate the Galois inverse by subfield arithmetic, then change basis back again by another bit-matrix multiply, by X. But this is followed directly by the affine transformation (substep 2), which includes another bit-matrix multiply, by the constant matrix M. (This can be regarded another change of basis, since M is invertible.) So we can combine the matrices into the product MX to save one bit-matrix multiply, as pointed out by [12]. Then adding the constant b completes the S-box function.

The inverse S-box function is similar, except the XOR with constant b comes first, followed by multiplication by the bit matrix $(MX)^{-1}$. Then after finding the inverse, we convert back to the standard basis through multiplication by the matrix X.

For each such constant-matrix multiply, the gate count can be reduced by "factoring out" combinations of input bits that are shared between different output bits (rows). One way to

do this is known as the "greedy algorithm," where at each stage one picks the combination of two input bits that is shared by the most output bits; that combination is then pre-computed in a single (XOR) gate, which output effectively becomes a new input to the remaining matrix multiply. The greedy algorithm is straightforward to implement, and generally gives good results.

But the greedy algorithm may not find the best result. We used a brute-force "tree search" approach to finding the optimal factoring. At each stage, each possible choice for factoring out a bit combination was tried, and the next stage examined recursively. Actually, some "pruning" of the tree is possible, when the bit-pair choice in the current stage is independent of that in the calling stage and had been checked previously. Appendix C gives the C program.

This method is guaranteed to find the minimal number of gates; the drawback is that one cannot tell how long it will take, due to the combinatorial complexity of the algorithm. For example, running on an Intel Xeon processor under Linux (without "pruning"), one particular 8×8 matrix took over 2 weeks, while many others took a fraction of a microsecond. (However, many of the matrices that took very long times had already been ruled poor candidates by the greedy algorithm, and could have been skipped.)

Using the "merged" S-box and inverse S-box of [12] complicates this picture, but reduces the hardware required overall when both encryption and decryption are needed. There, a block containing a single $GF(2^8)$ inverter can be used to compute either the S-box function or its inverse, depending on a selector signal. Given an input byte a, both $X^{-1}\,a$ and $(MX)^{-1}\,(a+b)$ are computed, with the first selected for encryption, the second for decryption. That selection is input into the inverter, and from the output byte c, both $(MX)\,c + b$ and $X\,c$ are computed; again the first is selected for encryption, the second for decryption.

With this merged approach, these basis-change matrix pairs can be optimized together, considering X^{-1} and $(MX)^{-1}$ together as a 16×8 matrix, and similarly (MX) and X, each pair taking one byte as input and giving two bytes as output. (Then $(MX)^{-1}\,(a + b)$ must be computed as $(MX)^{-1}\,a + [(MX)^{-1}\,b]$.) Combining in this way allows more commonality among rows (16 instead of 8) and so yields a more compact "factored" form. Of course, this also means the "tree search" optimizer has a much bigger task and longer run time. (Note: this is what actually induced our development of the "pruning" strategy, which typically gives a speedup factor of 10 to 20 times faster, enough to make full optimization feasible.)

The additive constant b of the affine transformation (or $(MX)^{-1}\,b$ for decryption), being an exclusive-OR with a known constant, just requires negating specific bits of the output of the basis change. (Actually, for the merged S-box, the multiplexors we use are themselves negating, so it is the bits other than those in b that need negating first.) As mentioned in Section 4.5, this usually involves replacing an XOR by an XNOR in the basis change (which is "free" since both XOR and XNOR are the same size in the CMOS library we consider), but sometimes this is not possible and a NOT gate is required.

At this time, not all of the matrices for all of the cases considered below have been fully optimized, but the data so far indicate how full optimization can improve on the greedy algorithm. For the architecture with separate encryptor and decryptor, all cases have been fully optimized: of 1728 matrices (8×8) optimized, 762 (44%) were improved by at least one XOR, and of those, 138 (18% of improved ones) were improved by two XOR's, and 11 (1.4% of improved ones) were improved by three XOR's. For the merged architecture, the

top 27 cases have been optimized: of 55 matrices (16×8) optimized, 24 (44%) were improved by one XOR, 10 (18%) were improved by two XOR's, and 6 (11%) were improved by three XOR's, so altogether 73% were improved.

We considered all of the subfield polynomial and normal bases that had a trace of unity. Over $GF(2^4)$, there are eight choices for ν that make $r(y) = y^2 + y + \nu$ irreducible, namely the four elements with the minimal polynomial (over $GF(2)$) $x^4 + x^3 + 1$, and the four elements with the minimal polynomial $x^4 + x^3 + x^2 + x + 1$. There are only two choices for N that make the polynomial $s(z) = z^2 + z + N$ irreducible over $GF(2^2)$, namely the two roots of $t(w) = w^2 + w + 1$. Each of these polynomials $r(y)$, $s(z)$, and $t(w)$ has two distinct roots, and for a polynomial basis we may choose either, or for a normal basis we use both. So including the choices for ν and N and the type of basis at each level, there are $(8 \times 3) \times (2 \times 3) \times (1 \times 3) = 432$ possible cases. (Note: the basis used in [12] corresponds to case number 252 in Appendix E.)

The most compact case was judged to be the one giving the least number of gates for the merged S-box architecture of [12], where a single inverter is shared for both encryption and decryption, using merged bit matrices X^{-1} and $(MX)^{-1}$ before the inverter, and (MX) and X after. The total gates include the two optimized 16×8 matrices, the two additions of the constant b, one inverter, and also the multiplexors. As it happens, the case giving the most compact circuit for this architecture also gives the most compact separate encryptor (with just X^{-1}, inverter, (MX), and b) and decryptor (accounting for the gate-level optimizations of Section 4.5).

(The envelope, please...)

The winner is case number 4 in the Appendix E table of all the cases. Here we will specify the relevant Galois elements in three forms: by our naming convention summarized in table D.3, by decimal and by hexadecimal numbers (in C notation), which refer to the representation in the standard basis (in powers of A). This case uses normal bases for all subfields. For $GF(2^8)/GF(2^4)$, the norm $\nu = \beta^8 = 236 =$ 0xEC, and $y = d = 255 =$ 0xFF, so the basis is $[d^{16}, d] =$ [0xFE,0xFF] (recall that for each of the normal bases, the sum of the two elements is the trace, which is unity). For $GF(2^4)/GF(2^2)$, $N = \Omega^2 = 188 =$ 0xBC and $z = \alpha^2 = 92 =$ 0x5C, so the basis is $[\alpha^8, \alpha^2] =$ [0x5D,0x5C]. And for $GF(2^2)$, $w = \Omega = 189 =$ 0xBD, so the basis is $[\Omega^2, \Omega] =$ [0xBC,0xBD].

For this case, $\nu = N^2 z$, i.e., $C = 0$ and $D = N^2$ in the table above, so this inverter is the smallest, consisting of 66 XOR's and 36 AND's. Incorporating the gate-level optimizations of Section 4.5 changes this to 56 XOR's, 34 NAND's, and 6 NOR's. The optimized versions of the merged basis change matrices have the following numbers of XOR's/XNOR's: $[X^{-1}\&(MX)^{-1}] = 20$, $[(MX)\&X] = 18$. Also, the additive constants of the affine transformation require 2 NOT's. For separate encryptor and decryptor, the optimized matrices have these sizes: $X^{-1} = 13$, $MX = 11$, $X = 13$, $(MX)^{-1} = 12$ (no NOT's required).

So the complete merged S-box and inverse, including inverter, transformation matrices, additive constant b, and multiplexors, totals 94 XOR/XNOR's + 34 NAND's + 6 NOR's + 2 NOT's + 16 MUX21I's (where MUX21I is a 2:1 selector and inverter [13]). Using the equivalencies 1 XOR/XNOR= $\frac{7}{4}$ NAND gates, 1 NOR= 1 NAND gate, 1 NOT= $\frac{3}{4}$ NAND gate, and 1 MUX21I= $\frac{7}{4}$ NAND gates [13], this S-box is equivalent in size to 234 NAND's, an improvement of 20% over the merged S-Box of [12] at 294 NAND's.

If separate encryptors and decryptors are preferable, then the S-box includes the bit

matrices X^{-1} and MX and inverter, totaling 80 XOR's + 34 NAND's + 6 NOR's, with equivalent size 180 NAND's; the inverse S-box uses $(MX)^{-1}$ and X and inverter, giving 81 XOR's + 34 NAND's + 6 NOR's, of size $181\frac{3}{4}$ NAND's.

Since we have not yet fully optimized the (16×8) matrices for all of the 432 possible cases, it is conceivable that one of the other cases could turn out to be better than case 4. We *have* optimized all cases whose estimated size, based on the greedy algorithm, was within 9 XOR's of the optimized size of case 4 (except in one case, where only 1 of the 2 matrices was optimized; it improved by 2 XOR's). So far, the best improvement in a single 16×8 matrix is 3 XOR's, and the best improvement in the pair of matrices for a single case is 5 XOR's. For some other case to be best, full optimization must improve a matrix pair, beyond what the greedy algorithm found, by at least 10 XOR's. We consider this highly unlikely, and so are confident that case 4 is indeed the best of all 432 cases.

6 Implementation Details

For the change of basis matrix, we want to change an element g of $GF(2^8)$, the standard AES representation as a byte of 8 bits $g_i \in GF(2)$, namely $g_7g_6g_5g_4g_3g_2g_1g_0$, meaning $g_7A^7 + g_6A^6 + g_5A^5 + g_4A^4 + g_3A^3 + g_2A^2 + g_1A + g_0$, into the new basis. Then in $GF(2^8)/GF(2^4)$, $g = \gamma_1 y^{16} + \gamma_0 y$, where for each element $\gamma \in GF(2^4)/GF(2^2)$, we have $\gamma = \Gamma_1 z^4 + \Gamma_0 z$, and each element $\Gamma \in GF(2^2)$ is considered a pair of bits b_1b_0, meaning $b_1w^2 + b_0w$. So the new byte representation $b_7b_6b_5b_4b_3b_2b_1b_0$ is related to the old by

$$\begin{aligned} & g_7A^7 + g_6A^6 + g_5A^5 + g_4A^4 + g_3A^3 + g_2A^2 + g_1A + g_0 \\ = \;& [(b_7w^2 + b_6w)z^4 + (b_5w^2 + b_4w)z]y^{16} + [(b_3w^2 + b_2w)z^4 + (b_1w^2 + b_0w)z]y \\ = \;& b_7w^2z^4y^{16} + b_6wz^4y^{16} + b_5w^2zy^{16} + b_4wzy^{16} + b_3w^2z^4y + b_2wz^4y + b_1w^2zy + b_0wzy \end{aligned}$$

The relevant arithmetic in $GF(2^8)$ (see Appendix D), using the standard A polynomial basis and logarithms base B, is: $y = 0\text{xFF} = B^7$, $z = 0\text{x5C} = B^{34}$, $w = 0\text{xBD} = B^{85}$, $y^{16} = B^{112} = 0\text{xFE}$, $z^4 = B^{136} = 0\text{x5D}$, $w^2 = B^{170} = 0\text{xBC}$, $w^2z^4y^{16} = B^{418} = B^{163} = 0\text{x64}$, $wz^4y^{16} = B^{333} = B^{78} = 0\text{x78}$, $w^2zy^{16} = B^{316} = B^{61} = 0\text{x6E}$, $wzy^{16} = B^{231} = 0\text{x8C}$, $w^2z^4y = B^{313} = B^{58} = 0\text{x68}$, $wz^4y = B^{228} = 0\text{x29}$, $w^2zy = B^{211} = 0\text{xDE}$, $wzy = B^{126} = 0\text{x60}$, so these become the columns of the basis change matrix X:

$$\begin{pmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix}$$

Then the reverse change of basis is given by X^{-1} (modulo 2):

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} g_7 \\ g_6 \\ g_5 \\ g_4 \\ g_3 \\ g_2 \\ g_1 \\ g_0 \end{pmatrix}$$

So to compute the S-box function of a given byte, first we do a bit-matrix multiply (by X^{-1}) to change into the basis for $GF(2^8)/GF(2^4)/GF(2^2)$, then calculate the inverse. Then change basis back again and perform the affine transformation, through another bit-matrix multiply by MX:

$$MX = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

and addition of the constant b.

The inverse S-box function is similar, except the XOR with constant b comes first. Then comes multiplication by the bit matrix

$$(MX)^{-1} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

And after finding the inverse, we convert back to the polynomial basis through multiplication by the matrix X.

The optimized versions of these matrices can be shown in product form to indicate the

factoring out of common bit combinations, as follows:

$$\left(\frac{X^{-1}}{(MX)^{-1}}\right) = \left(\begin{array}{cccccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
\hline
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&\\
0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&\\
0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&
\end{array}\right) \left(\frac{I}{0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1}\right)$$

$$\left(\frac{I}{\begin{array}{cccccccccccccc}0&0&0&1&0&0&0&0&0&0&0&0&0&1\\0&0&0&0&1&0&0&1&0&0&0&0&&\\0&0&0&0&0&1&0&0&1&0&0&0&&\\0&0&0&0&0&0&1&0&0&0&1&0&&\end{array}}\right) \left(\frac{I}{0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1}\right) \left(\frac{I}{\begin{array}{cccccccc}1&0&1&0&0&0&0&0\\1&0&0&1&0&0&0&0\\0&1&0&0&0&0&0&1\end{array}}\right)$$

$$\left(\frac{(MX)}{X}\right) = \left(\begin{array}{cccccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
\hline
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right) \left(\frac{I}{\begin{array}{cccccccccccccc}0&0&0&1&0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&0&1&0\end{array}}\right)$$

$$\left(\frac{I}{\begin{array}{cccccccccccc}0&0&1&0&0&0&0&1&0&0&0&\\0&0&1&0&0&0&1&0&0&0&0&\end{array}}\right) \left(\frac{I}{\begin{array}{cccccccc}1&0&0&0&1&0&0&0\\0&1&0&1&0&0&0&0\\0&1&0&0&0&0&0&1\\0&0&1&0&1&0&0&0\end{array}}\right)$$

where a horizontal line divides each matrix into two blocks, and I means an identity matrix of appropriate size. For each matrix row, the number of 1's, less one, is the number of two-input XOR gates needed for that row.

