Constructing Composite Field Representations for Efficient Conversion

Berk Sunar, *Member*, *IEEE*, Erkay Savas, *Member*, *IEEE*, and Çetin K. Koç, *Senior Member*, *IEEE*

Abstract—This paper describes a method of construction of a composite field representation from a given binary field representation. We derive the conversion (change of basis) matrix. The special case of when the degree of the ground field is relatively prime to the extension degree, where the irreducible polynomial generating the composite field has its coefficients from the binary prime field rather than the ground field, is also treated. Furthermore, certain generalizations of the proposed construction method, e.g., the use of nonprimitive elements and the construction of composite fields with special irreducible polynomials, are also discussed. Finally, we give storage-efficient conversion algorithms between the binary and composite fields when the degree of the ground field is relatively prime to the extension degree.

Index Terms—Composite and binary fields, primitive element, change of basis, AES.

1 Introduction

There has been a growing interest to develop hardware and software methods for implementing the finite field arithmetic operations particularly for cryptographic applications [13], [15], [11], [14], [16], [17], [18]. In order to obtain efficient implementations, the computations are often performed in bases other than the standard polynomial basis for the field $GF(2^k)$. Thus, we are often faced with the basis conversion problems between two different implementations of the same field such that the conversion between the two bases is efficient. For example, two such conversion problems were addressed recently [4], [3], [2]. In this paper, we are interested in the efficient conversion between the composite and binary representations.

A particularly interesting case occurs when the field $GF(2^k)$ is a composite field, i.e., k is not a prime and can be written as $k = nm$. It has been observed that efficient hardware and software implementations can be obtained for such fields [15], [11], [12], [19]. Thus, instead of performing the computations in the binary field, it is more efficient to implement the composite field to perform the computations. This methodology requires that we construct the composite field by suitably selecting n and m and also by finding an irreducible polynomial to generate the field $GF((2^n)^m)$. Furthermore, efficient methods are needed for conversion of elements between the binary and composite fields. The general methodology for constructing composite fields is well established [1]. The conversion problem between the composite and binary fields and the selection of a suitable primitive element was addressed [10]. In this work, Paar derives the conversion matrix between the fields $GF(2^k)$ and $GF((2^n)^m)$ which are already known (fixed) by their generating polynomials [10].

In this paper, we examine a slightly different problem: We construct a composite field $GF((2^n)^m)$ given the binary field $GF(2^k)$, assuming the generating polynomial of the composite field was not fixed or given a priori. We introduce practical algorithms for constructing the field $GF((2^n)^m)$ and for obtaining the conversion matrix given the binary field $GF(2^k)$. We also give efficient conversion algorithms for the case $\gcd(n, m) = 1$, which do not require the storage of the conversion matrix. Our approach requires the use of a primitive element in $GF(2^k)$ in order to construct the composite field $GF((2^n)^m)$. However, variations are possible, for example, a nonprimitive element can also be used. Furthermore, we show how to construct the composite field with a special irreducible generating polynomial, e.g., a trinomial, a pentanomial, or an equally-spaced-polynomial.

2 Fundamentals

Let $GF(2^k)$ denote the binary extension field defined over the prime field $GF(2)$. In order to construct $GF(2^k)$ and represent its elements, we need an irreducible polynomial $p(x)$ of degree k whose coefficients are in $GF(2)$. If α is a root of $p(x)$, then the set $B_1 = \{1, \alpha, \alpha^2, \ldots, \alpha^{k-1}\}$ forms a basis for the field $GF(2^k)$. An element A of $GF(2^k)$ can be expressed as $A = \sum_{i=0}^{k-1} a_i \alpha^i$, where $a_i \in GF(2)$ for $i = 0, 1, \ldots, k-1$. The row vector $(a_0, a_1, \ldots, a_{k-1})$ is called the representation of the element A in the basis B_1. Once the basis is selected, the rules for the field operations, e.g., addition, multiplication, and inversion, can be derived.

There are various ways to represent the elements of $GF(2^k)$, depending on the choice of the basis or the particular construction method. If k is the product of two integers as $k = mn$, then it is possible to derive a different representation method by defining $GF(2^k)$ over the field

• B. Sunar is with the Worcester Polytechnic Institute, Atwater Kent Room 302, 100 Institute Rd., Worcester, MA 01609. E-mail: sunar@wpi.edu.
• E. Savas is with Sabanci University, Faculty of Engineering and Natural Science, Orhanli-Tuzla, 34956 Istanbul, Turkey. E-mail: erkays@sabanciuniv.edu.
• Ç.K. Koç is with the Electrical and Computer Engineering Department, Oregon State University, Corvallis, OR 97331. E-mail: koc@ece.orst.edu.

Manuscript received 27 Sept. 2001; revised 22 Aug. 2002; accepted 31 Jan. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 115081.

$GF(2^n)$. The field $GF(2^n)$ over which the composite field is defined is called the ground field. An extension field defined over a subfield of $GF(2^k)$ other than the prime field $GF(2)$ is known as a composite field. We will use $GF((2^n)^m)$ to denote the composite field. Since there is only one field with 2^k elements, both the binary and the composite fields refer to this same field. However, their representation methods are different and it is possible to obtain one representation from the other.

Since the composite field is defined over $GF(2^n)$, we need an irreducible polynomial of degree m with coefficients in the ground field $GF(2^n)$. Let $q(x)$ be an irreducible polynomial of degree m defined over $GF(2^n)$. If β is a root of $q(x)$, then the set $B_2 = \{1, \beta, \beta^2, \dots, \beta^{m-1}\}$ forms a basis for $GF((2^n)^m)$.[1] An element $A \in GF((2^n)^m)$ can be written as $A = \sum_{i=0}^{m-1} a'_i \beta^i$, where $a'_i \in GF(2^n)$. The row vector $(a'_0, a'_1, \dots, a'_{m-1})$ is the composite field representation of A in the basis B_2. The coefficients in the composite field representation are in the ground field $GF(2^n)$ and, thus, we need to be able to perform field operations in $GF(2^n)$ in order to perform field operations in $GF((2^n)^m)$. Therefore, we need an irreducible polynomial $v(x)$ of degree n over $GF(2)$ in order to construct the ground field $GF(2^n)$. If γ is a root of $v(x)$, then the set $B_3 = \{1, \gamma, \gamma^2, \dots, \gamma^{n-1}\}$ is a basis for $GF(2^n)$, thus, an element $a \in GF(2^n)$ can be written as $a = \sum_{i=0}^{n-1} \bar{a}_i \gamma^i$, where $\bar{a}_i \in GF(2)$. The row vector $(\bar{a}_0, \bar{a}_1, \dots, \bar{a}_{n-1})$ represents the element $a \in GF(2^n)$ in the basis B_3.

