# Storage Free Basis Conversion over Composite Finite Fields of odd characteristics

M. Riaz Sial, Ersan Akyildiz

***Abstract*—We study the Finite Fields of type $F_q, q = p^{2pn}$ from the efficient implementation point of view. We found that we can represent these fields with irreducible polynomials in the form $f(x) = x^p - x - a$. By using this representation we have found a way of constructing normal basis for the field, together with transmission matrix between normal basis and algebraic basis (Polynomial Basis) of $F_q$ and vice versa. The key point is that this matrix and its inverse can be computed very efficiently without any memory requirement.**

***Index Terms**:* finite fields,composite field, basis conversion,polynomial basis, normal basis, Vandermonde matrix, reciprocal polynomial.

## I. Introduction

MOTIVATION of this work started from [2] and [3], in which authors have proposed efficient ways to compute pairing-based cryptographic protocols on hyperelliptic curves of genus 3 over finite fields of char 7, 3 respectively. Both authors used the same composite structure of finite fields of type $F_{p^{2pn}}$ for Tate-pairing computation. In [2] it is shown that using this structure as the value of prime p grows the complexity of the algorithm is reduced. Specially using $p = 7$ is more efficient as comparred to $p = 3$.

It is well known in the finite field arithmetic literature that multiplication in polynomial basis is much faster than in normal basis. On the other hand squaring in binary fields or exponentiation is much cheaper in the normal basis instead of polynomial basis as raising to the power p is just a shift operation. Normal basis are considered very attractive specially for hardware implementations due to the fact that shift operation is free to implement in hardware. To overcome the multiplication difficulty in normal basis many solutions are proposed to make it faster like Optimal normal basis(ONB) of type I and II have been introduced in many publications e.g [12] and [13]. So researchers had been always looking for the optimization of the algorithms using optimal choice of basis and algorithm. Conversion from one basis to other is a costly operation using matrix multiplication, generally requires $O(n^2)$ operations and $O(n^2)$ field elements storage [11]. Its hard specially for memory constrained devices, as one needs to store an (n x n) matrix for this operation.

Manuscript received July 17, 2013.
Riaz Sial is with the Institute of Applied Mathematics, Middle East Technical University Ankara, Turkey, e-mail: riazsial @gmail,com
Ersan Akyildiz is with the Institute of Applied Mathematics and Faculty of Arts and Sciences, Middle East Technical University Ankara, Turkey, e-mail: ersan@metu.edu.tr

In past there have been suggested solutions for this sake. In [12] Muchtadi-Alamsyah and F. Yuliawan focused on the basis conversion from polynomial to Optimal normal basis(ONB) and vice versa, in the finite fields of type $F_{2^{nm}}$, where $gcd(n, m) = 1$, $m + 1$ must be prime and 2 must be primitive in $Z^*_{m+1}$. OR for type 2 ONB $2m + 1$ must be prime and 2 is a primitive in $Z_{2m+1}$, or $2m + 1 \equiv 3 \mod 4$ and 2 generates the quadratic residues in $Z_{2m+1}$. They suggested way to covert basis with $O(m)$ memory complexity and $O(m)$ time complexity.

In [10] Authors have suggested its more efficient to work in composite fields instead of binary fields and have devised algorithm to construct composite field of type $F_{2^{nm}}$ from $F_{2^k}$ where $k = mn$. In [9] authors have suggested algorithms to convert basis in the Finite Field of type $F_{2^m}$ with the $O(m)$ memory complexity and $O(m)$ time complexity.

In the literature researchers have also used the hybrid approach for basis to benefit from both basis advantages. One such example is from Shokrollahi [8]. In this paper authors used hybrid basis system, say when multiplication is needed, two field elements are converted to polynomial basis and multiplication is carried out in polynomial basis, then the reduction is carried out in normal basis after converting back from polynomial basis. In [6] authors have demonstrated that using the hybrid type of basis representation for implementing elliptic curve protocol outperforms all existing FPGA set-up to break ECC-130 challenge.