The implementation of the Galois inverter has mostly been given in Section 4.2 above, since normal bases are used at each level. There can be found the top-level inverter, the $GF(2^4)$ inverter and multiplier, the $GF(2^2)$ inverter (square, i.e., bit swap), multiplier, and scalers for both $N = w^2$ and $N^2 = w$. The combination of multiplication with scaling by

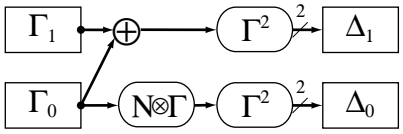

Figure 19: Normal $GF(2^4)$ square-scale: $\nu \otimes (\Gamma_1 Z^4 + \Gamma_0 Z)^2 = \Delta_1 Z^4 + \Delta_0 Z$

$N = w^2$ in $GF(2^2)$ is given by

$$\begin{aligned} f &= g_0 \otimes d_0 \\ N\Gamma\Delta &= [f \oplus ((g_1 \oplus g_0) \otimes (d_1 \oplus d_0))]w^2 + [f \oplus (g_1 \otimes d_1)]w \end{aligned}$$

The only other operation required is the square-scale operator in the normal basis $GF(2^4)$, as shown on page 18 for $C = 0$ and $D = N^2$, which is

$$\nu(Az^4 + Bz)^2 = [(A \oplus B)^2]z^4 + [N^2 \otimes B^2]z$$

where the squaring is free (see Figure 19).

Appendix A gives a C program that implements the S-box function (and its inverse) to illustrate the algorithm. This shows the hierarchical structure of the subfield approach, but does not include the low-level optimizations of Section 4.4. The output is a table that can be compared with the reference version in the file `boxes-ref.dat`, included in the "Reference code in ANSI C v2.2." link from The Rijndael Page:
`http://www.esat.kuleuven.ac.be/~rijmen/rijndael/`

Appendix B gives our compact implementation of the merged S-box and inverse as a Verilog module. All the low-level optimizations of Sections 4.4 and 4.5 are shown. These include: pre-computing sums of high and low parts of common factors for multipliers; in the $GF(2^8)$ inverter, using the bit sums of common factors to replace some terms in the scaled square of the sum of high and low inputs; similarly in the $GF(2^4)$ inverter; using NAND's instead of AND's, and replacing some XOR's and NAND's by NOR's.

We sucessfully tested this implementation using an FPGA (though our approach is really more appropriate for ASIC's). Specifically, we used an SRC-6E Reconfigurable Computer, which includes two Intel processors and two Virtex II FPGA's. As implemented on one FPGA, the function evaluation takes just one tick of the 100 MHz clock, the same amount of time needed for the table look-up approach.

We also implemented a complete AES encryptor/decryptor on this same system, using our S-box. Certain constraints (block RAM access) of this particular system prevent using table lookup for a fully unrolled pipelined version; 160 copies of the table (16 bytes/round$\times$10 rounds) would not fit. So for this system, our compact S-box allowed us to implement a fully pipelined encryptor/decryptor, where in the FPGA, effectively one block is processed for each clock tick.

7 Conclusion

The goal of this work is an algorithm to compute the S-box function of AES, that can be implemented in hardware with a minimal amount of circuitry. This should save a significant

amount of chip area in ASIC hardware versions of AES. Moreover, this area savings could allow many copies of the S-box circuit to fit on a chip, enough to "unroll" the loop of 10 rounds. This in turn would allow the AES process to be fully pipelined, increasing the rate of throughput significantly (for non-feedback modes of encryption), on smaller chips.

This algorithm employs the multi-level representation of arithmetic in $GF(2^8)$, similar to the previous compact implementation of Satoh et al[12]. Our work shows how this approach leads to a whole family of 432 implementations, depending on the particular isomorphism (basis) chosen, from which we found the best one. And in factoring the transformation (basis change) matrices for compactness, rather than rely on the greedy algorithm as in prior work, we fully optimized the matrices, using our tree search algorithm with pruning of redundant cases. This gave an improvement over the greedy algorithm in 73% of the (16×8) matrices that we optimized. Also new is the detailed description of this nested-subfield algorithm, including specification of all constants for each choice of representation.

Our best compact implementation gives an S-box that is 20% smaller than the previously most compact version of [12]. We have shown that none of the other 431 versions possible with this subfield approach is as small. This compact S-box could be useful for many future hardware implementations of AES, for a variety of security applications.

Acknowledgements

This work was supported by the National Security Agency.

Many thanks to Akashi Satoh for his patient and very helpful discussions.

References

[1] Pawel Chodowiec and Kris Gaj. Very compact FPGA implementation of the AES algorithm. In C.D. Walter et al., editor, *CHES 2003, LNCS 2779*, pages 319–333, 2003.

[2] Kimmo U. Jarvinen, Matti T. Tommiska, and Jorma O. Skytta. A fully pipelined memoryless 17.8 gbps AES128 encryptor. In *FPGA 03*. ACM, 2003.

[3] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge, New York, 1986.

[4] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, New York, 1977.

[5] Sumio Morioka and Akashi Satoh. A 10 Gbps full-AES crypto design with a twisted-BDD S-box architecture. In *IEEE International Conference on Computer Design*. IEEE, 2002.

[6] Sumio Morioka and Akashi Satoh. An optimized S-box circuit arthitecture for low power AES design. In *CHES2002, LNCS 2523*, pages 172–186, 2003.

[7] NIST. Commerce department announces winner of global information security competition. press release at `http://www.nist.gov/public_affairs/releases/g00-176.htm`, October 2000.

[8] NIST. Recommendation for block cipher modes of operation. Technical Report SP 800-38A, National Institute of Standards and Technology (NIST), December 2001.

[9] NIST. Specification for the ADVANCED ENCRYPTION STANDARD (AES). Technical Report FIPS PUB 197, National Institute of Standards and Technology (NIST), November 2001.

[10] Vincent Rijmen. Efficient implementation of the Rijndael S-box. available at `http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf`.

[11] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. Efficient Rijndael encryption implementation with composite field arithmetic. In *CHES2001, LNCS 2162*, pages 171–184, 2001.

[12] A. Satoh, S. Morioka, K. Takano, and Seiji Munetoh. A compact Rijndael hardware architecture with S-box optimization. In *Advances in Cryptology - ASIACRYPT 2001, LNCS 2248*, pages 239–254, 2001.

[13] Akashi Satoh. personal communication, July 2004.

[14] Nicholas Weaver and John Wawrzynek. High performance, compact AES implementations in Xilinx FPGAs. available at `http://www.cs.berkeley.edu/~nweaver/papers/AES_in_FPGAs.pdf`, September 2002.

[15] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC implementation of the AES Sboxes. In *CT-RSA 2002, LNCS 2271*, pages 67–78, 2002.

A S-box Algorithm in C

```
/* sbox.c
 *
 * by: David Canright
 *
 * illustrates compact implementation of AES S-box via subfield operations
 *   case # 4 : [d^16, d], [alpha^8, alpha^2], [Omega^2, Omega]
 *   nu = beta^8 = N^2*alpha^2, N = w^2
 */

#include <stdio.h>
#include <sys/types.h>

/* to convert between polynomial (A^7...1) basis A & normal basis X */
/* or to basis S which incorporates bit matrix of Sbox */
static int
    A2X[8] = {0x98, 0xF3, 0xF2, 0x48, 0x09, 0x81, 0xA9, 0xFF},
    X2A[8] = {0x64, 0x78, 0x6E, 0x8C, 0x68, 0x29, 0xDE, 0x60},
    X2S[8] = {0x58, 0x2D, 0x9E, 0x0B, 0xDC, 0x04, 0x03, 0x24},
    S2X[8] = {0x8C, 0x79, 0x05, 0xEB, 0x12, 0x04, 0x51, 0x53};

/* multiply in GF(2^2), using normal basis (Omega^2,Omega) */
int G4_mul( int x, int y ) {
    int a, b, c, d, e, p, q;

    a = (x & 0x2) >> 1; b = (x & 0x1);
    c = (y & 0x2) >> 1; d = (y & 0x1);
    e = (a ^ b) & (c ^ d);
    p = (a & c) ^ e;
    q = (b & d) ^ e;
    return ( (p<<1) | q );
    }

/* scale by N = Omega^2 in GF(2^2), using normal basis (Omega^2,Omega) */
int G4_scl_N( int x ) {
    int a, b, p, q;

    a = (x & 0x2) >> 1; b = (x & 0x1);
    p = b;
    q = a ^ b;
    return ( (p<<1) | q );
    }

/* scale by N^2 = Omega in GF(2^2), using normal basis (Omega^2,Omega) */
```

```
int G4_scl_N2( int x ) {
    int a, b, p, q;

    a = (x & 0x2) >> 1; b = (x & 0x1);
    p = a ^ b;
    q = a;
    return ( (p<<1) | q );
    }

/* square in GF(2^2), using normal basis (Omega^2,Omega) */
/* NOTE: inverse is identical */
int G4_sq( int x ) {
    int a, b;

    a = (x & 0x2) >> 1; b = (x & 0x1);
    return ( (b<<1) | a );
    }

/* multiply in GF(2^4), using normal basis (alpha^8,alpha^2) */
int G16_mul( int x, int y ) {
    int a, b, c, d, e, p, q;

    a = (x & 0xC) >> 2; b = (x & 0x3);
    c = (y & 0xC) >> 2; d = (y & 0x3);
    e = G4_mul( a ^ b, c ^ d );
    e = G4_scl_N(e);
    p = G4_mul( a, c ) ^ e;
    q = G4_mul( b, d ) ^ e;
    return ( (p<<2) | q );
    }

/* square & scale by nu in GF(2^4)/GF(2^2), normal basis (alpha^8,alpha^2) */
/*   nu = beta^8 = N^2*alpha^2, N = w^2 */
int G16_sq_scl( int x ) {
    int a, b, p, q;

    a = (x & 0xC) >> 2; b = (x & 0x3);
    p = G4_sq(a ^ b);
    q = G4_scl_N2(G4_sq(b));
    return ( (p<<2) | q );
    }

/* inverse in GF(2^4), using normal basis (alpha^8,alpha^2) */
int G16_inv( int x ) {
    int a, b, c, d, e, p, q;
```

```
    a = (x & 0xC) >> 2; b = (x & 0x3);
    c = G4_scl_N( G4_sq( a ^ b ) );
    d = G4_mul( a, b );
    e = G4_sq( c ^ d );   // really inverse, but same as square
    p = G4_mul( e, b );
    q = G4_mul( e, a );
    return ( (p<<2) | q );
    }

/* inverse in GF(2^8), using normal basis (d^16,d) */
int G256_inv( int x ) {
    int a, b, c, d, e, p, q;

    a = (x & 0xF0) >> 4; b = (x & 0x0F);
    c = G16_sq_scl( a ^ b );
    d = G16_mul( a, b );
    e = G16_inv( c ^ d );
    p = G16_mul( e, b );
    q = G16_mul( e, a );
    return ( (p<<4) | q );
    }

/* convert to new basis in GF(2^8) */
/* i.e., bit matrix multiply */
int G256_newbasis( int x, int b[] ) {
    int i, y = 0;

    for ( i=7; i >= 0; i-- ) {
        if ( x & 1 ) y ^= b[i];
        x >>= 1;
        }
    return ( y );
    }

/* find Sbox of n in GF(2^8) mod POLY */
int Sbox( int n ) {
     int t;

    t = G256_newbasis( n, A2X );
    t = G256_inv( t );
    t = G256_newbasis( t, X2S );
    return ( t ^ 0x63 );
    }
```

```
/* find inverse Sbox of n in GF(2^8) mod POLY */
int iSbox( int n ) {
     int t;

    t = G256_newbasis( n ^ 0x63, S2X );
    t = G256_inv( t );
    t = G256_newbasis( t, X2A );
    return ( t );
    }

/* compute tables of Sbox & its inverse; print 'em out */
int main () {
     int Sbox_tbl[256], iSbox_tbl[256], i, j;

    for (i = 0; i < 256; i++) {
        Sbox_tbl[i] = Sbox(i);
        iSbox_tbl[i] = iSbox(i);
        }
    printf ("char S[256] = {\n");
    for (i = 0; i < 16; i++) {
    for (j = 0; j < 16; j++) {
        printf ( "%3d, ", Sbox_tbl[i*16+j]);
        }
        printf ( "\n" );
        }
        printf ( "};\n\n" );
    printf ("char Si[256] = {\n");
    for (i = 0; i < 16; i++) {
    for (j = 0; j < 16; j++) {
        printf ( "%3d, ", iSbox_tbl[i*16+j]);
        }
        printf ( "\n" );
        }
        printf ( "};\n\n" );
    return(0);
    }
```