3 Construction of the Composite Field

The proposed construction method depends on the availability of a primitive element α in $GF(2^k)$. If available, B_2 and B_3 are constructed so that β and γ are expressed in terms of α explicitly, as powers of α. This will facilitate conversion. Before we explain the details of the construction, we introduce the following theorem.

Theorem 1. *For $\alpha \in GF((2^n)^m)$ and $\gamma = \alpha^r$, where $r = (2^{nm} - 1)/(2^n - 1)$:*

1. $\alpha^r \in GF(2^n)$,
2. *If α is a primitive element, then γ is primitive in $GF(2^n)$.*

Proof. Result 1 is shown in [6]. For result 2, suppose α is primitive but γ is not, then $\gamma^t = 1$ for some $t < 2^n - 1$, so $\alpha^u = 1$ for $u = rt < 2^{nm} - 1$, which means that α is not primitive, a contradiction. Hence, γ must also be primitive. □

Let $GF((2^n)^m)$ be an extension field of $GF(2^n)$ and $\alpha \in GF((2^n)^m)$. The set of the elements

$$\mathcal{C} = \{\alpha, \alpha^{2^n}, \alpha^{2^{2n}}, \dots, \alpha^{2^{(m-1)n}}\}$$

is called the *conjugates* of α with respect to $GF(2^n)$. The conjugates of α are not necessarily distinct elements of $GF((2^n)^m)$. Every element $\alpha \in GF((2^n)^m)$ is associated with a monic irreducible polynomial whose coefficients are in one of the subfields of $GF((2^n)^m)$. This polynomial is called the *minimal polynomial* of α and will be denoted by $m_\alpha(x)$. Since α is a primitive element, its conjugates in $GF((2^n)^m)$ are distinct and its minimal polynomial over $GF(2^n)$ is of degree m. The minimal polynomial of α is given as:

$$m_\alpha(x) = (x + \alpha)(x + \alpha^{2^n})(x + \alpha^{2^{2n}}) \cdots (x + \alpha^{2^{(m-1)n}}).$$

The polynomial $m_\alpha(x)$ is an irreducible polynomial of degree m with coefficients in $G(2^n)$. These definitions of the conjugates and the minimal polynomial of an element of the composite field are given with respect to a subfield of the composite field. Similarly, if the prime field $GF(2)$ is taken as the subfield, then we obtain the definitions of the conjugates and minimal polynomial of an element in the binary field $GF(2^k)$. For example, let $GF(2^k)$ be the binary field with $k = nm$ and α be a primitive element in $GF(2^k)$, then the conjugates of α and its minimal polynomial can be given as:

$$\mathcal{C}' = \left(\alpha, \alpha^2, \alpha^{2^2}, \dots, \alpha^{2^{(k-1)}}\right),$$
$$m'_\alpha(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^{2^2}) \cdots (x + \alpha^{2^{(k-1)}}).$$

The polynomials $m_\alpha(x)$ and $m'_\alpha(x)$ are the minimal polynomials of the same element α with respect to the subfields $GF(2^n)$ and $GF(2)$, respectively.

The arithmetic operations in $GF((2^n)^m)$ can be implemented much faster in software [15] or using fewer gates in hardware [11] if the degree-*m* irreducible polynomial is selected such that its coefficients are in $GF(2)$ instead of $GF(2^n)$. For this, it is necessary to construct a primitive polynomial over $GF(2^n)$ with coefficients from $GF(2)$. We define $\beta = \alpha^s$ such that

$$s = \frac{2^{nm} - 1}{2^m - 1} = 1 + 2^m + 2^{2m} + 2^{3m} + \ldots + 2^{(n-1)m}. \quad (1)$$

Note that the element $\beta = \alpha^s$ is the constant term of the minimal polynomial of α with respect to $GF(2^m)$ and, thus, it also belongs to $GF(2^m)$. We then construct the minimal polynomial of β with respect to $GF(2^n)$ as:

$$m_\beta(x) = (x + \beta)\left(x + \beta^{2^n}\right)\left(x + \beta^{2^{2n}}\right) \cdots \left(x + \beta^{2^{(m-1)n}}\right). \quad (2)$$

We have the following theorem regarding the reduction of $m_\beta(x)$ given above.

Theorem 2. *If $\gcd(m, n) = 1$, the minimal polynomial $m_\beta(x)$ given by (2) is equivalent to*

$$m_\beta(x) = (x + \beta)\left(x + \beta^2\right)\left(x + \beta^{2^2}\right) \cdots \left(x + \beta^{2^{(m-1)}}\right). \quad (3)$$

Proof. It is sufficient to show that the following identity holds:

$$\left\{1, 2^n, 2^{2n}, \dots, 2^{(m-1)n}\right\} = \left\{1, 2, 2^2, \dots, 2^{(m-1)}\right\} (\text{mod } 2^m - 1). \quad (4)$$

Hence, we need to show that the exponents satisfy the following set equality:

$$\{0, n, 2n, \dots, (m-1)n\} = \{0, 1, 2, \dots, (m-1)\} \ (\text{mod } m). \quad (5)$$

1. Here, we chose a polynomial basis for convenience. However, other basis (e.g., normal basis) representations are also handled by our construction.

The LHS may be viewed as the result of the mapping $x \mapsto nx \pmod m$ applied to the elements of the set on the RHS. Since $\gcd(m,n)=1$, the inverse $n^{-1} \bmod m$ exists and the map is invertible. Hence, there is a one-to-one correspondence between the two sets. It follows that the identities (5) and (4) hold. □

The polynomial $q(x)=m_\beta(x)$ given by (2) is of exactly the same form as the minimal polynomial of β with respect to the field $GF(2)$ and, therefore, its coefficients belong to $GF(2)$. Hence, $q(x)=m_\beta(x)$ may be used to construct extensions of $GF(2^n)$ whenever $\gcd(n,m)=1$ and, at the same time, yield efficient arithmetic since its coefficients are in $GF(2)$.

