Fast Computing of Quadratic Forms of HFE Polynomials over fields of characteristic *two*

Borja Gómez
kub0x@elhacker.net

November 3, 2020

Abstract

In this paper the author introduces methods that represent elements of a Finite Field F_q as matrices that linearize certain operations like the product of elements in F_q. Since the Central Polynomial Map $\mathcal{F}(X)$ coming from the HFE scheme involves multiplication of elements in a Finite Field F_q, using a *novel method* based in Linear Algebra the Quadratic Forms resulting from the polynomial map of the Public Key can be computed in few steps and these are bounded by the matrix R that represents the linear action of the polynomial remainder modulo $f(t)$, which is the irreducible polynomial that identifies F_q. When the irreducible polynomial $f(t)$ is of the form $t^a + t^b + 1$ *modulo* 2, the matrix R is computed deterministically in few steps and all the Quadratic Forms are derived from this matrix. The research done tells that the central Polynomial Map $\mathcal{F}(X)$ is computed extremely fast, for example, in the CAS *Mathematica*, taking an HFE Polynomial, Quadratic Forms are computed in ≈ 1.4 seconds for the case $n = 128$. This raises the more general lemma that Quadratic Forms obtained from BigField schemes are entirely dependent on the selected irreducible polynomial $f(t)$ as the matrix R is conditioned by the structure of this polynomial.

1 Matrix Representation of elements in algebraic structures

An important result of Representation Theory in Mathematics is that some finite algebraic structures and their internal operations can be *linearized* in the sense that elements from these structures can be represented as a Matrix. For example, Arthur Cayley showed that every finite group G has an embedding into the Symmetric Group of n symbols, where n is the order of G. This condition is stated as the map $\phi : G \mapsto Sym(G)$ where $Sym(G)$ can be viewed as a Permutation Group of $n \times n$ orthogonal matrices as the action of G into itself [1] is an automorphism. In the case of Group Rings $K[G]$, every element $a \in K[G]$ has a matrix representation that entirely depends on the group structure [2]. Here. the product of elements $a \cdot b = M_a \cdot \overrightarrow{b}$ is achieved by transforming a to its matrix representation and b to its vector representation $\overrightarrow{b}$, then computing the product $a \cdot b$ by using Linear Algebra, this is, only using matrix multiplication.

1.1 Matrix representation of elements of a Finite Field

Following the line, it is natural to ask if the product of elements a, b in a Finite Field can be represented as a matrix equation $M_a \cdot \overrightarrow{b} = \overrightarrow{c}$. The answer is affirmative, given $a, b \in F_q$ with $q = p^n$ and the irreducible polynomial $f(t)$ *modulo* p of degree n. The product $c(t) \equiv a(t) \cdot b(t) \pmod{f(t)}$ is computed by multiplying each monomial $b_i t^i$ to $a(t)$ then reducing modulo $f(t)$:

$$c(t) = \sum_{i=0}^{n-1} c_i t^i \equiv \sum_{i=0}^{n-1} a_i t^i \cdot \sum_{i=0}^{n-1} b_i t^i \pmod{f(t)}$$
$$c(t) = \sum_{i=0}^{n-1} c_i t^i \equiv \sum_{i=0}^{n-1} a(t) \cdot b_i t^i \pmod{f(t)}$$

Define $\varphi(a(t))$ as the map that sends the polynomial $a(t)$ to its coefficient vector representation:

$$\varphi : F_{q^n} \mapsto F_q^n \quad \varphi(a(t)) \mapsto (a_0, \ldots, a_{n-1}) = \overrightarrow{a}$$

Here, notice that the product $a(t)t^i \pmod{f(t)}$ gives distinct coefficient vectors over F_p when mapped to F_p^n by φ. Put these vectors ordered in the matrix M_a so the matrix product $M_a \cdot b = c$ is a linear combination of the columns of M_a, this is, $\overrightarrow{c} = \sum_{i=0}^{n-1} b_i \varphi(a(t)t^i \equiv_{f(t)})$ which gives us the desired product where the element $\overrightarrow{c} = (c_0, \ldots, c_{n-1})$ is the coefficient vector of the polynomial $c(t)$

$$\{\varphi(a(t)) \equiv_{f(t)}, \cdots, \varphi(a(t) \cdot t^{n-1}) \equiv_{f(t)}\} \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

It results that the equation corresponding to each column can be simplified since each column holds a linear combination of the coefficients of $a(t)$, so write each column of the matrix M_a in the form $A_i \cdot \varphi(a(t)) = A_i \overrightarrow{a}$. A small remark is that the first column, that corresponds to $\varphi(a(t) * t^0)$ can be written as $I \cdot \overrightarrow{a} = \overrightarrow{a}$, this is $A_1 = I_n$.

$$\{I \cdot \overrightarrow{a}, \cdots, A_n \cdot \overrightarrow{a}\} \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

Then, given $a, b \in F_q$ obtain the coefficient vector of $c(t)$ as the result of the equation $M_a \cdot \overrightarrow{b}$ where $c = (\sum_{i=1}^{n} b_i A_i) \cdot \overrightarrow{a}$ At this point, we've reached the closed form of the matrix representation M_a along with a deterministic way of computing the product $c = ab \in F_q$. The equation of the product as further seen in subsubsection 3.1.2, is used to compute the Quadratic Forms of products of two polynomials.