B S-box Algorithm in Verilog

```
/* S-box using all normal bases */
/*   case # 4 : [d^16, d], [alpha^8, alpha^2], [Omega^2, Omega] */
/*   beta^8 = N^2*alpha^2, N = w^2 */
/*   optimized using OR gates and NAND gates */

/* square in GF(2^2), using normal basis [Omega^2,Omega] */
/* inverse is the same as square in GF(2^2), using any normal basis */
module GF_SQ_2 ( A, Q );
    input  [1:0] A;
    output [1:0] Q;

    assign Q = { A[0], A[1] };
endmodule

/* scale by w = Omega in GF(2^2), using normal basis [Omega^2,Omega] */
module GF_SCLW_2 ( A, Q );
    input  [1:0] A;
    output [1:0] Q;

    assign Q = { (A[1] ^ A[0]), A[1] };
endmodule

/* scale by w^2 = Omega^2 in GF(2^2), using normal basis [Omega^2,Omega] */
module GF_SCLW2_2 ( A, Q );
    input  [1:0] A;
    output [1:0] Q;

    assign Q = { A[0], (A[1] ^ A[0]) };
endmodule

/* multiply in GF(2^2), shared factors, using normal basis [Omega^2,Omega] */
module GF_MULS_2 ( A, ab, B, cd, Q );
    input  [1:0] A;
    input  ab;
    input  [1:0] B;
    input  cd;
    output [1:0] Q;
    wire   abcd, p, q;

    assign abcd = ~(ab & cd);  /* note: ~& syntax for NAND won't compile */
    assign p = (~(A[1] & B[1])) ^ abcd;
    assign q = (~(A[0] & B[0])) ^ abcd;
    assign Q = { p, q };
```

```
endmodule

/* multiply & scale by N in GF(2^2), shared factors, basis [Omega^2,Omega] */
module GF_MULS_SCL_2 ( A, ab, B, cd, Q );
    input  [1:0] A;
    input  ab;
    input  [1:0] B;
    input  cd;
    output [1:0] Q;
    wire   t, p, q;

    assign t = ~(A[0] & B[0]);  /* note: ~& syntax for NAND won't compile */
    assign p = (~(ab & cd)) ^ t;
    assign q = (~(A[1] & B[1])) ^ t;
    assign Q = { p, q };
endmodule

/* inverse in GF(2^4)/GF(2^2), using normal basis [alpha^8, alpha^2] */
module GF_INV_4 ( A, Q );
    input  [3:0] A;
    output [3:0] Q;
    wire   [1:0] a, b, c, d, p, q;
    wire   sa, sb, sd;   /* for shared factors in multipliers */

    assign a = A[3:2];
    assign b = A[1:0];
    assign sa = a[1] ^ a[0];
    assign sb = b[1] ^ b[0];
/* optimize this section as shown below
    GF_MULS_2 abmul(a, sa, b, sb, ab);
    GF_SQ_2 absq( (a ^ b), ab2);
    GF_SCLW2_2 absclN( ab2, ab2N);
    GF_SQ_2 dinv( (ab ^ ab2N), d);
*/
    assign c = {   /* note: ~| syntax for NOR won't compile */
      ~(a[1] | b[1]) ^ (~(sa & sb)) ,
      ~(sa | sb) ^ (~(a[0] & b[0])) };
    GF_SQ_2 dinv( c, d);
/* end of optimization */
    assign sd = d[1] ^ d[0];
    GF_MULS_2 pmul(d, sd, b, sb, p);
    GF_MULS_2 qmul(d, sd, a, sa, q);
    assign Q = { p, q };
endmodule
```

```
/* square & scale by nu in GF(2^4)/GF(2^2), normal basis [alpha^8, alpha^2] */
/*   nu = beta^8 = N^2*alpha^2, N = w^2 */
module GF_SQ_SCL_4 ( A, Q );
    input  [3:0] A;
    output [3:0] Q;
    wire   [1:0] a, b, ab2, b2, b2N2;

    assign a = A[3:2];
    assign b = A[1:0];
    GF_SQ_2 absq(a ^ b,ab2);
    GF_SQ_2 bsq(b,b2);
    GF_SCLW_2 bmulN2(b2,b2N2);
    assign Q = { ab2, b2N2 };
endmodule

/* multiply in GF(2^4)/GF(2^2), shared factors, basis [alpha^8, alpha^2] */
module GF_MULS_4 ( A, a, Al, Ah, aa, B, b, Bl, Bh, bb, Q );
    input  [3:0] A;
    input  [1:0] a;
    input        Al;
    input        Ah;
    input        aa;
    input  [3:0] B;
    input  [1:0] b;
    input        Bl;
    input        Bh;
    input        bb;
    output [3:0] Q;
    wire   [1:0] ph, pl, ps, p;
    wire         t;

    GF_MULS_2 himul(A[3:2], Ah, B[3:2], Bh, ph);
    GF_MULS_2 lomul(A[1:0], Al, B[1:0], Bl, pl);
    GF_MULS_SCL_2 summul( a, aa, b, bb, p);
    assign Q = { (ph ^ p), (pl ^ p) };
endmodule

/* inverse in GF(2^8)/GF(2^4), using normal basis [d^16, d] */
module GF_INV_8 ( A, Q );
    input  [7:0] A;
    output [7:0] Q;
    wire   [3:0] a, b, c, d, p, q;
    wire   [1:0] sa, sb, sd, t;   /* for shared factors in multipliers */
    wire al, ah, aa, bl, bh, bb, dl, dh, dd;   /* for shared factors */
    wire c1, c2, c3;   /* for temp var */
```

```
    assign a = A[7:4];
    assign b = A[3:0];
    assign sa = a[3:2] ^ a[1:0];
    assign sb = b[3:2] ^ b[1:0];
    assign al = a[1] ^ a[0];
    assign ah = a[3] ^ a[2];
    assign aa = sa[1] ^ sa[0];
    assign bl = b[1] ^ b[0];
    assign bh = b[3] ^ b[2];
    assign bb = sb[1] ^ sb[0];
/* optimize this section as shown below
    GF_MULS_4 abmul(a, sa, al, ah, aa, b, sb, bl, bh, bb, ab);
    GF_SQ_SCL_4 absq( (a ^ b), ab2);
    GF_INV_4 dinv( (ab ^ ab2), d);
*/
    assign c1 = ~(ah & bh);
    assign c2 = ~(sa[0] & sb[0]);
    assign c3 = ~(aa & bb);
    assign c = {   /* note: ~| syntax for NOR won't compile */
      (~(sa[0] | sb[0]) ^ (~(a[3] & b[3]))) ^ c1 ^ c3 ,
      (~(sa[1] | sb[1]) ^ (~(a[2] & b[2]))) ^ c1 ^ c2 ,
      (~(al | bl) ^ (~(a[1] & b[1]))) ^ c2 ^ c3 ,
      (~(a[0] | b[0]) ^ (~(al & bl))) ^ (~(sa[1] & sb[1])) ^ c2 };
    GF_INV_4 dinv( c, d);
/* end of optimization */
    assign sd = d[3:2] ^ d[1:0];
    assign dl = d[1] ^ d[0];
    assign dh = d[3] ^ d[2];
    assign dd = sd[1] ^ sd[0];
    GF_MULS_4 pmul(d, sd, dl, dh, dd, b, sb, bl, bh, bb, p);
    GF_MULS_4 qmul(d, sd, dl, dh, dd, a, sa, al, ah, aa, q);
    assign Q = { p, q };
endmodule

/* MUX21I is an inverting 2:1 multiplexor */
module MUX21I ( A, B, s, Q );
    input        A;
    input        B;
    input        s;
    output       Q;
    assign Q = ~ ( s ? A : B );  /* mock-up for FPGA implementation */
endmodule

/* select and invert (NOT) byte, using MUX21I */
```

```
module SELECT_NOT_8 ( A, B, s, Q );
    input  [7:0] A;
    input  [7:0] B;
    input        s;
    output [7:0] Q;
    MUX21I m7(A[7],B[7],s,Q[7]);
    MUX21I m6(A[6],B[6],s,Q[6]);
    MUX21I m5(A[5],B[5],s,Q[5]);
    MUX21I m4(A[4],B[4],s,Q[4]);
    MUX21I m3(A[3],B[3],s,Q[3]);
    MUX21I m2(A[2],B[2],s,Q[2]);
    MUX21I m1(A[1],B[1],s,Q[1]);
    MUX21I m0(A[0],B[0],s,Q[0]);
endmodule

/* find either Sbox or its inverse in GF(2^8), by Canright Algorithm */
module bSbox ( A, encrypt, Q );
    input  [7:0] A;
    input        encrypt;  /* 1 for Sbox, 0 for inverse Sbox */
    output [7:0] Q;
    wire   [7:0] B, C, D, X, Y, Z;
    wire R1, R2, R3, R4, R5, R6, R7, R8, R9;
    wire T1, T2, T3, T4, T5, T6, T7, T8, T9, T10;

/* change basis from GF(2^8) to GF(2^8)/GF(2^4)/GF(2^2) */
/* combine with bit inverse matrix multiply of Sbox */
assign  R1  = A[7] ^ A[5] ;
assign  R2  = A[7] ~^ A[4] ;
assign  R3  = A[6] ^ A[0] ;
assign  R4  = A[5] ~^  R3  ;
assign  R5  = A[4] ^  R4  ;
assign  R6  = A[3] ^ A[0] ;
assign  R7  = A[2] ^  R1  ;
assign  R8  = A[1] ^  R3  ;
assign  R9  = A[3] ^  R8  ;
assign B[7] =  R7  ~^  R8  ;
assign B[6] =  R5  ;
assign B[5] = A[1] ^  R4  ;
assign B[4] =  R1  ~^  R3  ;
assign B[3] = A[1] ^  R2  ^  R6  ;
assign B[2] = ~ A[0] ;
assign B[1] =  R4  ;
assign B[0] = A[2] ~^  R9  ;
assign Y[7] =  R2  ;
assign Y[6] = A[4] ^  R8  ;
```

```
assign Y[5] = A[6] ^ A[4] ;
assign Y[4] =  R9  ;
assign Y[3] = A[6] ~^  R2  ;
assign Y[2] =  R7  ;
assign Y[1] = A[4] ^  R6  ;
assign Y[0] = A[1] ^  R5  ;
    SELECT_NOT_8 sel_in( B, Y, encrypt, Z );
    GF_INV_8 inv( Z, C );
/* change basis back from GF(2^8)/GF(2^4)/GF(2^2) to GF(2^8) */
assign  T1  = C[7] ^ C[3] ;
assign  T2  = C[6] ^ C[4] ;
assign  T3  = C[6] ^ C[0] ;
assign  T4  = C[5] ~^ C[3] ;
assign  T5  = C[5] ~^  T1  ;
assign  T6  = C[5] ~^ C[1] ;
assign  T7  = C[4] ~^  T6  ;
assign  T8  = C[2] ^  T4  ;
assign  T9  = C[1] ^  T2  ;
assign  T10  =  T3  ^  T5  ;
assign D[7] =  T4  ;
assign D[6] =  T1  ;
assign D[5] =  T3  ;
assign D[4] =  T5  ;
assign D[3] =  T2  ^  T5  ;
assign D[2] =  T3  ^  T8  ;
assign D[1] =  T7  ;
assign D[0] =  T9  ;
assign X[7] = C[4] ~^ C[1] ;
assign X[6] = C[1] ^  T10  ;
assign X[5] = C[2] ^  T10  ;
assign X[4] = C[6] ~^ C[1] ;
assign X[3] =  T8  ^  T9  ;
assign X[2] = C[7] ~^  T7  ;
assign X[1] =  T6  ;
assign X[0] = ~ C[2] ;
    SELECT_NOT_8 sel_out( D, X, encrypt, Q );
endmodule

/* test program: put Sbox output into register */
module Sbox_r ( A, S, Si, CLK );
    input  [7:0] A;
    output [7:0] S;
    output [7:0] Si;
    input  CLK  /* synthesis syn_noclockbuf=1 */ ;
    reg    [7:0] S;
```

```
    reg    [7:0] Si;
    wire   [7:0] s;
    wire   [7:0] si;

    bSbox sbe(A,1,s);
    bSbox sbd(A,0,si);
    always @ (posedge CLK) begin
    S <= s;
    Si <= si;
    end
endmodule
```

C Bit-Matrix Optimizer in C

```
/* bestboth.c
 *
 * by: David Canright
 *
 * for each input basis, and each of 4 transformation matrices,
 * takes bit matrix and finds equivalent with minimum # of gates
 *  combining both input matrices, and both output matrices
 * NOTE: matrix input order is: [A2X, X2A, X2S, S2X]
 *
 * input should have lines of the form:
 hexstring num
 * where hexstring contains all 4 matrices, num is an ID#, e.g.:
98F3F2480981A9FF64786E8C6829DE60582D9E0BDC0403248C7905EB12045153  4
 * for which the output should be:
basis #  4:
   A2X: 98F3F2480981A9FF   S2X: 8C7905EB12045153
 ncols =  8, gates = 42
  A2Xb: 00000000000128041008102240088080001
  S2Xb: 00280062000001000088000001020440l0
 [0,2],  [0,3],  [1,7],  [2,10],  [3,11],  [4,7],  [5,8],  [6,10],  [4,15],
 ncols = 17, gates = 20
   X2S: 582D9E0BDC040324   X2A: 64786E8C6829DE60
 ncols =  8, gates = 38
  X2Sb: 00000000000000004008248018000204010O
  X2Ab: 041000800021D000000000000000204080860
 [0,4],  [1,3],  [1,7],  [2,4],  [2,8],  [2,6],  [3,13],  [5,11],  [6,9],  [10,12],
 ncols = 18, gates = 18
***bestgates   4 =    38   =   20 +   18
 * which, for each matrix pair, shows the original versions (8 columns),
 *   the optimized versions, and a list of index pairs for precomputed XORs,
 *   which correspond to new columns. Also shown: # XOR gates required.
 * Note: a "quick" test case is:
F1261450CA86D330C502A8BF412B3590352582D03974323C65C4836C69953380    0
 *
 * uses pruning algorithm to eliminate redundant cases; minimal memory copying
 */

#include <stdio.h>
#include <string.h>
#define N 8

/* gatematrix is a structure with an array of 16-bit columns,
 list of indices (used in pairs), number of columns, and number of gates*/
```

```
typedef struct gatematrix
     { unsigned int mat[128]; char ind[256]; int n; int g; }
     GateMat;

static unsigned int share[65536];
static GateMat test;

/* blockPrint prints columns and index pairs for matrix pair */
void blockPrint (GateMat *p, const char *tag1, const char *tag2)
{
     int i;

     printf ("%6s: ", tag1);
     for (i = 0; i < p->n; i++)
          printf ("%02X", (p->mat[i]) & 0XFF );
     if ((p->n) > N) printf ("\n");
     printf ("%6s: ", tag2);
     for (i = 0; i < p->n; i++)
          printf ("%02X", ((p->mat[i]) & 0XFF00) >> 8 );
     if ((p->n) > N) printf ("\n");
     for (i = 0; i < (p->n)-N; i++)
          printf (" [%1d,%1d], ", p->ind[2*i], p->ind[2*i+1]);
     printf ("\n ncols = %2d, gates = %2d\n", p->n, p->g);

} /* end blockPrint */

/* copyMat copies from one to another*/
void copyMat (GateMat *p, GateMat *q)
{
     int i, n;

     n = q->n = p->n;
     q->g = p->g;
     memcpy( q->mat, p->mat, n * sizeof(unsigned int));
     memcpy( q->ind, p->ind, (n - N)*2);
} /* end copyMat */

/*
* bestgates is recursive:
*   takes current matrix, tries all possibilities of adding a gate
*   returns best # of gates
* p points to test matrix on input, and used to store output.
* tree search is pruned if this set of columns previously tried
*/
void bestgates ()
```

```
{
     char indb[256];
     int gb, nb, ci, cj;
     int i, j, n, c, g, io, jo;
     int nm, np, n2, n2p, t;

     gb = 1024;  /* best # gates, start high */
     n = test.n; g = test.g;
     nm=n-1; np=n+1; n2=2*(n-N); n2p=n2+1;
     if (n==N) io = jo = 0;  /* if orig matrix, no "old" index pair */
     else { io = (test.ind[n2-2]); jo = (test.ind[n2-1]); }
     for (i=0;i<nm;i++)   /* for each pair of columns */
     for (j=i+1;j<n;j++) {
     c = (test.mat[i]) & (test.mat[j]);
     if (t=share[c]) {   /* if can share a gate */
          if (i<io && j!=io && j!=jo && j<nm)  /* if prior, indep. pair */
               continue; /* then been there, done that; skip to next j */
          test.n = np;
          test.g = g - t;
          ci = test.mat[i];  /* save current columns */
          cj = test.mat[j];
          test.mat[i] ^= c;  /* update to new columns */
          test.mat[j] ^= c;
          test.mat[n] = c;
          test.ind[n2] = i;
          test.ind[n2p] = j;
          bestgates();       /* recurse with new matrix */
          test.mat[i] = ci;  /* restore current columns */
          test.mat[j] = cj;
          if ( test.g < gb ) {  /* if best yet, save data */
               memcpy( indb, test.ind+n2, (test.n - n)*2);
               nb = test.n;
               gb = test.g;
               }
          }
     }   /* end columns loop */
     if (gb < 1024) {   /* if improved, return best data */
          memcpy( test.ind+n2, indb, (nb - n)*2);
          test.n = nb;
          test.g = gb;
          }
/*      else {printf("%3d [%2d]",n,g); fflush(stdout);} */
} /* end bestgates */

/* bestmat reconstructs best matrix */
```

```
void bestmat (GateMat *p)
{
     int i, j, n, c;
     int nm, np, n2, n2p, t;
     GateMat best;

     n = test.n;
     p->g = test.g;
     for (i=0;i<N;i++) test.mat[i] = p->mat[i];
     for (n=0;n<(test.n-N);n++) {
     i = test.ind[n*2];
     j = test.ind[n*2+1];
     c = (test.mat[i]) & (test.mat[j]);
     test.mat[i] ^= c;
     test.mat[j] ^= c;
     test.mat[n+N] = c;
     }
} /* end bestmat */

/* main */
int main( int argc, char *argv[] ){
     char   line[256];
     char   name[4][4] = {"A2X", "X2S", "S2X", "X2A", };
     char   bname[4][5] = {"A2Xb", "X2Sb", "S2Xb", "X2Ab", };
     long int i, j, k, n, nid, gt;
     unsigned u;
     int InitMat[32];
     GateMat orig[2];

/* share[i] is initialized to 0 if # bits < 2 */
share[0] = 0;
for (i=1;i<65536;i++) {
     k=0;
     for (j=i&0xFFFF; j; j >>=1) k += j&1;
     share[i] = k-1;
     }

while ( fgets( line, 256, stdin ) == line ) {
     for ( i=0; i < 32; i++ ) {   /* read matrices, ID number */
          sscanf( line+2*i, "%02X", &u );
          InitMat[i] = u;
          }
          sscanf( line+65, "%d", &nid );
     printf("\nbasis #%3d:\n", nid);
```

```
/* NOTE: matrix input order is: [A2X, X2A, X2S, S2X] */
     for (i=0;i<8;i++) {    /* combine input pair; combine output pair */
          (orig[0]).mat[i] = InitMat[8*0+i] | (InitMat[8*3+i] <<8) ;
          (orig[1]).mat[i] = InitMat[8*2+i] | (InitMat[8*1+i] <<8) ;
          }

     gt = 0;
     for (k=0;k<2;k++) {   /* for each matrix pair */
     (orig[k]).n = 8;   /* initialize # columns, # gates */
     for (i=j=0; i<8; i++) j += share[ (orig[k]).mat[i] ];
     (orig[k]).g = j - 8;
     blockPrint (&(orig[k]), name[k], name[k+2]);
     fflush(stdout);

     copyMat(&(orig[k]), &test);
     bestgates();   /* optimize */
     bestmat(&(orig[k]));

     blockPrint (&test, bname[k], bname[k+2]);
     fflush(stdout);
     gt += test.g;   /* total # gates */

     }
     printf("***bestgates %3d = %5d   =%5d +%5d\n",
       nid, gt, (orig[0]).g, (orig[1]).g );
     fflush(stdout);
     }

    return(0);
    } /* end main */
```