4 Derivation of the Conversion Matrix

In this section, we show the derivation of the general conversion matrix from the composite field to the binary field representation. Let $p(x)$ be a degree-k primitive polynomial defined over $GF(2)$. We construct the field $GF(2^k)$ using $p(x)$, where α is a primitive element used to obtain the basis

$$B_1 = \{1, \alpha, \alpha^2, \ldots, \alpha^{k-1}\}.$$

Here, $p(x)$ is the minimal polynomial of α with respect to $GF(2)$. To obtain the composite field representation, we will obtain the minimal polynomial of α with respect to $GF(2^n)$. We denote this polynomial by $q(x)$, which is given as:

$$q(x) = (x+\alpha)\left(x+\alpha^{2^n}\right)\left(x+\alpha^{2^{2n}}\right)\cdots\left(x+\alpha^{2^{(m-1)n}}\right). \quad (6)$$

We use $q(x)$ to construct the field $GF((2^n)^m)$ defined over $GF(2^n)$, where the basis is

$$B_2 = \{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}.$$

Using the bases B_1 and B_2, we obtain two different representations of the element A as:

$$\text{Basis} \quad B_1: \quad A = \textstyle\sum_{i=0}^{k-1} a_i \alpha^i, \quad a_i \in GF(2).$$
$$\text{Basis} \quad B_2: \quad A = \textstyle\sum_{j=0}^{m-1} a'_j \alpha^j, \quad a'_j \in GF(2^n).$$

To obtain the conversion rule between these two representations of the field, we construct the basis of representation of the ground field $GF(2^n)$ in a special way. To obtain a basis, we select the constant coefficient γ of the minimal polynomial $q(x)$ with respect to the field $GF(2^n)$. γ is a coefficient of the minimal polynomial and, therefore, belongs to $GF(2^n)$. Since it is also primitive, γ's powers will generate a polynomial basis for $GF(2^n)$, $B_3 = \{1, \gamma, \gamma^2, \ldots, \gamma^{n-1}\}$. Therefore, the a'_js are represented using the basis B_3 as:

$$\text{Basis} \quad B_3: \quad a'_j = \textstyle\sum_{i=0}^{n-1} \bar{a}_{ji}\gamma^i, \quad \bar{a}_{ji} \in GF(2).$$

Furthermore, the irreducible polynomial for $GF(2^n)$ is the minimal polynomial of γ with respect to $GF(2)$, which is given as:

$$u(x) = (x+\gamma)(x+\gamma^2)\left(x+\gamma^{2^2}\right)\cdots\left(x+\gamma^{2^{n-1}}\right).$$

In order to obtain the conversion matrix from the composite field $GF((2^n)^m)$ to the binary field $GF(2^k)$, we write

$$A = \sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\gamma^i\alpha^j = \sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\alpha^{ri+j}. \quad (7)$$

Here, the terms α^{ri+j} are reduced using the generating polynomial $p(x)$ and their representations in B_1 are obtained as:

$$\alpha^{ri+j} = \sum_{h=0}^{k-1} t_{jih}\alpha^h, \quad (8)$$

where $t_{jih} \in GF(2)$ are the elements of the conversion matrix. By substituting (8) into (7), we derive the binary representation of A from its composite representation as

$$A = \sum_{h=0}^{k-1}\sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\, t_{jih}\, \alpha^h. \quad (9)$$

This sum determines the conversion matrix between two representations, as follows:

$$A = \left[\begin{array}{c|c|c|c} T_{0,0} & T_{0,1} & \cdots & T_{0,m-1} \\ \hline T_{1,0} & T_{1,1} & \cdots & T_{1,m-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline T_{m-1,0} & T_{m-1,1} & \cdots & T_{m-1,m-1} \end{array}\right] \bar{A} \quad (10)$$

where

$$A = \left[\begin{array}{c} a_0 \\ \vdots \\ a_{n-1} \\ \hline a_n \\ \vdots \\ a_{2n-1} \\ \hline \vdots \\ \hline a_{mn-n} \\ \vdots \\ a_{mn-1} \end{array}\right] \text{ and } \bar{A} = \left[\begin{array}{c} \bar{a}_{00} \\ \vdots \\ \bar{a}_{0(n-1)} \\ \hline \bar{a}_{10} \\ \vdots \\ \bar{a}_{1(n-1)} \\ \hline \vdots \\ \hline \bar{a}_{(m-1)0} \\ \vdots \\ \bar{a}_{(m-1)(n-1)} \end{array}\right].$$

Each one of the $T_{i,j}$ submatrices is an $n \times n$ matrix whose entries are from the field $GF(2)$. The entire T matrix is a $k \times k$ matrix with entries from $GF(2)$. Once the T matrix is obtained, the conversion matrix from the binary field to the composite field can be obtained by computing T^{-1}. Both of these matrices need to be precomputed and saved.

We presented a method to construct a composite field representation such that the conversion matrix is easily derived. The construction generates the irreducible polynomial used for the ground field. Alternatively, one can use a slightly modified version of our construction to generate the conversion matrix when the polynomial for the ground field $GF(2^n)$ representation is prespecified. In this case, the construction proceeds as before until $\gamma = \alpha^r$ and its associated minimal polynomial is found. Then, using the exhaustive search method introduced in [10], a mapping between γ and a primitive element in the prespecified representation is obtained. Combining the two mappings, the final conversion matrix is obtained.

4.1 Special Case of $\gcd(n,m)=1$

In Section 3, a method for constructing degree-m polynomials irreducible over $GF(2^n)$ with coefficients from $GF(2)$ was introduced. This requires that $\gcd(n,m)=1$ and

$\beta = \alpha^s$. We use $m_\beta(x)$ to construct the composite representation for $GF((2^n)^m)$. An element of $GF((2^n)^m)$ can be written as:

$$A = \sum_{j=0}^{m-1} a'_j \beta^j, \tag{11}$$

where $a'_j \in GF(2^n)$. To represent the subfield $GF(2^n)$, similar to the previous construction, we choose the basis generated by $\gamma = \alpha^r$, where $r = \frac{2^{nm}-1}{2^n-1}$, and obtain the representation of a'_j as:

$$a'_j = \sum_{i=0}^{n-1} \bar{a}_{ji} \gamma^i, \tag{12}$$

where $\bar{a}_{ji} \in GF(2)$. By combining these two representations, we obtain

$$A = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \bar{a}_{ji} \gamma^i \beta^j = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \bar{a}_{ji} \alpha^{(ri+sj)}, \tag{13}$$

where $\bar{a}_{ji} \in GF(2)$. We reduce the terms $\alpha^{(ri+sj)}$ using the generating polynomial $p(x)$ and obtain their representation the basis $\{1, \alpha, \alpha^2, \ldots, \alpha^{k-1}\}$ as:

$$\alpha^{ri+sj} = \sum_{h=0}^{k-1} t_{jih} \alpha^h, \tag{14}$$

where $t_{jih} \in GF(2)$ are the elements of the conversion matrix. By substitution, we derive the binary representation of A from its composite representation

$$A = \sum_{h=0}^{k-1} \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} \bar{a}_{ji}\, t_{jih}\, \alpha^h. \tag{15}$$

This sum gives the conversion matrix T between two representations, similar to (9).