In this paper we focused on the efficient basis conversion between polynomial and normal basis to expedite the computation of such algorithms based on composite finite fields. Here we noticed that due to Frobenius in $F_{q^p}$ over $F_q$ the computations can be much faster than the normal when p is an odd prime. Moreover same can be extended to $p^{2pn}$ type of field for all n co-prime to p. During this work luckily the basis conversion matrix happened to be a special type of Vandermonde matrix which can be constructed very efficiently. All the computations in matrix format are in the characteristic field which we show is very easy to compute as compared to parent field. Hence we purpose a storage free basis conversion for the composite Finite Fields of type $F_{p^{2pn}}$ over $F_{p^{2n}}$ which efficiently computes the presentation of an element from one basis to other with little extra computation cost . we also analyse the Tate Pairing algorithm described in [2] and give an estimate for the hardware implementation improvement for the Tate Pairing computation using our approach.

## II. PRELIMINARIES

In This part we will give an introduction to what are composite finite fields and what are the basis.

### A. Finite Fields and their Presentation

In the study of abstract algebra, a finite field or Galois field is a field in which there are finite number of elements. Finite fields are of prime importance in number theory, cryptography, error correcting codes, algebraic geometry etc.
Elements of the finite fields are presented using different basis system, a basis for $F_{p^n}$ of size $p^n$ elements over $F_p$ is a set of n elements of $F_{p^n}$ which are linearly independent. Once the type of basis is chosen then one needs to set the rules for field operations e.g multiplication, addition etc. There are mainly two types of basis known as polynomial and normal basis.

*1) Polynomial Basis:* If an element $\alpha$ of finite field is the generator of the field $F_{p^n}$ then $1, \alpha, \alpha^1, \alpha^2, ....., \alpha^{n-1}$ are called the polynomial basis. In these basis an element $A \in F_{p^n}$ can be written as

$$A = \sum_{i=0}^{n-1} a_i \alpha^i$$

, where $a_0, a_1, ....., a_{n-1} \in F_p$ are the coefficients .

*2) Normal Basis:* Let $F_{p^n}$ be the field extension of $F_p$ then a basis of $F_{p^n}$ over $F_p$ of the form $\beta, \beta^p, \beta^{p^2}, ....., \beta^{p^{n-1}}$, where $\beta$ is a suitable element of $F_{p^n}$ and other elements are its conjugates with respect to $F_p$ are called Normal basis of $F_{p^n}$ over $F_p$ .

*3) Composite Field:* Composite Field can be defined as an extension field defined over a base field which is already an extension of some field. For instance if we want to construct a field of size $P^{2pn}$elements then there exist only one finite field of this size however its representations may vary depending upon the basis and structure of the field you choose. Now we may represent the same field using $F_{p^m}$ structure, where $m = 2pn$ and the elements of the field can be represented as polynomials of degree $m-1$ whose coefficients belong to $F_p$. The same field can be represented using composite field structure $F_{P^{2pn}}$, then each element $A \in F_{P^{2pn}}$ can be represented as polynomials of degree $p-1$ whose coefficients belong to $F_{P^{2n}}$ provided p and n are co-prime. If $A \in F_{P^{2pn}}$ then

$$A = \sum_{i=0}^{p-1} a_i \alpha^i$$

, where $a_0, a_1, ....., a_{n-1} \in F_{P^{2n}}$ are the coefficients.

### B. Basis Conversion

Basis conversion refers to change the presentation of a field element from one basis to other. For cost effective solutions, better performance and compatibility of different systems its necessary to convert from one type of basis presentation to other.

## III. THEOREMS AND PROOFS

*Lemma 3.1:* $f(x) = x^p - x + 1$ is irreducible over any $F_{p^n}$ where $\gcd(p, n) = 1$.

PROOF.
In [1] theorem (3.78) says $f(x) = x^p - x - a$ ,$a \in F_p^*$ is irreducible over $F_q$ where $q = p^n$, if and only if it has no root $\in F_q$. Now [1] theorem (2.25) and [1] corollary (3.79) gives that $f(x) = x^p - x - a$ has a root in $F_q$ if and only if absolute $Tr_{F_q}(a) = 0$.
Hence $f(x) = x^p - x - a$ ,$a \in F_p^*$ is irreducible over $F_q$ if $Tr_{F_q}(a) \neq 0$. since according to [1] theorem 2.23, abs $Tr_{F_q}(a) = n * a$, so if $Tr_{F_q}(a) = 0$ then either a is zero or n is divisible by p. It completes the proof. ■