D Tables for $GF(2^8)$

D.1 Logarithm Table

For each number in decimal, hexadecimal, and binary, gives the logarithm base B in $GF(2^8)$, using the polynomial basis from the root A of $q(x) = x^8 + x^4 + x^3 + x + 1$, where $B = A + 1$. (See Table D.3 for an explanation of the names.)

dec	hex	binary	$\log_B$	name
0	00	00000000	$-\infty$	0
1	01	00000001	0	1
2	02	00000010	25	A
3	03	00000011	1	B
4	04	00000100	50	A^2
5	05	00000101	2	B^2
6	06	00000110	26	C^2
7	07	00000111	198	F^{64}
8	08	00001000	75	E^{64}
9	09	00001001	199	D^8
10	0A	00001010	27	F
11	0B	00001011	104	C^8
12	0C	00001100	51	γ
13	0D	00001101	238	β
14	0E	00001110	223	b^{32}
15	0F	00001111	3	K
16	10	00010000	100	A^4
17	11	00010001	4	B^4
18	12	00010010	224	d^{32}
19	13	00010011	14	d^2
20	14	00010100	52	C^4
21	15	00010101	141	F^{128}
22	16	00010110	129	K^{128}
23	17	00010111	239	b^{16}
24	18	00011000	76	g^4
25	19	00011001	113	J^{16}
26	1A	00011010	8	B^8
27	1B	00011011	200	A^8
28	1C	00011100	248	D
29	1D	00011101	105	E^8
30	1E	00011110	28	d^4
31	1F	00011111	193	d^{64}

dec	hex	binary	$\log_B$	name
32	20	00100000	125	R^{128}
33	21	00100001	194	s^{64}
34	22	00100010	29	j^{32}
35	23	00100011	181	h^2
36	24	00100100	249	k^2
37	25	00100101	185	a^{64}
38	26	00100110	39	f^8
39	27	00100111	106	M^{128}
40	28	00101000	77	M^{16}
41	29	00101001	228	f
42	2A	00101010	166	M^8
43	2B	00101011	114	f^{128}
44	2C	00101100	154	M^{32}
45	2D	00101101	201	f^2
46	2E	00101110	9	l
47	2F	00101111	120	T^8
48	30	00110000	101	m^{32}
49	31	00110001	47	c^{16}
50	32	00110010	138	N^{128}
51	33	00110011	5	r
52	34	00110100	33	l^{32}
53	35	00110101	15	T
54	36	00110110	225	T^{32}
55	37	00110111	36	l^4
56	38	00111000	18	l^2
57	39	00111001	240	T^{16}
58	3A	00111010	130	r^{128}
59	3B	00111011	69	N^{64}
60	3C	00111100	53	M^{64}
61	3D	00111101	147	f^4
62	3E	00111110	218	h
63	3F	00111111	142	j^{16}

dec	hex	binary	$\log_B$	name
64	40	01000000	150	E^{128}
65	41	01000001	143	D^{16}
66	42	01000010	219	L^4
67	43	01000011	189	L^{64}
68	44	01000100	54	F^2
69	45	01000101	208	C^{16}
70	46	01000110	206	G^{16}
71	47	01000111	148	H^4
72	48	01001000	19	g
73	49	01001001	92	J^4
74	4A	01001010	210	E^{16}
75	4B	01001011	241	D^2
76	4C	01001100	64	B^{64}
77	4D	01001101	70	A^{64}
78	4E	01001110	131	d^{128}
79	4F	01001111	56	d^8
80	50	01010000	102	γ^2
81	51	01010001	221	β^2
82	52	01010010	253	b^2
83	53	01010011	48	K^{16}
84	54	01010100	191	b^{64}
85	55	01010101	6	K^2
86	56	01010110	139	J^{128}
87	57	01010111	98	g^{32}
88	58	01011000	179	G^4
89	59	01011001	37	H
90	5A	01011010	226	J^{32}
91	5B	01011011	152	g^8
92	5C	01011100	34	α^2
93	5D	01011101	136	α^8
94	5E	01011110	145	A^{16}
95	5F	01011111	16	B^{16}

dec	hex	binary	$\log_B$	name
96	60	01100000	126	k^{128}
97	61	01100001	110	a^{16}
98	62	01100010	72	l^8
99	63	01100011	195	T^{64}
100	64	01100100	163	j^4
101	65	01100101	182	h^{64}
102	66	01100110	30	T^2
103	67	01100111	66	l^{64}
104	68	01101000	58	j^{64}
105	69	01101001	107	h^4
106	6A	01101010	40	r^8
107	6B	01101011	84	N^4
108	6C	01101100	250	R
109	6D	01101101	133	s^{128}
110	6E	01101110	61	S^{64}
111	6F	01101111	186	n^{64}
112	70	01110000	43	m
113	71	01110001	121	c^{128}
114	72	01110010	10	r^2
115	73	01110011	21	N
116	74	01110100	155	a^4
117	75	01110101	159	k^{32}
118	76	01110110	94	c^{32}
119	77	01110111	202	m^{64}
120	78	01111000	78	f^{16}
121	79	01111001	212	M
122	7A	01111010	172	m^4
123	7B	01111011	229	c^2
124	7C	01111100	243	k^4
125	7D	01111101	115	a^{128}
126	7E	01111110	167	S^8
127	7F	01111111	87	n^8

dec	hex	binary	$\log_B$	name
128	80	10000000	175	R^{16}
129	81	10000001	88	s^8
130	82	10000010	168	N^8
131	83	10000011	80	r^{16}
132	84	10000100	244	S
133	85	10000101	234	n
134	86	10000110	214	h^8
135	87	10000111	116	j^{128}
136	88	10001000	79	S^{16}
137	89	10001001	174	n^{16}
138	8A	10001010	233	S^2
139	8B	10001011	213	n^2
140	8C	10001100	231	k^8
141	8D	10001101	230	a
142	8E	10001110	173	h^{16}
143	8F	10001111	232	j
144	90	10010000	44	s^4
145	91	10010001	215	R^8
146	92	10010010	117	n^{128}
147	93	10010011	122	S^{128}
148	94	10010100	235	R^4
149	95	10010101	22	s^2
150	96	10010110	11	s
151	97	10010111	245	R^2
152	98	10011000	89	m^8
153	99	10011001	203	c^4
154	9A	10011010	95	R^{32}
155	9B	10011011	176	s^{16}
156	9C	10011100	156	f^{32}
157	9D	10011101	169	M^2
158	9E	10011110	81	N^{16}
159	9F	10011111	160	r^{32}

dec	hex	binary	$\log_B$	name
160	A0	10100000	127	b^{128}
161	A1	10100001	12	K^4
162	A2	10100010	246	L
163	A3	10100011	111	L^{16}
164	A4	10100100	23	J
165	A5	10100101	196	g^{64}
166	A6	10100110	73	H^{64}
167	A7	10100111	236	G
168	A8	10101000	216	F^8
169	A9	10101001	67	C^{64}
170	AA	10101010	31	D^{32}
171	AB	10101011	45	E
172	AC	10101100	164	H^{32}
173	AD	10101101	118	G^{128}
174	AE	10101110	123	L^{128}
175	AF	10101111	183	L^8
176	B0	10110000	204	γ^4
177	B1	10110001	187	β^4
178	B2	10110010	62	D^{64}
179	B3	10110011	90	E^2
180	B4	10110100	251	b^4
181	B5	10110101	96	K^{32}
182	B6	10110110	177	F^{16}
183	B7	10110111	134	C^{128}
184	B8	10111000	59	G^{64}
185	B9	10111001	82	H^{16}
186	BA	10111010	161	C^{32}
187	BB	10111011	108	F^4
188	BC	10111100	170	Ω^2
189	BD	10111101	85	Ω
190	BE	10111110	41	H^8
191	BF	10111111	157	G^{32}

dec	hex	binary	$\log_B$	name
192	C0	11000000	151	c^8
193	C1	11000001	178	m^{16}
194	C2	11000010	135	T^{128}
195	C3	11000011	144	l^{16}
196	C4	11000100	97	s^{32}
197	C5	11000101	190	R^{64}
198	C6	11000110	220	a^{32}
199	C7	11000111	252	k
200	C8	11001000	188	c^{64}
201	C9	11001001	149	m^{128}
202	CA	11001010	207	k^{16}
203	CB	11001011	205	a^2
204	CC	11001100	55	a^8
205	CD	11001101	63	k^{64}
206	CE	11001110	91	h^{32}
207	CF	11001111	209	j^2
208	D0	11010000	83	M^4
209	D1	11010001	57	f^{64}
210	D2	11010010	132	l^{128}
211	D3	11010011	60	T^4
212	D4	11010100	65	r^{64}
213	D5	11010101	162	N^{32}
214	D6	11010110	109	h^{128}
215	D7	11010111	71	j^8
216	D8	11011000	20	r^4
217	D9	11011001	42	N^2
218	DA	11011010	158	S^{32}
219	DB	11011011	93	n^{32}
220	DC	11011100	86	m^2
221	DD	11011101	242	c
222	DE	11011110	211	S^4
223	DF	11011111	171	n^4

dec	hex	binary	$\log_B$	name
224	E0	11100000	68	α^4
225	E1	11100001	17	α
226	E2	11100010	146	H^{128}
227	E3	11100011	217	G^2
228	E4	11100100	35	A^{32}
229	E5	11100101	32	B^{32}
230	E6	11100110	46	J^2
231	E7	11100111	137	g^{128}
232	E8	11101000	180	E^4
233	E9	11101001	124	D^{128}
234	EA	11101010	184	J^8
235	EB	11101011	38	g^2
236	EC	11101100	119	β^8
237	ED	11101101	153	γ^8
238	EE	11101110	227	D^4
239	EF	11101111	165	E^{32}
240	F0	11110000	103	G^8
241	F1	11110001	74	H^2
242	F2	11110010	237	L^2
243	F3	11110011	222	L^{32}
244	F4	11110100	197	J^{64}
245	F5	11110101	49	g^{16}
246	F6	11110110	254	b
247	F7	11110111	24	K^8
248	F8	11111000	13	C
249	F9	11111001	99	F^{32}
250	FA	11111010	140	A^{128}
251	FB	11111011	128	B^{128}
252	FC	11111100	192	K^{64}
253	FD	11111101	247	b^8
254	FE	11111110	112	d^{16}
255	FF	11111111	7	d

D.2 Antilogarithm Table

Same information as previous table, but ordered by logarithm base B.