4.2 An Example

We show the construction of the conversion matrix T from the composite field $GF((2^3)^4)$ to the binary field $GF(2^{12})$. Let $GF(2^{12})$ be constructed using the primitive polynomial $p(x) = x^{12} + x^7 + x^4 + x^3 + 1$ and α be a root of $p(x)$, thus α is a primitive element in $GF(2^{12})$. As we have shown, $\gamma = \alpha^r$ is a primitive element in the ground field $GF(2^3)$, where $r = (2^{12} - 1)/(2^3 - 1) = 585$. We construct the composite field $GF((2^3)^4)$ over the field $GF(2^3)$ using the irreducible polynomial $q(x)$ which is constructed according to (1). The irreducible polynomial $q(x)$ is of degree 4 and its coefficients are from the ground field $GF(2^3)$, which is given as follows:

$$\begin{aligned} q(x) &= (x + \alpha)\left(x + \alpha^{2^3}\right)\left(x + \alpha^{2^6}\right)\left(x + \alpha^{2^9}\right) \\ &= x^4 + \alpha^{1755} x^3 + \alpha^{2340} x^2 + \alpha^{585}. \end{aligned} \tag{16}$$

Note that α is in $GF(2^{12})$, however, $\alpha^r = \alpha^{585}$ is an element of $GF(2^3)$ and so are $\alpha^{1755} = (\alpha^{585})^3$ and $\alpha^{2340} = (\alpha^{585})^4$. Furthermore, we have $(\alpha^{585})^7 = (\alpha^{1755})^7 = (\alpha^{2340})^7 = 1$. In order to represent the elements of the ground field $GF(2^3)$, we use the constant term in $q(x)$ as the basis element, which is $\gamma = \alpha^{585}$. An element A is expressed in basis B_2 as

$$A = a'_0 + a'_1 \alpha + a'_2 \alpha^2 + a'_3 \alpha^3, \tag{17}$$

where $a'_j \in GF(2^3)$. We can express a'_j in $GF(2^3)$ using $\gamma = \alpha^{585}$ as the basis element

$$a'_j = \bar{a}_{j0} + \bar{a}_{j1}\gamma + \bar{a}_{j2}\gamma^2 = \bar{a}_{j0} + \bar{a}_{j1}\alpha^{585} + \bar{a}_{j2}\alpha^{1170}, \tag{18}$$

where $\bar{a}_{ji} \in GF(2)$ for $j = 0, 1, 2, 3$ and $i = 0, 1, 2$. Therefore, the representation of A in the composite field is found as:

$$\begin{aligned} A = \bar{a}_{00} &+ \bar{a}_{01}\alpha^{585} + \bar{a}_{02}\alpha^{1170} + \bar{a}_{10}\alpha + \bar{a}_{11}\alpha^{586} + \\ &\bar{a}_{12}\alpha^{1171} + \bar{a}_{20}\alpha^2 + \bar{a}_{21}\alpha^{587} + \bar{a}_{22}\alpha^{1172} + \bar{a}_{30}\alpha^3 + \\ &\bar{a}_{31}\alpha^{588} + \bar{a}_{32}\alpha^{1173}. \end{aligned} \tag{19}$$

The next step is to reduce the terms α^{585i+j} for $j = 0, 1, 2, 3$ and $i = 0, 1, 2$ using the generating polynomial $p(x) = x^{12} + x^7 + x^4 + x^3 + 1$. This will give us α terms in the above expression with exponents between 0 and 11. A term of the form α^{585i+j} is reduced modulo $p(x)$ by successively using the relation $\alpha^{12} = \alpha^7 + \alpha^4 + \alpha^3 + 1$. We obtain the representation of A in the binary field $GF(2^{12})$ using the basis $B_1 = \{1, \alpha, \alpha^2, \ldots, \alpha^{11}\}$ as:

$$\begin{aligned} A = a_0 &+ a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4 + a_5\alpha^5 + a_6\alpha^6 + \\ &a_7\alpha^7 + a_8\alpha^8 + a_9\alpha^9 + a_{10}\alpha^{10} + a_{11}\alpha^{11}. \end{aligned}$$

The relationship between the terms a_h for $h = 0, 1, \ldots, 11$ and $\bar{a}_{ji}$ for $j = 0, 1, 2, 3$ and $i = 0, 1, 2$ determines the elements t_{jih} of the conversion matrix T. For example, the first row of the matrix T is obtained by gathering the constant terms in the right-hand side of (19) after the substitution, which gives the constant coefficient in the left hand side, i.e., the term a_0. A simple inspection shows that:

$$a_0 = \bar{a}_{00} + \bar{a}_{01} + \bar{a}_{02} + \bar{a}_{11} + \bar{a}_{21} + \bar{a}_{22} + \bar{a}_{32},$$

which determines the first row of T. Similarly, a_1 is obtained by summing the coefficients of α as:

$$a_1 = \bar{a}_{02} + \bar{a}_{10} + \bar{a}_{11} + \bar{a}_{12} + \bar{a}_{21} + \bar{a}_{31} + \bar{a}_{32},$$

which determines the next row of T. The remaining terms a_i for $i = 2, 3, \ldots, 11$ are obtained similarly, i.e., by gathering the coefficients of α^i for $i = 2, 3, \ldots, 11$, respectively. Therefore, we obtain the 12×12 matrix T as follows:

$$A = \left[\begin{array}{ccc|ccc|ccc|ccc}
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\hline
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0
\end{array}\right] \bar{A}$$

where

$$A = \left[\begin{array}{c} a_0 \\ a_1 \\ a_2 \\ \hline a_3 \\ a_4 \\ a_5 \\ \hline a_6 \\ a_7 \\ a_8 \\ \hline a_9 \\ a_{10} \\ a_{11} \end{array}\right] \quad \text{and} \quad \bar{A} = \left[\begin{array}{c} \bar{a}_{00} \\ \bar{a}_{01} \\ \bar{a}_{02} \\ \hline \bar{a}_{10} \\ \bar{a}_{11} \\ \bar{a}_{12} \\ \hline \bar{a}_{20} \\ \bar{a}_{21} \\ \bar{a}_{22} \\ \hline \bar{a}_{30} \\ \bar{a}_{31} \\ \bar{a}_{32} \end{array}\right].$$

This matrix gives the representation of an element in the binary field $GF(2^{12})$ given its representation in the composite field $GF((2^3)^4)$. The inverse transformation, i.e., the conversion from $GF(2^{12})$ to $GF((2^3)^4)$, requires the computation of T^{-1}.