*Lemma 3.2:* Let $\beta = \alpha^{-1}$, where $\alpha$ is the root of $f(x) = x^p - x + 1$ and $F_{p^p} = F_p[x]/ < f(x) = x^p - x + 1 >$ then $\beta$ generates a normal basis namely $\{\beta, \beta^p, \beta^{p^2}, ....., \beta^{p^{p-1}}\}$ of $F_{p^p}$ over $F_p$.
PROOF.
Since reciprocal of an irreducible polynomial is also irreducible so the reciprocal of $f(x) = x^p - x + 1$ (which is irreducible over prime p from lemma 3.1) is $x^p(f(\beta)) = x^p - x^{p-1} + 1$ is also irreducible. Root of this polynomial generates normal basis. For proof reader is directed to corollary (2.6) in [4]. ■

*Theorem 3.3:* $\alpha$ be the root of irreducible polynomial $f(x) = x^p - x + 1$ over $F_p$ and $\beta = \alpha^{-1}$ then $\beta^{p^i} = 1 - (\alpha - i)^{p-1}$, where $i = 0, 1, 2, \cdots, p-1$.
PROOF.
Since $\alpha$ is the root of $f(x) = x^p - x + 1$, we have $\alpha^p = \alpha - 1$ and

$$\begin{aligned}
\beta &= f(\alpha) = a_0 + a_1\alpha + \cdots + a_k\alpha^{p-1} \\
&= \frac{1}{\alpha} = 1 - \alpha^{p-1} \\
\beta^p &= f(\alpha)^p = f(\alpha^p) = f(\alpha - 1) = 1 - (\alpha - 1)^{p-1} \\
\beta^{p^2} &= (\beta^p)^p = f(\alpha - 1)^p = f(\alpha^p - (1)^p) \\
&\quad \text{due to Frobenius} \\
&= f(\alpha - 2) = 1 - (\alpha - 2)^{p-1} \\
&\downarrow \\
&\downarrow \\
(\beta^{p^{i-1}})^p &= f(\alpha - (i-2))^p = f(\alpha^p - (i-2)) \\
&= f(\alpha - 1 - i + 2) = 1 - (\alpha - (i-1))^{p-1} \\
&\quad \text{Hence by induction} \\
\beta^{p^i} &= f(\alpha - (i-1))^p = f(\alpha - i) = 1 - (\alpha - i)^{p-1}
\end{aligned} \tag{III.1}$$

■

*Lemma 3.4:*

$$\binom{p-1}{k} \equiv \frac{(p-1)!}{k!(p-1-k)!} \equiv (-1)^k \mod p$$

where $k = 0, 1, 2, ....., p-1$.

PROOF. Note that $(p-1)! = (-1) \mod p$ known as Wilson's Theorem.

$$\begin{aligned}
\text{Let } S_k &\equiv (k!(p-1-k)!) \mod p, \\
&\text{where } 0 \le k \le p-1 \\
s_0 &\equiv 0!(p-1)! \equiv (-1) \mod p \\
\text{since } S_{k+1} &= (k+1)!(p-1-k-1)! \\
&= \frac{(k+1)k!(p-1-k)!}{p-1-k} \\
S_{k+1}(p-1-k) &\equiv s_k(k+1) \mod p \\
S_{k+1}(-1-k) &\equiv (-1)(s_k)(-k-1) \mod p \\
S_{k+1} &\equiv (-1)(s_k) \mod p \\
&\text{From the above equality and} \\
\text{Since } s_0 &\equiv (-1) \mod p \\
&\text{we can write} \\
s_k &\equiv (-1)^{k+1} \mod p \\
so\frac{(p-1)!}{k!(p-1-k)!} &\equiv \frac{(p-1)!}{s_k} \mod p \\
&\equiv \frac{-1}{(-1)^{k+1}} \equiv (-1)^k \mod p
\end{aligned}$$

■

*Theorem 3.5:* The matrix

$$M = \begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix}$$

is transition matrix from polynomial basis of the form $\{1, \alpha, \alpha^2, \cdots, \alpha^{p-1}\}$ to normal basis of the form $\{\beta, \beta^p, \beta^{p^2}, \cdots, \beta^{p^{p-1}}\}$ of the space $F_{p^p}$ over $F_p$, where $\alpha$ is the root of $f(x) = x^p - x + 1$ and $\beta = \frac{1}{\alpha}$