dec	hex	binary	$\log_B$	name
0	00	00000000	$-\infty$	0
1	01	00000001	0	1
3	03	00000011	1	B
5	05	00000101	2	B^2
15	0F	00001111	3	K
17	11	00010001	4	B^4
51	33	00110011	5	r
85	55	01010101	6	K^2
255	FF	11111111	7	d
26	1A	00011010	8	B^8
46	2E	00101110	9	l
114	72	01110010	10	r^2
150	96	10010110	11	s
161	A1	10100001	12	K^4
248	F8	11111000	13	C
19	13	00010011	14	d^2
53	35	00110101	15	T
95	5F	01011111	16	B^{16}
225	E1	11100001	17	α
56	38	00111000	18	l^2
72	48	01001000	19	g
216	D8	11011000	20	r^4
115	73	01110011	21	N
149	95	10010101	22	s^2
164	A4	10100100	23	J
247	F7	11110111	24	K^8
2	02	00000010	25	A
6	06	00000110	26	C^2
10	0A	00001010	27	F
30	1E	00011110	28	d^4
34	22	00100010	29	j^{32}
102	66	01100110	30	T^2

dec	hex	binary	$\log_B$	name
170	AA	10101010	31	D^{32}
229	E5	11100101	32	B^{32}
52	34	00110100	33	l^{32}
92	5C	01011100	34	α^2
228	E4	11100100	35	A^{32}
55	37	00110111	36	l^4
89	59	01011001	37	H
235	EB	11101011	38	g^2
38	26	00100110	39	f^8
106	6A	01101010	40	r^8
190	BE	10111110	41	H^8
217	D9	11011001	42	N^2
112	70	01110000	43	m
144	90	10010000	44	s^4
171	AB	10101011	45	E
230	E6	11100110	46	J^2
49	31	00110001	47	c^{16}
83	53	01010011	48	K^{16}
245	F5	11110101	49	g^{16}
4	04	00000100	50	A^2
12	0C	00001100	51	γ
20	14	00010100	52	C^4
60	3C	00111100	53	M^{64}
68	44	01000100	54	F^2
204	CC	11001100	55	a^8
79	4F	01001111	56	d^8
209	D1	11010001	57	f^{64}
104	68	01101000	58	j^{64}
184	B8	10111000	59	G^{64}
211	D3	11010011	60	T^4
110	6E	01101110	61	S^{64}
178	B2	10110010	62	D^{64}

dec	hex	binary	$\log_B$	name
205	CD	11001101	63	k^{64}
76	4C	01001100	64	B^{64}
212	D4	11010100	65	r^{64}
103	67	01100111	66	l^{64}
169	A9	10101001	67	C^{64}
224	E0	11100000	68	α^4
59	3B	00111011	69	N^{64}
77	4D	01001101	70	A^{64}
215	D7	11010111	71	j^8
98	62	01100010	72	l^8
166	A6	10100110	73	H^{64}
241	F1	11110001	74	H^2
8	08	00001000	75	E^{64}
24	18	00011000	76	g^4
40	28	00101000	77	M^{16}
120	78	01111000	78	f^{16}
136	88	10001000	79	S^{16}
131	83	10000011	80	r^{16}
158	9E	10011110	81	N^{16}
185	B9	10111001	82	H^{16}
208	D0	11010000	83	M^4
107	6B	01101011	84	N^4
189	BD	10111101	85	Ω
220	DC	11011100	86	m^2
127	7F	01111111	87	n^8
129	81	10000001	88	s^8
152	98	10011000	89	m^8
179	B3	10110011	90	E^2
206	CE	11001110	91	h^{32}
73	49	01001001	92	J^4
219	DB	11011011	93	n^{32}
118	76	01110110	94	c^{32}

dec	hex	binary	$\log_B$	name
154	9A	10011010	95	R^{32}
181	B5	10110101	96	K^{32}
196	C4	11000100	97	s^{32}
87	57	01010111	98	g^{32}
249	F9	11111001	99	F^{32}
16	10	00010000	100	A^4
48	30	00110000	101	m^{32}
80	50	01010000	102	γ^2
240	F0	11110000	103	G^8
11	0B	00001011	104	C^8
29	1D	00011101	105	E^8
39	27	00100111	106	M^{128}
105	69	01101001	107	h^4
187	BB	10111011	108	F^4
214	D6	11010110	109	h^{128}
97	61	01100001	110	a^{16}
163	A3	10100011	111	L^{16}
254	FE	11111110	112	d^{16}
25	19	00011001	113	J^{16}
43	2B	00101011	114	f^{128}
125	7D	01111101	115	a^{128}
135	87	10000111	116	j^{128}
146	92	10010010	117	n^{128}
173	AD	10101101	118	G^{128}
236	EC	11101100	119	β^8
47	2F	00101111	120	T^8
113	71	01110001	121	c^{128}
147	93	10010011	122	S^{128}
174	AE	10101110	123	L^{128}
233	E9	11101001	124	D^{128}
32	20	00100000	125	R^{128}
96	60	01100000	126	k^{128}

dec	hex	binary	$\log_B$	name
160	A0	10100000	127	b^{128}
251	FB	11111011	128	B^{128}
22	16	00010110	129	K^{128}
58	3A	00111010	130	r^{128}
78	4E	01001110	131	d^{128}
210	D2	11010010	132	l^{128}
109	6D	01101101	133	s^{128}
183	B7	10110111	134	C^{128}
194	C2	11000010	135	T^{128}
93	5D	01011101	136	α^8
231	E7	11100111	137	g^{128}
50	32	00110010	138	N^{128}
86	56	01010110	139	J^{128}
250	FA	11111010	140	A^{128}
21	15	00010101	141	F^{128}
63	3F	00111111	142	j^{16}
65	41	01000001	143	D^{16}
195	C3	11000011	144	l^{16}
94	5E	01011110	145	A^{16}
226	E2	11100010	146	H^{128}
61	3D	00111101	147	f^4
71	47	01000111	148	H^4
201	C9	11001001	149	m^{128}
64	40	01000000	150	E^{128}
192	C0	11000000	151	c^8
91	5B	01011011	152	g^8
237	ED	11101101	153	γ^8
44	2C	00101100	154	M^{32}
116	74	01110100	155	a^4
156	9C	10011100	156	f^{32}
191	BF	10111111	157	G^{32}
218	DA	11011010	158	S^{32}

dec	hex	binary	$\log_B$	name
117	75	01110101	159	k^{32}
159	9F	10011111	160	r^{32}
186	BA	10111010	161	C^{32}
213	D5	11010101	162	N^{32}
100	64	01100100	163	j^4
172	AC	10101100	164	H^{32}
239	EF	11101111	165	E^{32}
42	2A	00101010	166	M^8
126	7E	01111110	167	S^8
130	82	10000010	168	N^8
157	9D	10011101	169	M^2
188	BC	10111100	170	Ω^2
223	DF	11011111	171	n^4
122	7A	01111010	172	m^4
142	8E	10001110	173	h^{16}
137	89	10001001	174	n^{16}
128	80	10000000	175	R^{16}
155	9B	10011011	176	s^{16}
182	B6	10110110	177	F^{16}
193	C1	11000001	178	m^{16}
88	58	01011000	179	G^4
232	E8	11101000	180	E^4
35	23	00100011	181	h^2
101	65	01100101	182	h^{64}
175	AF	10101111	183	L^8
234	EA	11101010	184	J^8
37	25	00100101	185	a^{64}
111	6F	01101111	186	n^{64}
177	B1	10110001	187	β^4
200	C8	11001000	188	c^{64}
67	43	01000011	189	L^{64}
197	C5	11000101	190	R^{64}

dec	hex	binary	$\log_B$	name
84	54	01010100	191	b^{64}
252	FC	11111100	192	K^{64}
31	1F	00011111	193	d^{64}
33	21	00100001	194	s^{64}
99	63	01100011	195	T^{64}
165	A5	10100101	196	g^{64}
244	F4	11110100	197	J^{64}
7	07	00000111	198	F^{64}
9	09	00001001	199	D^8
27	1B	00011011	200	A^8
45	2D	00101101	201	f^2
119	77	01110111	202	m^{64}
153	99	10011001	203	c^4
176	B0	10110000	204	γ^4
203	CB	11001011	205	a^2
70	46	01000110	206	G^{16}
202	CA	11001010	207	k^{16}
69	45	01000101	208	C^{16}
207	CF	11001111	209	j^2
74	4A	01001010	210	E^{16}
222	DE	11011110	211	S^4
121	79	01111001	212	M
139	8B	10001011	213	n^2
134	86	10000110	214	h^8
145	91	10010001	215	R^8
168	A8	10101000	216	F^8
227	E3	11100011	217	G^2
62	3E	00111110	218	h
66	42	01000010	219	L^4
198	C6	11000110	220	a^{32}
81	51	01010001	221	β^2
243	F3	11110011	222	L^{32}

dec	hex	binary	$\log_B$	name
14	0E	00001110	223	b^{32}
18	12	00010010	224	d^{32}
54	36	00110110	225	T^{32}
90	5A	01011010	226	J^{32}
238	EE	11101110	227	D^4
41	29	00101001	228	f
123	7B	01111011	229	c^2
141	8D	10001101	230	a
140	8C	10001100	231	k^8
143	8F	10001111	232	j
138	8A	10001010	233	S^2
133	85	10000101	234	n
148	94	10010100	235	R^4
167	A7	10100111	236	G
242	F2	11110010	237	L^2
13	0D	00001101	238	β
23	17	00010111	239	b^{16}
57	39	00111001	240	T^{16}
75	4B	01001011	241	D^2
221	DD	11011101	242	c
124	7C	01111100	243	k^4
132	84	10000100	244	S
151	97	10010111	245	R^2
162	A2	10100010	246	L
253	FD	11111101	247	b^8
28	1C	00011100	248	D
36	24	00100100	249	k^2
108	6C	01101100	250	R
180	B4	10110100	251	b^4
199	C7	11000111	252	k
82	52	01010010	253	b^2
246	F6	11110110	254	b

D.3 Polynomial Table

Each minimal polynomial over $GF(2)$ is listed as a bit string of coefficients, e.g., 100011011 means $x^8 + x^4 + x^3 + x + 1 = q(x)$. Reversing the bit string corresponds to inverting the roots; the ordering is in such pairs. The conjugate roots are given in terms of $\log_B$; the first listed is given the name shown. The "order" is in the multiplicative subgroup, e.g., $\gamma^5 = 1$.

name	polynomial	order	$\log_B$ of conjugates							
0	10	1	$-\infty$							
1	11	1	0							
Ω	111	3	85	170						
α	10011	15	17	34	68	136				
β	11001	15	238	221	187	119				
γ	11111	5	51	102	204	153				
A	100011011	51	25	50	100	200	145	35	70	140
a	110110001	51	230	205	155	55	110	220	185	115
B	100011101	255	1	2	4	8	16	32	64	128
b	101110001	255	254	253	251	247	239	223	191	127
C	100101011	255	13	26	52	104	208	161	67	134
c	110101001	255	242	229	203	151	47	94	188	121
D	100101101	255	248	241	227	199	143	31	62	124
d	101101001	255	7	14	28	56	112	224	193	131
E	100111001	17	45	90	180	105	210	165	75	150
F	100111111	85	27	54	108	216	177	99	198	141
f	111111001	85	228	201	147	39	78	156	57	114
G	101001101	255	236	217	179	103	206	157	59	118
g	101100101	255	19	38	76	152	49	98	196	137
H	101011111	255	37	74	148	41	82	164	73	146
h	111110101	255	218	181	107	214	173	91	182	109
J	101100011	255	23	46	92	184	113	226	197	139
j	110001101	255	232	209	163	71	142	29	58	116
K	101110111	85	3	6	12	24	48	96	192	129
k	111011101	85	252	249	243	231	207	159	63	126
L	101111011	85	246	237	219	183	111	222	189	123
l	110111101	85	9	18	36	72	144	33	66	132
M	110000111	255	212	169	83	166	77	154	53	106
m	111000011	255	43	86	172	89	178	101	202	149
N	110001011	85	21	42	84	168	81	162	69	138
n	110100011	85	234	213	171	87	174	93	186	117
R	110011111	51	250	245	235	215	175	95	190	125
r	111110011	51	5	10	20	40	80	160	65	130
S	111001111	255	244	233	211	167	79	158	61	122
s	111100111	255	11	22	44	88	176	97	194	133
T	111010111	17	15	30	60	120	240	225	195	135

E All Possible Bases

The following table shows all 432 possible combinations of bases for $GF(2^8)$, $GF(2^4)$, and $GF(2^2)$ for which the trace is unity ($\tau = T = 1$). Each subfield basis is given as an ordered pair; if the second entry is 1 then it is a polynomial basis, otherwise a normal basis. The $GF(2^8)$ basis uses roots of $r(y) = y^2 + y + \nu$, the $GF(2^4)$ basis uses roots of $s(z) = z^2 + z + N$, where ν and N are the respective norms, and the $GF(2^2)$ basis uses roots of $t(w) = w^2 + w + 1$.

The basis and norm entries use the naming convention summarized in Table D.3. Explicitly, in terms of the standard AES basis: in subfield $GF(2^2)$, $\Omega = 189 = 0\text{xBD}$; in subfield $GF(2^4)$, $\alpha = 225 = 0\text{xE1}$, $\beta = 13 = 0\text{x0D}$, and $\gamma = 12 = 0\text{x0C}$; in the main field, $d = 255 = 0\text{xFF}$ and $L = 162 = 0\text{xA2}$.

The coefficients C and D of ν with respect to the $GF(2^4)$ basis are given in terms of N, as is the root w, as on page 18.

Under "XOR Gates," the first column shows the number of XOR gates for the inverter; each also includes 36 AND's. For bases 1–144, this number includes all of the low-level optimizations given in Section 4.4; bases 145 and beyond use a polynomial basis for $GF(2^8)$, and for those cases the inverter number is an estimate (except for 8 cases where these optimizations were explicitly included: 159, 177, 191, 209, 234, 252, 260, and 278). The last three columns show the XOR's for a complete S-box, an inverse S-box, and a merged combination of both with a shared inverter (excluding multiplexors); each would also have 36 AND's and possibly a few NOT's (for the affine transformation). A superscript $^{\circ}$ in the last column means the 16×8 basis change matrices were fully optimized by the tree-search algorithm; otherwise they were factored by the greedy algorithm. (*All* of the 8×8 matrices were fully optimized.)