5 Comparison of the Two Methods for Deriving Conversion Matrices

In this section, we compare the complexity of our method against that of the method proposed in [10]. We will refer to the second method as the "exhaustive search method" since it requires an exhaustive search in $GF(2^k)$ in order to construct the conversion matrix between the binary and composite fields. The construction is based on finding the relation between the two primitive elements α and $\beta(=\alpha^t)$ of the two representations such that the condition

$$R(\alpha^t) = 0 \pmod{Q(y), P(x)}$$

is satisfied. Here, $R(z)$, $Q(y)$, and $P(x)$ are generating polynomials for the fields $GF(2^k)$, $GF(2^n)$, and $GF((2^n)^m)$, respectively. The lack of an explicit mathematical connection between α and β makes it difficult to compute the discrete logarithm $t = \log_\alpha(\beta)$ by direct means. Hence, an exhaustive search is performed. The method utilizes a table with $2^k - 1$ entries in order to reduce the complexity by k. The table keeps track of the conjugacy classes that are already checked. Although this method reduces the complexity of the algorithm by a factor of k, its time complexity is still exponential and can be given as:

$$\mathcal{O}\left(\frac{\Phi(2^k - 1)}{k}\right),$$

where Φ denotes the Euler totient function. This prohibits the applicability of the reduction method for even moderate values of k because of the size of the table. For a detailed explanation of the method and its complexity, see [10, pp. 9-12].

For small $k = n \cdot m$, this algorithm provides a general solution to the conversion problem. However, when k gets larger, it becomes impossible to apply this algorithm to construct the conversion matrix because of its exponential time complexity. Therefore, the algorithm might become inapplicable to this case even for the moderate values of k. Note also that the method requires primitive polynomials to construct the finite fields, $GF(2^n)$, $GF((2^n)^m)$, and $GF(2^k)$. The case in which the field polynomials are nonprimitive irreducible polynomials is not covered in the algorithm.

The new method provides a polynomial time algorithm for the same purpose. We start with analyzing the complexity of the general case studied in Section 4 and then give the complexity of special case when $\gcd(n, m) = 1$. Constructing the conversion matrix in the general case involves the computation of the powers of the primitive element in $GF(2^k)$

$$\alpha^{ri+j} \quad i = 0, 1, \ldots, n-1 \text{ and } j = 0, 1, \ldots, m-1.$$

We need to perform field multiplications in $GF(2^k)$ in order to calculate these powers of the primitive element. In the following, we present a complexity of the method for the general case in terms of the number of multiplication in $GF(2^k)$.

The first m powers of the primitive elements, $\alpha^0, \alpha^1, \ldots, \alpha^{m-1}$, come for free without any field multiplication operation since these powers do not exceed the degree of the irreducible polynomial of $GF(2^k)$. The $(m+1)$st power of the primitive element to compute is α^r and it involves an exponentiation operation in $GF(2^k)$. We can easily calculate the exact number of field multiplications needed to calculate the exponentiation since the exponent r has a special form as $r = 1 + 2^n + 2^{2n} + \ldots + 2^{(m-1)n}$. Namely, the exponent, r has m nonzero bits in its binary expansion and, thus, $m + k - n - 1$ multiplication operations are required to perform the exponentiation treating squaring operations in $GF(2^k)$ as field multiplications. Then, we need to compute the powers of the primitive elements $\alpha^{2r}, \alpha^{3r}, \ldots \alpha^{(n-1)r}$, which requires $(n-2)$ field multiplications. And, finally, we can compute the rest of the exponents,

$$\alpha^{ri+j} \quad i = 1, 2, \ldots, n-1 \text{ and } j = 1, 2, \ldots, m-1$$

by performing $(n-1) \cdot (m-1)$ multiplications, thus the total number of field multiplications to compute all the powers can be given as $2k - n - 2$.

The complexity in terms of the number of field multiplications for the special case of $\gcd(n, m) = 1$, studied in Section 5, can be computed in a similar manner. For this case, we need to calculate the following powers of the primitive element in $GF(2^k)$:

$$\alpha^{ri+sj} \quad i = 0, 1, \ldots, n-1 \text{ and } j = 0, 1, \ldots, m-1.$$

These elements can be written as:

$$\begin{array}{lllll} \alpha^0 & \alpha^s & \alpha^{2s} & \ldots & \alpha^{(m-1)s} \\ \alpha^r & \alpha^{r+s} & \alpha^{r+2s} & \ldots & \alpha^{r+(m-1)s} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \alpha^{(n-1)r} & \alpha^{(n-1)r+s} & \alpha^{(n-1)r+2s} & \ldots & \alpha^{(n-1)r+(m-1)s}. \end{array}$$

First, we calculate α^r and α^s, which require $k - n + m - 1$ and $k + n - m - 1$ field multiplications, respectively. The powers of α in the first row, $\{\alpha^0, \alpha^s, \ldots, \alpha^{(m-1)s}\}$, require $m - 2$ field multiplications. Similarly, the remaining powers in the first column require $n - 2$ field multiplications. For the rest of the powers of the primitive element, we need to perform $(n-1) \cdot (m-1)$ field multiplications. Thus, we find that the total number of multiplications to obtain the conversion matrix in the special case $\gcd(n, m) = 1$ is equal to $3k - 5$.