PROOF. Let

$$\begin{aligned}
F_{p^p} &= F_p[x]/ < f(x) = x^p - x + 1 > \\
&= F_p[\alpha]/ < f(\alpha) = \alpha^p - \alpha + 1 > \text{and let} \\
\beta &= \alpha^{-1}
\end{aligned} \tag{III.2}$$

From theorem (3.3) we know that:

$$\beta^{p^i} = 1 - (\alpha - i)^{p-1}$$

where
$\beta^{p^i} for\ 0 \le i \le p-1 = \{\beta, \beta^p, \beta^{p^2}, \cdots, \beta^{p^{p-1}}\}$ are the normal basis.

From above equation using binomial expansion formulae we get

$$\begin{aligned}
\beta^{p^i} &= 1 - (\alpha + (-i))^{p-1} \\
&= 1 - \sum_{k=0}^{p-1} \binom{p-1}{k} \alpha^{p-1-k}(-i)^k
\end{aligned} \tag{III.3}$$

From Lemma (3.4) and above Eqn. we can write

$$\begin{aligned}
(\alpha - i)^{p-1} &= \sum_{k=0}^{p-1} (-1)^k \alpha^{p-1-k}(-1)^k (i)^k \mod p \\
&= \sum_{k=0}^{p-1} (-1)^{2k} \alpha^{p-1-k} i^k \mod p \\
&= \sum_{k=0}^{p-1} \alpha^{(p-1-k)} i^k \mod p
\end{aligned} \tag{III.4}$$

From above equations we can easily conclude that,

$\beta^{p^0} = 1 - \alpha^{p-1}$, as when $i = 0$, eqn:(III.4) $= \alpha^{p-1}$

$$\beta_i = \beta^{p^i} = -\sum_{k=0}^{p-2} \alpha^{(p-1-k)} i^k \mod p, \text{ for } 1 \le i \le p-1 \tag{III.5}$$

Now if we compute the above equation for all $\beta_i$ in the tabular form we get our desired matrix and hence normal basis can be computed from polynomial basis as below:

$$\begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \dots \\ \beta^{p^{p-1}} \end{bmatrix} =$$

$$\begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix} * \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

As each row of the matrix is just raising the power of field element, so no need of memory to store the matrix. ■

*Lemma 3.6:*

$$\sum_{m=0}^{p-2} k^m \mod p, k \in F_{p^*} \equiv \begin{cases} -1 \mod p & \text{if } k = 1, \\ 0 \mod p & \text{otherwise.} \end{cases}$$

PROOF. For k = 1 its straight forward sum of $p-1$ number of 1 = -1 mod p
For $k \neq 1$ lets have

$$\sum_{m=0}^{p-2} k^m \mod p \equiv k^0 + k^1 + k^2 + \cdots + k^{p-2} \mod p \tag{III.6}$$

Since this is a simple geometric series
$a + ar + ar^2 + ... + ar^{n-1}$, where
$a = 1$
$r = k$,
$n = p - 1$.
Then we have

$$\sum_{m=0}^{p-2} k^m \mod p \equiv \frac{a(1 - r^{p-1})}{1 - r} \mod p$$

Since $r^{p-1} \equiv 1 \mod p$, for $r = k, k \neq 1, k \in F_{p^*}$ So

$$\sum_{m=0}^{p-2} k^m \mod p \equiv 0 \mod p$$

■

*Theorem 3.7:* If the matrix

$$M = \begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix}$$

The inverse of the matrix M $\mod p$ can be computed by simple transpose of the permuted rows of matrix M.

PROOF. If we analyse the matrix M of dim (p x p) contains a sub-matrix A of size (p-1) x (p-1) which is of special type known as Vandermonde matrix. This sub-matrix is shown below.

$$A = \begin{bmatrix} -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} \end{bmatrix}$$

This Vandermonde matrix is well known and is invertible.