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w =$	inv.	S-box	S-box^{-1}	Both
1	$[d^{16}, d]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β^8	Ω	N^2	1	N	66	99	94	122
2	$[d^{16}, d]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β^8	Ω	N^2	1	N	66	94	93	119
3	$[d^{16}, d]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β^8	Ω	N^2	1	N^2	66	94	92	116
4	$[d^{16}, d]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β^8	Ω^2	0	N^2	N^2	66	90	91	104[o]
5	$[d^{16}, d]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β^8	Ω^2	0	N^2	N^2	66	92	92	109[o]
6	$[d^{16}, d]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β^8	Ω^2	0	N^2	N	66	92	94	112[o]
7	$[d^{16}, d]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β^8	Ω	N	N^2	N	67	97	94	119
8	$[d^{16}, d]$	$[\alpha, 1]$	$[\Omega, 1]$	β^8	Ω	N	N^2	N	68	95	95	117
9	$[d^{16}, d]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β^8	Ω	N	N^2	N^2	68	98	94	119
10	$[d^{16}, d]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β^8	Ω	N	1	N	67	95	94	117
11	$[d^{16}, d]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β^8	Ω	N	1	N	67	93	94	115
12	$[d^{16}, d]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β^8	Ω	N	1	N^2	67	94	94	118
13	$[d^{16}, d]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β^8	Ω^2	N^2	0	N^2	67	91	92	114
14	$[d^{16}, d]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β^8	Ω^2	N^2	0	N^2	67	93	93	114
15	$[d^{16}, d]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β^8	Ω^2	N^2	0	N	67	94	95	118
16	$[d^{16}, d]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β^8	Ω^2	N^2	N^2	N^2	67	91	93	111[o]
17	$[d^{16}, d]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β^8	Ω^2	N^2	N^2	N^2	67	92	94	112[o]
18	$[d^{16}, d]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β^8	Ω^2	N^2	N^2	N	67	93	94	117
19	$[d^{32}, d^2]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β	Ω	N^2	0	N	66	96	98	124
20	$[d^{32}, d^2]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β	Ω	N^2	0	N	66	97	99	118
21	$[d^{32}, d^2]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β	Ω	N^2	0	N^2	66	95	97	116
22	$[d^{32}, d^2]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β	Ω^2	N^2	1	N^2	66	96	97	116
23	$[d^{32}, d^2]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β	Ω^2	N^2	1	N^2	66	94	92	107[o]
24	$[d^{32}, d^2]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β	Ω^2	N^2	1	N	66	94	94	117
25	$[d^{32}, d^2]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β	Ω	N^2	N^2	N	67	97	98	121
26	$[d^{32}, d^2]$	$[\alpha, 1]$	$[\Omega, 1]$	β	Ω	N^2	N^2	N	67	96	97	122
27	$[d^{32}, d^2]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β	Ω	N^2	N^2	N^2	67	97	95	121
28	$[d^{32}, d^2]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β	Ω	N^2	0	N	67	97	98	122
29	$[d^{32}, d^2]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β	Ω	N^2	0	N	67	96	93	117
30	$[d^{32}, d^2]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β	Ω	N^2	0	N^2	67	96	97	118
31	$[d^{32}, d^2]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β	Ω^2	N	N^2	N^2	67	96	97	122
32	$[d^{32}, d^2]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β	Ω^2	N	N^2	N^2	68	98	94	115
33	$[d^{32}, d^2]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β	Ω^2	N	N^2	N	68	99	96	120
34	$[d^{32}, d^2]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β	Ω^2	N	1	N^2	67	94	97	112[o]
35	$[d^{32}, d^2]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β	Ω^2	N	1	N^2	67	93	93	110[o]
36	$[d^{32}, d^2]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β	Ω^2	N	1	N	67	92	94	118

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
37	$[d^{64}, d^4]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β^2	Ω	1	N^2	N	66	94	93	115
38	$[d^{64}, d^4]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β^2	Ω	1	N^2	N	66	94	93	117
39	$[d^{64}, d^4]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β^2	Ω	1	N^2	N^2	66	95	92	111°
40	$[d^{64}, d^4]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	66	96	92	117
41	$[d^{64}, d^4]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	66	94	96	119
42	$[d^{64}, d^4]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	66	96	97	119
43	$[d^{64}, d^4]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	1	N	67	92	90	111°
44	$[d^{64}, d^4]$	$[\alpha, 1]$	$[\Omega, 1]$	β^2	Ω	N	1	N	67	91	92	107°
45	$[d^{64}, d^4]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	1	N^2	67	91	91	109°
46	$[d^{64}, d^4]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	N^2	N	67	96	94	120
47	$[d^{64}, d^4]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β^2	Ω	N	N^2	N	68	95	95	116
48	$[d^{64}, d^4]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	N^2	N^2	68	97	97	119
49	$[d^{64}, d^4]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	N^2	N^2	67	94	95	119
50	$[d^{64}, d^4]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	N^2	N^2	67	93	95	109°
51	$[d^{64}, d^4]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	N^2	N	67	94	96	118
52	$[d^{64}, d^4]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	67	95	95	117
53	$[d^{64}, d^4]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	67	93	94	115
54	$[d^{64}, d^4]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	67	96	97	115
55	$[d^{128}, d^8]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β^4	Ω	0	N^2	N	66	94	97	122
56	$[d^{128}, d^8]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β^4	Ω	0	N^2	N	66	96	96	122
57	$[d^{128}, d^8]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β^4	Ω	0	N^2	N^2	66	95	97	118
58	$[d^{128}, d^8]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β^4	Ω^2	1	N^2	N^2	66	96	96	115
59	$[d^{128}, d^8]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β^4	Ω^2	1	N^2	N^2	66	96	95	119
60	$[d^{128}, d^8]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β^4	Ω^2	1	N^2	N	66	96	97	114
61	$[d^{128}, d^8]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β^4	Ω	N^2	0	N	67	96	98	119
62	$[d^{128}, d^8]$	$[\alpha, 1]$	$[\Omega, 1]$	β^4	Ω	N^2	0	N	67	94	96	116
63	$[d^{128}, d^8]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β^4	Ω	N^2	0	N^2	67	96	98	119
64	$[d^{128}, d^8]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β^4	Ω	N^2	N^2	N	67	96	98	120
65	$[d^{128}, d^8]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β^4	Ω	N^2	N^2	N	67	95	95	118
66	$[d^{128}, d^8]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β^4	Ω	N^2	N^2	N^2	67	96	97	120
67	$[d^{128}, d^8]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β^4	Ω^2	N	1	N^2	67	97	98	121
68	$[d^{128}, d^8]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β^4	Ω^2	N	1	N^2	67	94	93	119
69	$[d^{128}, d^8]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β^4	Ω^2	N	1	N	67	99	98	123
70	$[d^{128}, d^8]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β^4	Ω^2	N	N^2	N^2	67	98	98	122
71	$[d^{128}, d^8]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β^4	Ω^2	N	N^2	N^2	68	97	97	124
72	$[d^{128}, d^8]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β^4	Ω^2	N	N^2	N	68	100	99	119

°fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
73	$[L^{16}, L]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	γ^2	Ω	0	N	N	66	94	93	117
74	$[L^{16}, L]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	γ^2	Ω	0	N	N	66	96	95	120
75	$[L^{16}, L]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	γ^2	Ω	0	N	N^2	66	96	93	119
76	$[L^{16}, L]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	γ^2	Ω^2	N	1	N^2	66	92	93	115
77	$[L^{16}, L]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	γ^2	Ω^2	N	1	N^2	66	93	94	119
78	$[L^{16}, L]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	γ^2	Ω^2	N	1	N	66	93	95	117
79	$[L^{16}, L]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	γ^2	Ω	N	0	N	67	93	92	114
80	$[L^{16}, L]$	$[\alpha, 1]$	$[\Omega, 1]$	γ^2	Ω	N	0	N	67	92	93	116
81	$[L^{16}, L]$	$[\alpha, 1]$	$[\Omega^2, 1]$	γ^2	Ω	N	0	N^2	67	94	93	115
82	$[L^{16}, L]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	γ^2	Ω	N	N	N	67	94	96	116
83	$[L^{16}, L]$	$[\alpha^4, 1]$	$[\Omega, 1]$	γ^2	Ω	N	N	N	67	94	93	116
84	$[L^{16}, L]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	γ^2	Ω	N	N	N^2	67	93	94	115
85	$[L^{16}, L]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	γ^2	Ω^2	N^2	N	N^2	67	92	93	111[o]
86	$[L^{16}, L]$	$[\alpha^2, 1]$	$[\Omega, 1]$	γ^2	Ω^2	N^2	N	N^2	67	92	94	116
87	$[L^{16}, L]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	γ^2	Ω^2	N^2	N	N	67	93	95	117
88	$[L^{16}, L]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	γ^2	Ω^2	N^2	1	N^2	67	94	94	111
89	$[L^{16}, L]$	$[\alpha^8, 1]$	$[\Omega, 1]$	γ^2	Ω^2	N^2	1	N^2	68	92	92	115
90	$[L^{16}, L]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	γ^2	Ω^2	N^2	1	N	67	93	93	115
91	$[L^{32}, L^2]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	γ^4	Ω	1	N	N	66	96	95	119
92	$[L^{32}, L^2]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	γ^4	Ω	1	N	N	66	95	96	121
93	$[L^{32}, L^2]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	γ^4	Ω	1	N	N^2	66	97	97	118
94	$[L^{32}, L^2]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	γ^4	Ω^2	0	N	N^2	66	94	94	112[o]
95	$[L^{32}, L^2]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	γ^4	Ω^2	0	N	N^2	66	93	96	118
96	$[L^{32}, L^2]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	γ^4	Ω^2	0	N	N	66	93	95	119
97	$[L^{32}, L^2]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	γ^4	Ω	N^2	1	N	67	98	99	125
98	$[L^{32}, L^2]$	$[\alpha, 1]$	$[\Omega, 1]$	γ^4	Ω	N^2	1	N	67	97	98	124
99	$[L^{32}, L^2]$	$[\alpha, 1]$	$[\Omega^2, 1]$	γ^4	Ω	N^2	1	N^2	68	98	99	122
100	$[L^{32}, L^2]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	γ^4	Ω	N^2	N	N	67	92	93	117
101	$[L^{32}, L^2]$	$[\alpha^4, 1]$	$[\Omega, 1]$	γ^4	Ω	N^2	N	N	67	96	96	120
102	$[L^{32}, L^2]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	γ^4	Ω	N^2	N	N^2	67	97	97	122
103	$[L^{32}, L^2]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	γ^4	Ω^2	N	0	N^2	67	95	97	121
104	$[L^{32}, L^2]$	$[\alpha^2, 1]$	$[\Omega, 1]$	γ^4	Ω^2	N	0	N^2	67	94	96	118
105	$[L^{32}, L^2]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	γ^4	Ω^2	N	0	N	67	95	96	119
106	$[L^{32}, L^2]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	γ^4	Ω^2	N	N	N^2	67	95	98	121
107	$[L^{32}, L^2]$	$[\alpha^8, 1]$	$[\Omega, 1]$	γ^4	Ω^2	N	N	N^2	67	96	95	116
108	$[L^{32}, L^2]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	γ^4	Ω^2	N	N	N	67	97	97	118

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w =$	inv.	S-box	S-box^{-1}	Both
109	$[L^{64}, L^4]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	γ^8	Ω	N	0	N	66	91	93	114
110	$[L^{64}, L^4]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	γ^8	Ω	N	0	N	66	91	92	108^o
111	$[L^{64}, L^4]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	γ^8	Ω	N	0	N^2	66	91	93	108^o
112	$[L^{64}, L^4]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	γ^8	Ω^2	1	N	N^2	66	93	90	114
113	$[L^{64}, L^4]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	γ^8	Ω^2	1	N	N^2	66	94	90	109^o
114	$[L^{64}, L^4]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	γ^8	Ω^2	1	N	N	66	91	90	108^o
115	$[L^{64}, L^4]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	γ^8	Ω	N	N	N	67	94	96	115
116	$[L^{64}, L^4]$	$[\alpha, 1]$	$[\Omega, 1]$	γ^8	Ω	N	N	N	67	93	94	115
117	$[L^{64}, L^4]$	$[\alpha, 1]$	$[\Omega^2, 1]$	γ^8	Ω	N	N	N^2	67	92	94	109^o
118	$[L^{64}, L^4]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	γ^8	Ω	N	0	N	67	91	94	112^o
119	$[L^{64}, L^4]$	$[\alpha^4, 1]$	$[\Omega, 1]$	γ^8	Ω	N	0	N	67	93	93	118
120	$[L^{64}, L^4]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	γ^8	Ω	N	0	N^2	67	92	93	116
121	$[L^{64}, L^4]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	γ^8	Ω^2	N^2	1	N^2	67	96	93	119
122	$[L^{64}, L^4]$	$[\alpha^2, 1]$	$[\Omega, 1]$	γ^8	Ω^2	N^2	1	N^2	68	95	93	114
123	$[L^{64}, L^4]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	γ^8	Ω^2	N^2	1	N	67	95	93	116
124	$[L^{64}, L^4]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	γ^8	Ω^2	N^2	N	N^2	67	95	92	115
125	$[L^{64}, L^4]$	$[\alpha^8, 1]$	$[\Omega, 1]$	γ^8	Ω^2	N^2	N	N^2	67	93	90	110^o
126	$[L^{64}, L^4]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	γ^8	Ω^2	N^2	N	N	67	94	92	116
127	$[L^{128}, L^8]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	γ	Ω	N	1	N	66	96	97	119
128	$[L^{128}, L^8]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	γ	Ω	N	1	N	66	97	97	119
129	$[L^{128}, L^8]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	γ	Ω	N	1	N^2	66	98	98	121
130	$[L^{128}, L^8]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	γ	Ω^2	N	0	N^2	66	92	91	115
131	$[L^{128}, L^8]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	γ	Ω^2	N	0	N^2	66	94	95	120
132	$[L^{128}, L^8]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	γ	Ω^2	N	0	N	66	92	95	116
133	$[L^{128}, L^8]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	γ	Ω	N^2	N	N	67	98	97	120
134	$[L^{128}, L^8]$	$[\alpha, 1]$	$[\Omega, 1]$	γ	Ω	N^2	N	N	67	98	99	119
135	$[L^{128}, L^8]$	$[\alpha, 1]$	$[\Omega^2, 1]$	γ	Ω	N^2	N	N^2	67	97	100	121
136	$[L^{128}, L^8]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	γ	Ω	N^2	1	N	67	99	99	123
137	$[L^{128}, L^8]$	$[\alpha^4, 1]$	$[\Omega, 1]$	γ	Ω	N^2	1	N	67	96	99	122
138	$[L^{128}, L^8]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	γ	Ω	N^2	1	N^2	68	98	99	126
139	$[L^{128}, L^8]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	γ	Ω^2	N	N	N^2	67	96	99	119
140	$[L^{128}, L^8]$	$[\alpha^2, 1]$	$[\Omega, 1]$	γ	Ω^2	N	N	N^2	67	96	98	120
141	$[L^{128}, L^8]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	γ	Ω^2	N	N	N	67	96	99	122
142	$[L^{128}, L^8]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	γ	Ω^2	N	0	N^2	67	95	94	116
143	$[L^{128}, L^8]$	$[\alpha^8, 1]$	$[\Omega, 1]$	γ	Ω^2	N	0	N^2	67	95	97	119
144	$[L^{128}, L^8]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	γ	Ω^2	N	0	N	67	95	97	117