6 Use of Nonprimitive Elements

The proposed method of construction of the composite field $GF((2^n)^m)$ depends on the availability of a primitive element

α in $GF(2^k)$, which is the root of a degree-k primitive polynomial $p(x)$ defined over $GF(2)$. We then derive the transformation (change of basis) matrix T from $GF(2^k)$ to $GF((2^n)^m)$ using the minimal polynomial of α with respect to $GF(2^n)$ as $q(x) = m_\alpha(x)$. A question arises about the derivation of the transformation matrix in the case when a nonprimitive polynomial $h(x)$ is used to construct the field $GF(2^k)$. In this case, we cannot construct the composite field $GF((2^n)^m)$ properly and obtain the transformation matrix T. Fortunately, we do not need a specific primitive element, any primitive element would work. The primitive elements in a finite field are abundant and it is easy to find one given a representation of the field $GF(2^k)$. Let $h(x)$ be a nonprimitive irreducible polynomial used to construct the binary field $GF(2^k)$ and also let σ be a root of $h(x)$. The set

$$B_0 = \{1, \sigma, \sigma^2, \dots, \sigma^{k-1}\} \tag{20}$$

forms a basis for the field $GF(2^k)$. Let α be a primitive element in the field $GF(2^k)$. We can use the primitive element α to construct the composite field $GF((2^n)^m)$ properly, as in Section 4 (or, as in Section 4.1 if $\gcd(n,m) = 1$). According to Section 4, we have the bases B_1, B_2, and B_3 as:

$$B_1 = \{1, \alpha, \alpha^2, \dots, \alpha^{k-1}\}, \quad B_2 = \{1, \alpha, \alpha^2, \dots, \alpha^{m-1}\},$$
$$B_3 = \{1, \gamma, \gamma^2, \dots, \gamma^{n-1}\},$$

where α is a primitive element in $GF(2^k)$ and $\gamma = \alpha^r$ with $r = (2^{nm}-1)/(2^n-1)$. We represent an element of the binary field $GF(2^k)$ using the basis B_1. On the other hand, we represent an element of $GF((2^n)^m)$ using the basis B_2, where the coefficients in this representation are represented using the basis B_3. However, since an element of $GF(2^k)$ is initially given in B_0, we need to embed the change of basis matrix from B_0 to B_1 to the final transformation matrix. According to (7) in Section 4, we have

$$A = \sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\gamma^i\alpha^j = \sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\alpha^{ri+j}.$$

Assuming the representation of the primitive element α in the basis B_0 is given, we obtain the representations of the terms α^{ri+j} in B_0 for $i = 0, 1, \dots, n-1$ and $j = 0, 1, \dots, m-1$, as:

$$\alpha^{ri+j} = \sum_{h=0}^{k-1} \bar{t}_{ijh}\sigma^h. \tag{21}$$

This gives the modified transformation matrix based on the equation

$$A = \sum_{h=0}^{k-1}\sum_{j=0}^{m-1}\sum_{i=0}^{n-1} \bar{a}_{ji}\, \bar{t}_{jih}\, \sigma^h, \tag{22}$$

which is analogous to (9).

7 Composite Fields with Special Irreducible Polynomials

In Section 4.1, we constructed the composite field $GF((2^n)^m)$ for $\gcd(n,m) = 1$ in such a way that the degree-m irreducible polynomial $q(x)$ has its coefficients from $GF(2)$ rather than $GF(2^n)$. This selection yields efficient composite field arithmetic, as was demonstrated in [15]. This particular polynomial can be further specialized in the sense that it could be an irreducible trinomial or pentanomial or equally-spaced-polynomial (ESP) or all-one-polynomial (AOP). Here, we describe two methods by which we can select the degree-m irreducible polynomial generating the field $GF((2^n)^m)$. Let $q^*(x)$ be the irreducible degree-m polynomial of the desired form, e.g., trinomial, pentanomial, ESP, AOP, etc.

- The first method is to find a primitive element in α in $GF(2^k)$ such that $q(x) = q^*(x)$ where

$$\begin{aligned} s &= \frac{2^{nm}-1}{2^m-1} \\ &= 1 + 2^m + 2^{2m} + 2^{3m} + \dots + 2^{(n-1)m}, \\ \beta &= \alpha^s, \\ q(x) &= (x+\beta)(x+\beta^2)(x+\beta^{2^2})\cdots(x+\beta^{2^{(m-1)}}). \end{aligned}$$

  However, this method requires that we exhaustively try primitive elements $\alpha \in GF(2^k)$, which becomes prohibitive as k grows since it requires exponential time.
- The second method is simpler and more efficient: We go ahead with the original construction method by selecting an arbitrary primitive element α from $GF(2^k)$ and, in the end, obtain $q(x)$, which is an arbitrary irreducible polynomial of degree m over the field $GF(2)$ to construct the field $GF((2^n)^m)$. We then take the desired irreducible polynomial $q^*(x)$ and construct the change of basis matrix from the field $GF((2^n)^m)$ generated by $q(x)$ to the field $GF((2^n)^m)$ generated by $q^*(x)$. The arithmetic is performed in the latter field more efficiently due to the special structure of $q^*(x)$ and then mapped back to the former field if and when necessary.

8 Storage-Efficient Conversion

The proposed conversion methods between the binary and composite fields involve matrix multiplication. It also requires storing two matrices, each of which has $(nm)^2$ entries. In low-cost hardware implementations, we may not have sufficient amount of memory for these matrices. Fortunately, there are other approaches which do not require the conversion matrices be stored. For example, Kaliski and Yin proposed storage-efficient conversion methods for the binary fields with different bases [4], [3]. Here, we take a similar approach and introduce storage-efficient conversion algorithms between the binary and composite fields. Here, we address only the case $\gcd(n,m) = 1$ since this is the most practical case for the existing applications.

According to the setup, we have two communicating parties: The first party uses the binary field and can compute only in this field, while the second one uses the composite field and can compute only in the composite field. To each party, its own basis and arithmetic are considered to be internal, while those of the other party are external. Thus, the first party should be able to convert an element given in the second party's basis (i.e., external basis) to the first party's basis (i.e., internal basis) using only the arithmetic which is available to the first party (i.e., internal arithmetic). Similar conditions hold for the second party. In addition, conversion algorithms may also be required in the reverse directions in case only one of the parties is able to implement the necessary conversion routines. Following the terminology introduced in [4], [3], we will use the term *import* to denote conversion of a finite field element from the external basis to the internal basis using only internal arithmetic. Similarly, *export* is used to denote the conversion from the internal basis to the external basis.