Now if we look at each row then we observe that $R_i$ is nothing but just powers of i from 0 to $p - 2$. then we can write corresponding rows and the scalar product of two rows as under:

$$\begin{aligned} -R_i &= i^0, i^1, i^2, ...i^{p-1} \\ -R_j &= j^0, j^1, j^2, ...j^{p-1} \\ R_i * R_j &= (i*j)^0 + (i*j)^1 + (i*j)^2 + .... + (i*j)^{p-2} \\ &= k^0 + k^1 + k^2 + \cdots + k^{p-2} \mod p \\ &= \sum_{m=0}^{p-2} k^m \mod p, k \in F_{p^*} \\ &= \begin{cases} -1 & \text{if } k = 1, \text{where } k = i*j \mod p \\ 0 & \text{otherwise: due to Lemma (3.6)} \end{cases} \end{aligned}$$

Mathematically: if $R_i * R_i = 1$ and $R_i * R_j = 0$ where $i \neq j$ then such matrix is called **orthogonal matrix** and inverse of such matrix is just the transpose of it.

However in our case matrix A from equation III.7 this is not true but $R_i * R_j = -1$ only if $i * j \mod p = 1$ else 0. So to get the inverse of matrix A such that

$$A^{-1} * A = I$$

we need to compute the transpose of the matrix A with permuted rows such that column(i) of $A^{-1}$ = - $(Row(j)ofA)^t$, where $j = i^{-1} \mod p$. we can write as

$$C_i = -(R_{(\frac{1}{i} \mod p)})^t \quad \text{(III.7)}$$

where C is the column of $A^{-1}$ and R is row of matrix $A$. ■

### A. Algorithm to compute inverse of M

Inverse of the transition (p x p)matrix M is nothing but the permutation of the M as described below:
1. Column 1 = $p^{th}$ column of M but upside down.
2. Last row of the matrix is all 1.
3. Rest of the (p-1)x(p-1)matrix = $A^{-1}$ as discussed above

Example.
Here is an example for the case p=7.

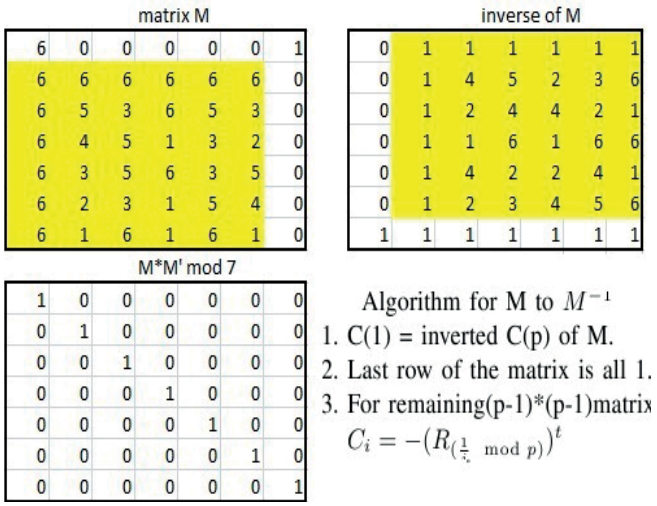


Fig. 1. Basis conversion matrix and its inverse

## IV. BASIS CONVERSION

### A. Polynomial to Normal Basis Conversion

Normal basis can be computed from polynomial basis using transition matrix as discussed in theorem (3.5). Where as the transition matrix is constructed using the algorithm given in IV-C with the time complexity of $\frac{p^2}{4}$. The equation is as shown below

$$\begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \dots \\ \beta^{p^{p-1}} \end{bmatrix} =$$

$$\begin{bmatrix} -1 & 0 & \dots & 0 & 1 \\ -(1)^0 & -(1)^1 & \dots & -(1)^{p-2} & 0 \\ -(2)^0 & -(2)^1 & \dots & -(2)^{p-2} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -(p-1)^0 & -(p-1)^1 & \dots & -(p-1)^{p-2} & 0 \end{bmatrix} * \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

*B. Normal to Polynomial Conversion*

As inverse of conversion matrix discussed in theorem (3.2) above can be now used to compute polynomial basis from normal basis using the formulae below.

$$\begin{bmatrix} 0 & (1)^0 & \dots & (p-1)^0 \\ 0 & (1)^1 & \dots & (p-1)^1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & (1)^{p-2} & \dots & (p-1)^{p-2} \\ 1 & 1 & \dots & 1 \end{bmatrix} * \begin{bmatrix} \beta \\ \beta^p \\ \beta^{p^2} \\ \dots \\ \beta^{p^{p-1}} \end{bmatrix} = \begin{bmatrix} \alpha^{p-1} \\ \alpha^{p-2} \\ \vdots \\ \alpha \\ 1 \end{bmatrix}$$

Note: Its clear from the above conversion equations that there is no need of memory to store this matrix, as it can be computed with very little effort during basis computation.