^{o}fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
145	$[d,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β^8	Ω	N^2	1	N	72	100	98	122
146	$[d,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β^8	Ω	N^2	1	N	72	99	98	122
147	$[d,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β^8	Ω	N^2	1	N^2	72	97	97	121
148	$[d,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β^8	Ω^2	0	N^2	N^2	72	96	97	115
149	$[d,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β^8	Ω^2	0	N^2	N^2	72	97	99	112[o]
150	$[d,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β^8	Ω^2	0	N^2	N	72	98	100	119
151	$[d,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β^8	Ω	N	N^2	N	73	101	98	123
152	$[d,1]$	$[\alpha,1]$	$[\Omega,1]$	β^8	Ω	N	N^2	N	74	100	99	117
153	$[d,1]$	$[\alpha,1]$	$[\Omega^2,1]$	β^8	Ω	N	N^2	N^2	74	99	98	120
154	$[d,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β^8	Ω	N	1	N	73	97	97	123
155	$[d,1]$	$[\alpha^4,1]$	$[\Omega,1]$	β^8	Ω	N	1	N	73	97	96	120
156	$[d,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β^8	Ω	N	1	N^2	73	99	100	124
157	$[d,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β^8	Ω^2	N^2	0	N^2	73	98	99	116
158	$[d,1]$	$[\alpha^2,1]$	$[\Omega,1]$	β^8	Ω^2	N^2	0	N^2	73	97	98	118
159	$[d,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β^8	Ω^2	N^2	0	N	73	98	101	121
160	$[d,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β^8	Ω^2	N^2	N^2	N^2	73	97	95	117
161	$[d,1]$	$[\alpha^8,1]$	$[\Omega,1]$	β^8	Ω^2	N^2	N^2	N^2	73	96	96	116
162	$[d,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β^8	Ω^2	N^2	N^2	N	73	98	99	122
163	$[d^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β^8	Ω	N^2	1	N	72	102	101	127
164	$[d^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β^8	Ω	N^2	1	N	72	101	99	124
165	$[d^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β^8	Ω	N^2	1	N^2	72	102	100	128
166	$[d^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β^8	Ω^2	0	N^2	N^2	72	97	98	117
167	$[d^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β^8	Ω^2	0	N^2	N^2	72	98	99	119
168	$[d^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β^8	Ω^2	0	N^2	N	72	100	98	120
169	$[d^{16},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β^8	Ω	N	N^2	N	73	99	98	122
170	$[d^{16},1]$	$[\alpha,1]$	$[\Omega,1]$	β^8	Ω	N	N^2	N	74	103	102	125
171	$[d^{16},1]$	$[\alpha,1]$	$[\Omega^2,1]$	β^8	Ω	N	N^2	N^2	74	103	102	128
172	$[d^{16},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β^8	Ω	N	1	N	73	101	102	125
173	$[d^{16},1]$	$[\alpha^4,1]$	$[\Omega,1]$	β^8	Ω	N	1	N	73	99	101	124
174	$[d^{16},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β^8	Ω	N	1	N^2	73	101	101	125
175	$[d^{16},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β^8	Ω^2	N^2	0	N^2	73	98	96	121
176	$[d^{16},1]$	$[\alpha^2,1]$	$[\Omega,1]$	β^8	Ω^2	N^2	0	N^2	73	99	100	120
177	$[d^{16},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β^8	Ω^2	N^2	0	N	73	100	101	120
178	$[d^{16},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β^8	Ω^2	N^2	N^2	N^2	73	98	99	121
179	$[d^{16},1]$	$[\alpha^8,1]$	$[\Omega,1]$	β^8	Ω^2	N^2	N^2	N^2	73	99	101	121
180	$[d^{16},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β^8	Ω^2	N^2	N^2	N	73	100	99	122

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
181	$[d^2,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β	Ω	N^2	0	N	72	100	101	124
182	$[d^2,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β	Ω	N^2	0	N	72	102	101	124
183	$[d^2,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β	Ω	N^2	0	N^2	72	99	100	121
184	$[d^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β	Ω^2	N^2	1	N^2	72	97	99	120
185	$[d^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β	Ω^2	N^2	1	N^2	72	98	98	116
186	$[d^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β	Ω^2	N^2	1	N	72	96	99	119
187	$[d^2,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β	Ω	N^2	N^2	N	73	102	103	125
188	$[d^2,1]$	$[\alpha,1]$	$[\Omega,1]$	β	Ω	N^2	N^2	N	73	101	100	122
189	$[d^2,1]$	$[\alpha,1]$	$[\Omega^2,1]$	β	Ω	N^2	N^2	N^2	73	100	102	125
190	$[d^2,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β	Ω	N^2	0	N	73	100	104	124
191	$[d^2,1]$	$[\alpha^4,1]$	$[\Omega,1]$	β	Ω	N^2	0	N	73	101	101	126
192	$[d^2,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β	Ω	N^2	0	N^2	73	102	103	126
193	$[d^2,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β	Ω^2	N	N^2	N^2	73	101	102	123
194	$[d^2,1]$	$[\alpha^2,1]$	$[\Omega,1]$	β	Ω^2	N	N^2	N^2	74	101	99	120
195	$[d^2,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β	Ω^2	N	N^2	N	74	103	103	127
196	$[d^2,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β	Ω^2	N	1	N^2	73	98	97	121
197	$[d^2,1]$	$[\alpha^8,1]$	$[\Omega,1]$	β	Ω^2	N	1	N^2	73	97	97	116
198	$[d^2,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β	Ω^2	N	1	N	73	101	99	124
199	$[d^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β	Ω	N^2	0	N	72	102	102	130
200	$[d^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β	Ω	N^2	0	N	72	103	100	126
201	$[d^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β	Ω	N^2	0	N^2	72	102	100	125
202	$[d^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β	Ω^2	N^2	1	N^2	72	99	98	121
203	$[d^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β	Ω^2	N^2	1	N^2	72	96	97	116
204	$[d^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β	Ω^2	N^2	1	N	72	99	99	123
205	$[d^{32},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β	Ω	N^2	N^2	N	73	100	99	125
206	$[d^{32},1]$	$[\alpha,1]$	$[\Omega,1]$	β	Ω	N^2	N^2	N	73	102	99	121
207	$[d^{32},1]$	$[\alpha,1]$	$[\Omega^2,1]$	β	Ω	N^2	N^2	N^2	73	101	102	125
208	$[d^{32},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β	Ω	N^2	0	N	73	100	103	128
209	$[d^{32},1]$	$[\alpha^4,1]$	$[\Omega,1]$	β	Ω	N^2	0	N	73	100	99	120
210	$[d^{32},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β	Ω	N^2	0	N^2	73	102	101	122
211	$[d^{32},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β	Ω^2	N	N^2	N^2	73	101	102	127
212	$[d^{32},1]$	$[\alpha^2,1]$	$[\Omega,1]$	β	Ω^2	N	N^2	N^2	74	101	99	125
213	$[d^{32},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β	Ω^2	N	N^2	N	74	101	102	125
214	$[d^{32},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β	Ω^2	N	1	N^2	73	98	100	124
215	$[d^{32},1]$	$[\alpha^8,1]$	$[\Omega,1]$	β	Ω^2	N	1	N^2	73	99	97	121
216	$[d^{32},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β	Ω^2	N	1	N	73	98	99	125

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
217	$[d^4, 1]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β^2	Ω	1	N^2	N	72	100	100	123
218	$[d^4, 1]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β^2	Ω	1	N^2	N	72	101	99	121
219	$[d^4, 1]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β^2	Ω	1	N^2	N^2	72	100	98	120
220	$[d^4, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	72	98	99	123
221	$[d^4, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	72	100	101	120
222	$[d^4, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	72	98	101	122
223	$[d^4, 1]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	1	N	73	98	99	122
224	$[d^4, 1]$	$[\alpha, 1]$	$[\Omega, 1]$	β^2	Ω	N	1	N	73	97	98	114[o]
225	$[d^4, 1]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	1	N^2	73	100	101	118
226	$[d^4, 1]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	N^2	N	73	99	101	122
227	$[d^4, 1]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β^2	Ω	N	N^2	N	74	102	103	124
228	$[d^4, 1]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	N^2	N^2	74	103	101	123
229	$[d^4, 1]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	N^2	N^2	73	99	101	126
230	$[d^4, 1]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	N^2	N^2	73	98	101	120
231	$[d^4, 1]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	N^2	N	73	98	100	124
232	$[d^4, 1]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	73	100	98	120
233	$[d^4, 1]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	73	97	98	122
234	$[d^4, 1]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	73	102	100	124
235	$[d^{64}, 1]$	$[\alpha^4, \alpha]$	$[\Omega^2, \Omega]$	β^2	Ω	1	N^2	N	72	100	99	118
236	$[d^{64}, 1]$	$[\alpha^4, \alpha]$	$[\Omega, 1]$	β^2	Ω	1	N^2	N	72	99	97	118
237	$[d^{64}, 1]$	$[\alpha^4, \alpha]$	$[\Omega^2, 1]$	β^2	Ω	1	N^2	N^2	72	98	98	123
238	$[d^{64}, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	72	99	98	122
239	$[d^{64}, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	72	96	99	117
240	$[d^{64}, 1]$	$[\alpha^8, \alpha^2]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	72	98	99	123
241	$[d^{64}, 1]$	$[\alpha, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	1	N	73	96	96	116
242	$[d^{64}, 1]$	$[\alpha, 1]$	$[\Omega, 1]$	β^2	Ω	N	1	N	73	98	96	116
243	$[d^{64}, 1]$	$[\alpha, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	1	N^2	73	97	98	119
244	$[d^{64}, 1]$	$[\alpha^4, 1]$	$[\Omega^2, \Omega]$	β^2	Ω	N	N^2	N	73	101	100	120
245	$[d^{64}, 1]$	$[\alpha^4, 1]$	$[\Omega, 1]$	β^2	Ω	N	N^2	N	74	100	102	121
246	$[d^{64}, 1]$	$[\alpha^4, 1]$	$[\Omega^2, 1]$	β^2	Ω	N	N^2	N^2	74	101	102	119
247	$[d^{64}, 1]$	$[\alpha^2, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	N^2	N^2	73	97	99	124
248	$[d^{64}, 1]$	$[\alpha^2, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	N^2	N^2	73	97	97	116
249	$[d^{64}, 1]$	$[\alpha^2, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	N^2	N	73	98	100	121
250	$[d^{64}, 1]$	$[\alpha^8, 1]$	$[\Omega^2, \Omega]$	β^2	Ω^2	N^2	0	N^2	73	98	98	120
251	$[d^{64}, 1]$	$[\alpha^8, 1]$	$[\Omega, 1]$	β^2	Ω^2	N^2	0	N^2	73	97	97	116
252	$[d^{64}, 1]$	$[\alpha^8, 1]$	$[\Omega^2, 1]$	β^2	Ω^2	N^2	0	N	73	99	99	115[o]

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
253	$[d^8,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β^4	Ω	0	N^2	N	72	101	103	125
254	$[d^8,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β^4	Ω	0	N^2	N	72	101	102	126
255	$[d^8,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β^4	Ω	0	N^2	N^2	72	100	102	126
256	$[d^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β^4	Ω^2	1	N^2	N^2	72	100	100	120
257	$[d^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β^4	Ω^2	1	N^2	N^2	72	100	100	119
258	$[d^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β^4	Ω^2	1	N^2	N	72	100	99	117
259	$[d^8,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β^4	Ω	N^2	0	N	73	100	100	122
260	$[d^8,1]$	$[\alpha,1]$	$[\Omega,1]$	β^4	Ω	N^2	0	N	73	101	101	123
261	$[d^8,1]$	$[\alpha,1]$	$[\Omega^2,1]$	β^4	Ω	N^2	0	N^2	73	100	104	125
262	$[d^8,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β^4	Ω	N^2	N^2	N	73	100	102	126
263	$[d^8,1]$	$[\alpha^4,1]$	$[\Omega,1]$	β^4	Ω	N^2	N^2	N	73	99	101	128
264	$[d^8,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β^4	Ω	N^2	N^2	N^2	73	102	103	127
265	$[d^8,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β^4	Ω^2	N	1	N^2	73	103	101	124
266	$[d^8,1]$	$[\alpha^2,1]$	$[\Omega,1]$	β^4	Ω^2	N	1	N^2	73	100	98	119
267	$[d^8,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β^4	Ω^2	N	1	N	73	99	101	120
268	$[d^8,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β^4	Ω^2	N	N^2	N^2	73	101	101	120
269	$[d^8,1]$	$[\alpha^8,1]$	$[\Omega,1]$	β^4	Ω^2	N	N^2	N^2	74	101	104	128
270	$[d^8,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β^4	Ω^2	N	N^2	N	74	103	103	128
271	$[d^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	β^4	Ω	0	N^2	N	72	99	99	122
272	$[d^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	β^4	Ω	0	N^2	N	72	101	99	124
273	$[d^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	β^4	Ω	0	N^2	N^2	72	100	100	122
274	$[d^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	β^4	Ω^2	1	N^2	N^2	72	101	99	124
275	$[d^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	β^4	Ω^2	1	N^2	N^2	72	99	101	120
276	$[d^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	β^4	Ω^2	1	N^2	N	72	99	102	126
277	$[d^{128},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	β^4	Ω	N^2	0	N	73	101	102	125
278	$[d^{128},1]$	$[\alpha,1]$	$[\Omega,1]$	β^4	Ω	N^2	0	N	73	101	101	122
279	$[d^{128},1]$	$[\alpha,1]$	$[\Omega^2,1]$	β^4	Ω	N^2	0	N^2	73	102	102	125
280	$[d^{128},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	β^4	Ω	N^2	N^2	N	73	101	99	125
281	$[d^{128},1]$	$[\alpha^4,1]$	$[\Omega,1]$	β^4	Ω	N^2	N^2	N	73	102	100	122
282	$[d^{128},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	β^4	Ω	N^2	N^2	N^2	73	102	102	126
283	$[d^{128},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	β^4	Ω^2	N	1	N^2	73	101	100	125
284	$[d^{128},1]$	$[\alpha^2,1]$	$[\Omega,1]$	β^4	Ω^2	N	1	N^2	73	99	98	121
285	$[d^{128},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	β^4	Ω^2	N	1	N	73	102	102	126
286	$[d^{128},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	β^4	Ω^2	N	N^2	N^2	73	100	100	123
287	$[d^{128},1]$	$[\alpha^8,1]$	$[\Omega,1]$	β^4	Ω^2	N	N^2	N^2	74	103	101	125
288	$[d^{128},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	β^4	Ω^2	N	N^2	N	74	104	103	127