We represent an element $\bar{A}$ of the composite field using

$$\begin{aligned}\bar{A} = (&\bar{a}_{00}, \bar{a}_{01}, \ldots, \bar{a}_{0,n-1}, \bar{a}_{10}, \bar{a}_{11}, \ldots, \bar{a}_{1,n-1},\\ &\ldots, \bar{a}_{m-1,0}, \bar{a}_{m-1,1}, \ldots, \bar{a}_{m-1,n-1}),\end{aligned}$$

where $\bar{a}_{ji} \in GF(2)$ for $0 \le i \le n-1$ and $0 \le j \le m-1$. This representation can also be interpreted as:

$$\bar{A} = (a'_0, a'_1, \ldots, a'_{m-1}),$$

where $a'_i = (a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}) \in GF(2^n)$ for $0 \le i \le m-1$. On the other hand, an element A of the binary field is represented using the binary string $A = (a_0, a_1, \ldots, a_{mn-1})$, where $a_i \in GF(2)$ for $0 \le i \le mn-1$.

In order to obtain the binary representation A of $\bar{A}$, we need to know the integers r and s. The primitive element α is basically the string $(0, 1, 0, \ldots 0)$, which is a convenient feature in our construction. We precompute $X = \alpha^r$ and $Y = \alpha^s$ and save these values. This computation is performed using binary field arithmetic.

Algorithm A—Importing from Composite to Binary

```
Inputs:   $\bar{A} = (\bar{a}_{00}, \bar{a}_{01}, \ldots, \bar{a}_{m-1,n-1})$
          $r$, $s$, $\alpha$, $X = \alpha^r$, and $Y = \alpha^s$
Output:   $A = (a_0, a_1, \ldots, a_{mn-1})$
Step 1:   $A := 0$
Step 2:   for $j = 0$ to $m-1$
Step 3:       for $i = 0$ to $n-1$
Step 4:           if $(\bar{a}_{ji} = 1)$ then $A = A + X^i Y^j$
Step 5:   return $A$
```

Algorithm A provides a general framework for the conversion and it is obviously not the most computationally efficient algorithm. Depending on the amount of additional memory available, one can precompute some intermediate values and use them in the multiplication process of X^i and Y^j. For instance, the values X^i for $0 \le i \le n-1$ and Y^j for $0 \le j \le m-1$ can be precomputed and saved, then multiplied as needed. This method requires less storage ($O(n+m)$ elements instead of $O(nm)$) and improves the computational efficiency by reducing the number of the multiplications. One can also use the conversion algorithms proposed in [3, Section 3.1] for improved efficiency.

Exporting from a binary to a composite field representation can be done using the algorithm proposed in [3, Section 3.5], which is adapted to our construction as Algorithm B, shown below.

Algorithm B—Exporting from Binary to Composite

```
Inputs:   $A = (a_0, a_1, \ldots, a_{mn-1})$, $r$, $s$, $\alpha$, $X$, $Y$, $V_{00}$
          where $X = \alpha^{-r}$, $Y = \alpha^{rn-s}$, and
          $(A \times V_{00})_0 = \bar{a}_{00}$
Output:   $\bar{A} = (\bar{a}_{00}, \bar{a}_{01}, \ldots, \bar{a}_{m-1,n-1})$
Step 1:   $A := A \times V_{00}$
Step 2:   for $i = 0$ to $m-1$
Step 3:       for $j = 0$ to $n-1$
Step 4:           $\bar{a}_{ij} := a_0$
Step 5:           $A := A - \bar{a}_{ij} \times V_{00}$
Step 6:           $A := A \times X$
Step 7:       $A := A \times Y$
Step 8:   return $\bar{A}$
```

For details of the derivation of V_{00}, one can refer to [4], [3].

To *import* from binary to the composite field representation, a party needs to precompute and store the primitive element α in the composite basis $\{1, \beta, \beta^2, \ldots, \beta^{m-1}\}$ using the conversion matrix. We assume the primitive element α is expressed as:

$$Z = \alpha = \alpha'_0 + \alpha'_1\beta + \ldots + \alpha'_{m-1}\beta^{m-1}.$$

Algorithm C—Importing from Binary to Composite

```
Inputs:   $A = (a_0, a_1, \ldots, a_{mn-1})$
          $Z = (\alpha'_0, \alpha'_1, \ldots, \alpha'_{m-1})$
Output:   $\bar{A} = (a'_0, a'_1, \ldots, a'_{m-1})$
Step 1:   $\bar{A} := 0$
Step 2:   if$(a_0 = 1)$ then $a'_0 := 1$
Step 3:   for $i = 1$ to $mn-1$
Step 4:       if $(a_i = 1)$ then $\bar{A} = \bar{A} + Z^i$
Step 5:   return $\bar{A}$
```

Just as in the case of Algorithm A, Algorithm C provides a framework for the conversion and optimizations via precomputation are possible.

Exporting from composite to binary representations can be accomplished using Algorithm D which is a direct adaptation of the algorithm in [3, Section 2.3]. The derivation of V_0 is explained in this reference in detail.

Algorithm D—Exporting from Composite to Binary

```
Inputs:   $\bar{A} = (a'_0, a'_1, \ldots, a'_{m-1})$, $Z^{-1}$, $V_0$ where
          $Z = (\alpha'_0, \alpha'_1, \ldots, \alpha'_{m-1})$, $(\bar{A} \times V_0)_0 = a_0$
Output:   $A = (a_0, a_1, \ldots, a_{mn-1})$
Step 1:   $\bar{A} := \bar{A} \times V_0$
Step 2:   for $i = 1$ to $mn-1$
Step 3:       $a_i := \bar{a}_{00}$
Step 4:       $\bar{A} := \bar{A} - a_i \times V_0$
Step 5:       $\bar{A} := \bar{A} \times Z^{-1}$
Step 5:   return $A$
```

9 Conclusions

We addressed a particular conversion problem in finite fields. We construct a composite field $GF((2^n)^m)$ given the binary field $GF(2^k)$ and the integers n and m such that $k = nm$ and obtain the conversion matrices between these two representations of the same field. A variation of this idea is explored in [10], in which, given both of these fields and their field polynomials, the method searches for a suitable primitive element to obtain the conversion matrix. We are motivated by the fact that, while the setup of [10] is more general, it requires exponential time since a suitable primitive element needs to be obtained. For many practical implementations, any composite field can do the job of minimizing the time or hardware complexity.