*C. Algorithm for PB-NB conversion*

Input : alpha[$a_1, a_2, .....a_p$];
Output: beta [$b_1, b_2, .....b_p$];

Algorithm:
z = (p-1)/2 ;
prm = p
beta[1] = (alpha[prm] - alpha[1]) mod prm;
for i = 1 to z do
    x = 1; m = 1;
    y1 = 0; y2 = 0; x1 = 0; x2 = 0;
    for j = 1 to z do
        y1 = (y1 - x * alpha[j]) mod prm;
        y2 = (y2 - x * alpha[z+j]) mod prm;
        x = x*i mod prm;
        x1 = (x1 - m * alpha[j]) mod prm;
        x2 = (x2 - m * alpha[z+j]) mod prm;
        m = m*(-i) mod prm;
    end for;
    beta[i+1] := (y1+ x*y2) mod prm;
    beta[prm-i+1] := (x1+ m*x2) mod prm;
end for;
output = beta;

The above algorithm takes the coefficients of an element $A \in F_{p^p}$ in the polynomial basis representation as input and computes its coefficients in the normal basis. The time complexity of the algorithm can be seen as $p^2/4$ without any storage requirement. Same algorithm can be extended for the coefficients $\alpha_i$ if they belong to $F_{p^{2^n}}$.

Moreover Normal to polynomial basis conversion can be done using by using the algorithm as shown in figure 1.
Remarks:
1. It is important to highlight here that size of Finite field changes exponentially with the size of the prime p. For instance if we use the structure $F_{p^{2^{pn}}}$ and keeping the size of the n = 29 as in [2] constant then changing the size of the prime from 3 bit to 5 bit will result in changing the size of the finite field from 1000 to 9000 bits approximately. This shows that even at very high security level, the size of the p is not much so the conversion matrix cost will not be high as compared to multiplication in the field. 2. The matrix M and its inverse as discussed above are shown to prove the inversion, however for basis transition actually there is no matrix arithmetic involved because every row can be computed one by one for each basis element as shown in the algorithm IV-C
3. Since $f(x) = x^p - x - a$ is also irreducible over $F_{p^2}$, this work can be easily extended to the fields $F_{p^{2^{pn}}}$. From this, one can use generic algorithms e.g Montgomery, Karatsuba, algorithms to have efficient computation of field operations on $F_{p^{2^{pn}}}$, for any m, where $\gcd(p, n) = 1$. As authors in [2] suggested in their algorithm.

3. Algorithm for Tate Pairing as in [2].

Let $C_b = Y^2 = X^p - X + b, b = \mp 1$ and $\rho \in F_{p^p} \in F = F_{p^{p^n}}$ be a root of the polynomial $x^p - x + 2b$ over $F_p$ *and* $\sigma \in F_{p^2} \in K = F_p^{2pn}$ be a root of the polynomial $x^2+1$. The polynomial $x^p-x+2b$ is an irreducible polynomial over the field $F_p$. Hence all its roots belong to the field $F_{p^p}$. Simultaneously, the polynomial $x^p - x + 2b$ is also irreducible over the field $F_{p^2}$ then,

Let the points $P = (\alpha, \beta) \in C_b(k)$ and
Q = $(\rho - x, \sigma y) \in F \times K$, $(x, y) \in C_b(k)$, where
$\sigma^2 = -1$, $\rho^p - \rho + 2b = 0$, $k = F_{p^n}, F = F_{p^{pn}}), and\ K = F_{p^{2^{pn}}}$, be given.