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
289	$[L,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^2	Ω	0	N	N	72	99	100	125
290	$[L,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^2	Ω	0	N	N	72	98	98	120
291	$[L,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^2	Ω	0	N	N^2	72	100	101	126
292	$[L,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N	1	N^2	72	98	98	119
293	$[L,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^2	Ω^2	N	1	N^2	72	96	96	116
294	$[L,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^2	Ω^2	N	1	N	72	97	99	121
295	$[L,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^2	Ω	N	0	N	73	98	97	119
296	$[L,1]$	$[\alpha,1]$	$[\Omega,1]$	γ^2	Ω	N	0	N	73	97	96	118
297	$[L,1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^2	Ω	N	0	N^2	73	98	100	117
298	$[L,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^2	Ω	N	N	N	73	98	98	123
299	$[L,1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^2	Ω	N	N	N	73	99	100	120
300	$[L,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^2	Ω	N	N	N^2	73	101	102	125
301	$[L,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N^2	N	N^2	73	98	99	122
302	$[L,1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^2	Ω^2	N^2	N	N^2	73	96	99	119
303	$[L,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^2	Ω^2	N^2	N	N	73	97	100	119
304	$[L,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N^2	1	N^2	73	99	95	119
305	$[L,1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^2	Ω^2	N^2	1	N^2	74	99	98	120
306	$[L,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^2	Ω^2	N^2	1	N	73	99	99	122
307	$[L^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^2	Ω	0	N	N	72	100	101	124
308	$[L^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^2	Ω	0	N	N	72	102	102	126
309	$[L^{16},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^2	Ω	0	N	N^2	72	100	100	124
310	$[L^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N	1	N^2	72	99	98	120
311	$[L^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^2	Ω^2	N	1	N^2	72	98	98	125
312	$[L^{16},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^2	Ω^2	N	1	N	72	97	99	122
313	$[L^{16},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^2	Ω	N	0	N	73	100	98	119
314	$[L^{16},1]$	$[\alpha,1]$	$[\Omega,1]$	γ^2	Ω	N	0	N	73	101	97	120
315	$[L^{16},1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^2	Ω	N	0	N^2	73	101	100	123
316	$[L^{16},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^2	Ω	N	N	N	73	100	100	119
317	$[L^{16},1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^2	Ω	N	N	N	73	100	100	121
318	$[L^{16},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^2	Ω	N	N	N^2	73	104	100	121
319	$[L^{16},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N^2	N	N^2	73	99	100	120
320	$[L^{16},1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^2	Ω^2	N^2	N	N^2	73	98	100	122
321	$[L^{16},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^2	Ω^2	N^2	N	N	73	100	99	121
322	$[L^{16},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^2	Ω^2	N^2	1	N^2	73	97	97	117
323	$[L^{16},1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^2	Ω^2	N^2	1	N^2	74	99	98	118
324	$[L^{16},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^2	Ω^2	N^2	1	N	73	101	100	124

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
325	$[L^2,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^4	Ω	1	N	N	72	99	99	124
326	$[L^2,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^4	Ω	1	N	N	72	98	99	122
327	$[L^2,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^4	Ω	1	N	N^2	72	99	99	127
328	$[L^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	0	N	N^2	72	97	99	116
329	$[L^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^4	Ω^2	0	N	N^2	72	98	100	124
330	$[L^2,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^4	Ω^2	0	N	N	72	98	99	118
331	$[L^2,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^4	Ω	N^2	1	N	73	101	102	123
332	$[L^2,1]$	$[\alpha,1]$	$[\Omega,1]$	γ^4	Ω	N^2	1	N	73	101	101	124
333	$[L^2,1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^4	Ω	N^2	1	N^2	74	101	104	124
334	$[L^2,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^4	Ω	N^2	N	N	73	99	99	120
335	$[L^2,1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^4	Ω	N^2	N	N	73	98	97	121
336	$[L^2,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^4	Ω	N^2	N	N^2	73	99	100	121
337	$[L^2,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	N	0	N^2	73	100	102	126
338	$[L^2,1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^4	Ω^2	N	0	N^2	73	99	102	125
339	$[L^2,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^4	Ω^2	N	0	N	73	100	103	124
340	$[L^2,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	N	N	N^2	73	99	98	118
341	$[L^2,1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^4	Ω^2	N	N	N^2	73	101	102	123
342	$[L^2,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^4	Ω^2	N	N	N	73	101	102	124
343	$[L^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^4	Ω	1	N	N	72	100	103	126
344	$[L^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^4	Ω	1	N	N	72	101	100	125
345	$[L^{32},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^4	Ω	1	N	N^2	72	102	103	127
346	$[L^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	0	N	N^2	72	101	99	123
347	$[L^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^4	Ω^2	0	N	N^2	72	96	96	117
348	$[L^{32},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^4	Ω^2	0	N	N	72	99	98	120
349	$[L^{32},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^4	Ω	N^2	1	N	73	102	103	130
350	$[L^{32},1]$	$[\alpha,1]$	$[\Omega,1]$	γ^4	Ω	N^2	1	N	73	102	101	125
351	$[L^{32},1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^4	Ω	N^2	1	N^2	74	103	104	129
352	$[L^{32},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^4	Ω	N^2	N	N	73	102	101	126
353	$[L^{32},1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^4	Ω	N^2	N	N	73	101	102	126
354	$[L^{32},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^4	Ω	N^2	N	N^2	73	103	102	128
355	$[L^{32},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	N	0	N^2	73	102	101	124
356	$[L^{32},1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^4	Ω^2	N	0	N^2	73	99	99	118
357	$[L^{32},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^4	Ω^2	N	0	N	73	99	102	125
358	$[L^{32},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^4	Ω^2	N	N	N^2	73	102	100	124
359	$[L^{32},1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^4	Ω^2	N	N	N^2	73	100	100	124
360	$[L^{32},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^4	Ω^2	N	N	N	73	100	102	122

[o]fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
361	$[L^4,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	0	N	72	95	97	119
362	$[L^4,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^8	Ω	N	0	N	72	96	97	117
363	$[L^4,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^8	Ω	N	0	N^2	72	97	98	117
364	$[L^4,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	1	N	N^2	72	97	96	118
365	$[L^4,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^8	Ω^2	1	N	N^2	72	96	94	115
366	$[L^4,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^8	Ω^2	1	N	N	72	95	93	116
367	$[L^4,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	N	N	73	96	97	114^o
368	$[L^4,1]$	$[\alpha,1]$	$[\Omega,1]$	γ^8	Ω	N	N	N	73	98	96	118
369	$[L^4,1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^8	Ω	N	N	N^2	73	98	99	119
370	$[L^4,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	0	N	73	97	98	119
371	$[L^4,1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^8	Ω	N	0	N	73	96	97	118
372	$[L^4,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^8	Ω	N	0	N^2	73	99	99	117
373	$[L^4,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	N^2	1	N^2	73	101	98	122
374	$[L^4,1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^8	Ω^2	N^2	1	N^2	74	98	99	120
375	$[L^4,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^8	Ω^2	N^2	1	N	73	98	98	119
376	$[L^4,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	N^2	N	N^2	73	98	95	110^o
377	$[L^4,1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^8	Ω^2	N^2	N	N^2	73	97	95	116
378	$[L^4,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^8	Ω^2	N^2	N	N	73	99	98	120
379	$[L^{64},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	0	N	72	99	102	127
380	$[L^{64},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ^8	Ω	N	0	N	72	102	101	128
381	$[L^{64},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ^8	Ω	N	0	N^2	72	99	101	128
382	$[L^{64},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	1	N	N^2	72	99	98	119
383	$[L^{64},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ^8	Ω^2	1	N	N^2	72	99	100	120
384	$[L^{64},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ^8	Ω^2	1	N	N	72	99	98	122
385	$[L^{64},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	N	N	73	100	100	121
386	$[L^{64},1]$	$[\alpha,1]$	$[\Omega,1]$	γ^8	Ω	N	N	N	73	102	100	124
387	$[L^{64},1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ^8	Ω	N	N	N^2	73	101	103	124
388	$[L^{64},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ^8	Ω	N	0	N	73	98	101	123
389	$[L^{64},1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ^8	Ω	N	0	N	73	100	99	124
390	$[L^{64},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ^8	Ω	N	0	N^2	73	102	101	121
391	$[L^{64},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	N^2	1	N^2	73	99	100	123
392	$[L^{64},1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ^8	Ω^2	N^2	1	N^2	74	102	100	122
393	$[L^{64},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ^8	Ω^2	N^2	1	N	73	101	99	124
394	$[L^{64},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ^8	Ω^2	N^2	N	N^2	73	101	99	120
395	$[L^{64},1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ^8	Ω^2	N^2	N	N^2	73	99	99	121
396	$[L^{64},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ^8	Ω^2	N^2	N	N	73	102	98	121

^{o}fully optimized results

Case	Bases			Norms		Coefficients			XOR Gates			
#	$GF(2^8)$	$GF(2^4)$	$GF(2^2)$	ν	N	C	D	$w=$	inv.	S-box	S-box^{-1}	Both
397	$[L^8,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ	Ω	N	1	N	72	102	103	125
398	$[L^8,1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ	Ω	N	1	N	72	101	101	123
399	$[L^8,1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ	Ω	N	1	N^2	72	103	102	123
400	$[L^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	0	N^2	72	99	101	120
401	$[L^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ	Ω^2	N	0	N^2	72	99	100	119
402	$[L^8,1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ	Ω^2	N	0	N	72	99	98	121
403	$[L^8,1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ	Ω	N^2	N	N	73	101	103	123
404	$[L^8,1]$	$[\alpha,1]$	$[\Omega,1]$	γ	Ω	N^2	N	N	73	101	103	120
405	$[L^8,1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ	Ω	N^2	N	N^2	73	102	106	123
406	$[L^8,1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ	Ω	N^2	1	N	73	102	104	129
407	$[L^8,1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ	Ω	N^2	1	N	73	101	103	125
408	$[L^8,1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ	Ω	N^2	1	N^2	74	103	104	128
409	$[L^8,1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	N	N^2	73	102	102	125
410	$[L^8,1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ	Ω^2	N	N	N^2	73	99	103	121
411	$[L^8,1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ	Ω^2	N	N	N	73	101	102	125
412	$[L^8,1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	0	N^2	73	101	100	121
413	$[L^8,1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ	Ω^2	N	0	N^2	73	101	101	119
414	$[L^8,1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ	Ω^2	N	0	N	73	100	101	123
415	$[L^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,\Omega]$	γ	Ω	N	1	N	72	100	101	127
416	$[L^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega,1]$	γ	Ω	N	1	N	72	101	100	120
417	$[L^{128},1]$	$[\alpha^4,\alpha]$	$[\Omega^2,1]$	γ	Ω	N	1	N^2	72	102	103	128
418	$[L^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	0	N^2	72	98	99	118
419	$[L^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega,1]$	γ	Ω^2	N	0	N^2	72	97	101	122
420	$[L^{128},1]$	$[\alpha^8,\alpha^2]$	$[\Omega^2,1]$	γ	Ω^2	N	0	N	72	100	100	123
421	$[L^{128},1]$	$[\alpha,1]$	$[\Omega^2,\Omega]$	γ	Ω	N^2	N	N	73	101	104	122
422	$[L^{128},1]$	$[\alpha,1]$	$[\Omega,1]$	γ	Ω	N^2	N	N	73	101	100	122
423	$[L^{128},1]$	$[\alpha,1]$	$[\Omega^2,1]$	γ	Ω	N^2	N	N^2	73	102	103	127
424	$[L^{128},1]$	$[\alpha^4,1]$	$[\Omega^2,\Omega]$	γ	Ω	N^2	1	N	73	103	103	126
425	$[L^{128},1]$	$[\alpha^4,1]$	$[\Omega,1]$	γ	Ω	N^2	1	N	73	100	101	125
426	$[L^{128},1]$	$[\alpha^4,1]$	$[\Omega^2,1]$	γ	Ω	N^2	1	N^2	74	106	105	131
427	$[L^{128},1]$	$[\alpha^2,1]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	N	N^2	73	101	102	124
428	$[L^{128},1]$	$[\alpha^2,1]$	$[\Omega,1]$	γ	Ω^2	N	N	N^2	73	99	101	122
429	$[L^{128},1]$	$[\alpha^2,1]$	$[\Omega^2,1]$	γ	Ω^2	N	N	N	73	100	104	128
430	$[L^{128},1]$	$[\alpha^8,1]$	$[\Omega^2,\Omega]$	γ	Ω^2	N	0	N^2	73	98	100	118
431	$[L^{128},1]$	$[\alpha^8,1]$	$[\Omega,1]$	γ	Ω^2	N	0	N^2	73	97	99	120
432	$[L^{128},1]$	$[\alpha^8,1]$	$[\Omega^2,1]$	γ	Ω^2	N	0	N	73	101	101	121

[o]fully optimized results

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943-5100	2
3.	Alan Hunsberger National Security Agency 9800 Savage Road, Ste. 6538 Fort Meade, MD 20755-6538	2
4.	Douglas Fouts, Code EC/Fs Naval Postgraduate School Monterey, CA 93943-5207	1
5.	David Canright, Code MA/Ca Naval Postgraduate School Monterey, CA 93943-5216	10