Our conversion techniques will benefit computations in finite fields of large composite extensions. Applications may vary from implementations of simple operations such as finite field inversion, as in the implementation of Rijndael [12], to more complex operations, as in the scalar-point multiplication operation used in elliptic curve cryptosystems [13].[2] The ANSI X9.62 standard [20] specifies elliptic curve cryptosystems built over the composite extensions $GF(2^{176})$, $GF(2^{208})$, $GF(2^{272})$, $GF(2^{304})$, and $GF(2^{368})$, which are known to be resistant to the attack in [7]. These applications are particularly suited for our construction method since the exhaustive search method is not feasible for such large extensions.

Acknowledgments

This material is based upon work supported by the US National Science Foundation under Grant No. ANI-0112889.

References

[1] J.V. Brawley and G.E. Schnibben, *Infinite Algebraic Extensions of Finite Fields.* Providence, R.I.: Am. Math. Soc., 1989.

[2] B.S. Kaliski Jr. and M. Liskov, "Efficient Finite Field Basis Conversion Involving Dual Bases," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç and C. Paar, eds., pp. 135-143, Berlin: Springer-Verlag, 1999.

[3] B.S. Kaliski Jr. and Y.L. Yin, "Methods and Apparatuses for Efficient Finite Field Conversion," US Patent Number 5,854,759, 29 Dec. 1998.

[4] B.S. Kaliski Jr. and Y.L. Yin, "Storage-Efficient Finite Field Basis Conversion," *Selected Areas in Cryptography,* S. Tavares and H. Meijer, eds., pp. 81-93, Berlin: Springer-Verlag, 1998.

[5] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* New York: Cambridge Univ. Press, 1994.

[6] A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields.* Boston: Kluwer Academic, 1993.

[7] M. Jacobson, A.J. Menezes, and A. Stein, "Solving Elliptic Curve Discrete Logarithm Problems Using Weil Descent," CACR Technical Technical Report CORR2001-31, Univ. of Waterloo, May 2001.

[8] IEEE Standard, "Specifications for Public Key Cryptography," IEEE P1363, 2000.

[9] IEEE Standard, "Specifications for Public Key Cryptography: Additional Techniques," IEEE P1363a, working document, 2001.

[10] C. Paar, "Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields," PhD thesis, Universität GH Essen, VDI Verlag, 1994.

[11] C. Paar, P. Fleishmann, and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents," *IEEE Trans. Computers,* vol. 48, no. 10, pp. 1025-1034, Oct. 1999.

[12] A. Rudra, P.K. Dubey, C.S. Jutla, V. Kumar, J.R. Rao, and P. Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç, D. Naccache, and C. Paar, eds., pp. 171-184, Berlin: Springer-Verlag, 2001.

[13] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck, "Fast Key Exchange with Elliptic Curve Systems," *Advances in Cryptology—CRYPTO 95,* D. Coppersmith, ed., pp. 43-56, Berlin: Springer-Verlag, 1995.

[14] J.H. Silverman, "Fast Multiplication in Finite Field $GF(2^N)$," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç and C. Paar, eds., pp. 122-134, Berlin: Springer-Verlag, 1999.

[15] E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle, "A Fast Software Implementation for Aarithmetic Operations in GF(2^n)," *Advances in Cryptology—ASIACRYPT '96,* K. Kim and T. Matsumoto, eds., pp. 65-76, Berlin: Springer-Verlag, 1996.

[16] H. Wu, "Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç and C. Paar, eds., pp. 280-291, Berlin: Springer-Verlag, 1999.

[17] H. Wu, M.A. Hasan, and I.F. Blake, "Highly Regular Architectures for Finite Field Computation Using Redundant Basis," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç and C. Paar, eds., pp. 269-279, Berlin: Springer-Verlag, 1999.

[18] A. Reyhani-Masoleh and M.A. Hasan, "On Efficient Normal Basis Multiplication," *Proc. Indocrypt 2000,* pp. 213-224, 2000.

[19] S. Oh, C H. Kim, J. Lim, and D.H. Cheon, "Efficient Normal Basis Multipliers in Composite Fields," *IEEE Trans. Computers,* vol. 49, no. 10, pp. 1133-1138, Oct. 2000.

[20] American Bankers Assoc., "X9.62 American National Standards Institute Standard, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)," Jan. 1999.

2. Note an attack based on Weil descent was shown [7] to be effective on elliptic curve discrete logarithm problems built over certain composite extensions. Hence, curve parameters should be carefully selected, to avoid potential security weaknesses.

Berk Sunar received the BSc degree in electrical and electronics engineering from Middle East Technical University in 1995 and the PhD degree in electrical and computer engineering (ECE) from Oregon State University in December 1998. After briefly working as a member of the research faculty at Oregon State University, he joined Worcester Polytechnic Institute as an assistant professor. He currently heads the Cryptography and Information Security Laboratory (CRIS). He received the US National Science Foundation CAREER award in 2002. His research interests include finite fields, elliptic curve cryptography, low-power crypographic hardware design, and computer arithmetic. He is a member of the IEEE, the IEEE Computer Society, the ACM, and the International Association of Cryptologic Research (IACR).

Erkay Savas received the BS (1989) and MS (1994) degrees in electrical engineering from the Electronics and Communications Engineering Department at Istanbul Technical University. He completed the PhD degree in the Department of Electrical and Computer Engineering (ECE) at Oregon State University in June 2000. He worked for various companies and research institutions before he joined Sabanci University as an assistant professor in 2002. He is the director of the Cryptography and Information Security (CISec) Group of Sabanci University. His research insterests include cryptography, data and communication security, high performance computing and computer arithmetic. He is a member of the IEEE.

Çetin K. Koç received the PhD (1988) and MS (1985) degrees in electrical and computer engineering from the University of California at Santa Barbara and the MS (1982) and BS (1980, summa cum laude) degrees in electrical engineering from Istanbul Technical University. He is a professor in the Department of Electrical and Computer Engineering at Oregon State University, whih he joined in 1992. He is the founder and director of the Information Security Laboratory at Oregon State University. He received the OSU College of Engineering Research Award for Outstanding and Sustained Research Leadership in 2001. His research interests are in security, cryptography, computer arithmetic, finite fields, and mobile computing. He is a member of the editorial board of the new journal *IEEE Transactions on Mobile Computing*. He is a senior member of IEEE and a member of the professional societies, IEEE Computer Society, IEEE Information Theory Society, and International Association for Cryptologic Research (IACR).