Initialization:
$f = 1$; $x = x^p$; $y = y^p$; $d = -(2n-1)b$ Calculation:
(1) for i = 1 to n,

$$\begin{aligned} \alpha &= (\alpha^p)^p \\ \beta &= (\beta^p)^p \\ g &= (\beta y\sigma - (\alpha + x + d - \rho)^{(p+1)/2} \\ f &= f^p g;\ y = -y;\ d = d + 2b \end{aligned} \tag{IV.1}$$

(2) $f = f^{p^{pn}-1}$
Return f.

As it is shown in algorithm above that there is $p^{th}$ power computation in every round and the final exponentiation, so if we change normal basis instead of polynomial as used in [2] it will make the computation of the Tate Pairing approximately 2.7 times faster. Moreover for hardware implementations the cost of basis conversion will be negligible as from the algorithm for $F_7$ multiplication discussed in [2] is nothing but just the shift of the bits, and in conversion matrix all the multiplications are in $F_7$ field.

## V. Conclusion

In this paper we proved the irreducibility of such polynomials over all primes, the normality of reciprocal polynomial, and the existence of such transition matrix for basis conversion. We give an algorithm to convert the basis with the time complexity of $\frac{p^2}{4}$ but without any storage requirement. Since we could not find any such work in odd primes characteristic composite field so its hard to compare with the binary extension fields. we have shown an important class of finite fields which can be used to benefit from the advantages of both types of basis, due to the fact that conversion cost is negligible as compared to exponentiation cost.

## VI. Future Work

Further research in this area will explore the new possibilities of improvement, as to the best of our knowledge no previous such work exist in the literature. Its direct use can be seen in algorithm in [2], however it needs to be implemented and evaluated, Implementation of these fields and finding the optimal efficient algorithms is promising area to work on.

## References


[1] Rudolf Lidl and Harald Niederreiter, *Book of Finite fields*, 2000

[2] S. B. Gashkov, A. A. Bolotov, A. A. Burtsev, S. Yu. Zhebet, and A. B. Frolov *On hardware and software implementation of arithmetic in finite fields of characteristic 7 for calculation of pairings*,journal of Mathematical Sciences, Vol. 168, No. 1, 2010.

[3] Eunjeong Lee, Hyang-Sook Lee and Yoonjin LeeFast,*computation of Tate pairing on general divisors of genus 3 hyperelliptic curves*,2006.

[4] Chih-Hua Chien, Trieu-Kien Truong, Yaotsu Chang and Chih-Hsuan Chen, *A Fast Algorithm to Determine Normal Polynomial over Finite Fields* ,IMECS 2007

[5] Kieren MacMillan and Jonathan Sondow, *Proofs of Power Sum and Binomial Coefficient Congruences Via Pascals Identity*,2010.

[6] Junfeng Fan , Daniel V. Bailey *Breaking Elliptic Curve Cryptosystems using Reconfigurable Hardware*,2010.

[7] Burton S. Kaliski Jr. and Moses Liskov, *Efficient Finite Field Basis Conversion Involving Dual Bases*,1999.

[8] J. v. z. Gathen, A. Shokrollahi, and J. Shokrollahi, *Efficient multiplication using type 2 optimal normal bases*,Lecture Notes in Computer Science,Springer, 2007.

[9] Burton S. Kaliski and yiqun Lisa Yin,1999 *storage efficient finite field basis conversion*, SAC 98, LNCS 156: 8193.

[10] Berk Sunar,Erkay Savas,Cetin K.Koc, *Constructing Composite Field Representations for Efficient Conversion*,2003.

[11] Burt Kaliski, Moses Liskov and Yiqun Lisa Yin, *Efficient Finite Field Basis Conversion Techniques*,1999.

[12] I.Muchtadi-Alamsyah and F.Yuliawan, *Basis Conversion in Composite Field*, International Journal of Mathematics and Computation vol 16 Issue 2 (2013).

[13] Berk Sunar, Cetin K. Koc,*Constructing Composite Field Representations for Efficient Conversion*, IEEE Transactions on computers, vol X, No. X, month 2003.

[14] Shigeki Kobayashi, Yasuyuki Nogami, Tatsuo Sugimura, *A Relation between SelfReciprocal Transformation and Normal Basis over Odd-Characteristic Field*, 4th ICCIT conferance 2009.