1.2 Properties of the Matrix Representation

The matrices coming from the matrix representation of elements in F_q commute, and they form a subgroup of $GL(n, q)$ of order $p^n - 1$ under matrix multiplication. Some useful properties are:

- The product of two matrices representing elements $a, b \in F_q$ such that $M_a \cdot M_b = M_b \cdot M_a = M_c$ gives the matrix M_c where the first column is the coefficient vector of the element $c \in F_q$.
- Every matrix M_a has an inverse $M_{a^{-1}}$ which enables the computation of $a^{-1} \in F_q$.
- The matrix representation of a Linearised Polynomial acting on the element $a \in F_q$ is computable by using linear algebra, such that $\mathcal{L}(a(t))$ gives $M_{\mathcal{L}(a(t))}$

2 Deterministic Computation of the R Matrix

Continuing with the research, computing the matrices A_i must be done in a reasonable amount of time and avoiding symbolic expressions at any cost. Then we need an optimal strategy to build an efficient solution to the problem of finding the matrix R that represents the action of reducing a polynomial from $F_p[X]$ to F_q. It results that there exists a deterministic way of computing such matrix R. Not all irreducible polynomials $f(t)$

raise a similar structure in the R matrix, here irreducible monic polynomials containing 3 terms are studied, this is, polynomials $f(t) = t^a + t^b + 1$.

To avoid naive approaches, we know that the highest degree polynomial resulting from $a(t)t^i$ is $a(t)t^{n-1}$. Then, conclude that the highest polynomial that reduces modulo $f(t)$ is $a(t)t^{n-1}$ and it has degree $2(n-1)$.

2.1 Structure of the matrix R

As the matrix R reduces a tuple of dimension $2n-1$ into a n tuple, when $Deg(a(t)) \leq n-1$ then $R \cdot \varphi(a(t)) = R \cdot \overrightarrow{a} = \overrightarrow{a}$ is invariant. This means that R contains the $n \times n$ Identity Matrix as the first half submatrix. We call the other submatrix R_1, which is entirely dependent on the selected irreducible polynomial $f(t)$. Hence, R is represented as:

$$R = [I_n \quad R_1]$$

2.1.1 Irreducible Polynomials of the kind $t^a + t^b + 1$

As said before, the matrix R entirely depends on the selected irreducible polynomial *modulo p*. Here, an algorithm that computes the matrix R is presented, where the submatrix R_1 is computed in $\mathcal{O}((n-1)^2)$ time. The algorithms needs an irreducible polynomial $f(t)$ *modulo* 2 of the type $t^a + t^b + 1$ to correctly work.

R_1 is computed iteratively: the first column c_1 from R_1 corresponds to the coefficient vector of the irreducible polynomial $f(t) = \sum_{i=0}^{n} \alpha_i t^i$ of degree n but without taking the monic term $\alpha_n t^n$ so $c_1 = (r_{1,1}, \ldots, r_{1,n}) = (\alpha_0, \ldots, \alpha_{n-1})$. Let $c_1(t) = \varphi^{-1}(c_1)$ be the polynomial representation of the column c_1, now, the 2nd column c_2 results from the product $\varphi(c_1(t)t)$, which means that each position of c_1 is shifted cyclically *downwards* since the degree of $c_1(t)$ is being incremented by 1. When $Deg(c_i(t)t^i) = n$, the shifting process is done but applying the *binary OR* operation of the coefficient vector of $f(t)$ with the shifted vector.

Input: The coefficient vector of the irreducible polynomial $f(t)$ and its degree n
Output: The matrix R that represents the linear action of the reduction modulo $f(t)$
Function *ComputeR(f, n)* **is**
    coeffs $\leftarrow (\alpha_0, \ldots, \alpha_{n-1})$
    $R_1 \leftarrow$ Matrix(0,n,n-1)
    R_1 [Col,1] $\leftarrow$ coeffs
    for $i = 1, i < n, i++$ **do**
        col $\leftarrow$ R[Col,i]
        newcol = Array(0,n)
        for $j = 1, j < n, j++$ **do**
            if *col*$[n] == 1$ *AND* $j == n-1$ **then**
                newcol $\leftarrow$ R_1[Col,1] OR newcol
            else
                newcol[j+i] = col[j]
            end
        end
        R_1[Col,i+1]=newcol
    end
    return $[I_n | R_1]$
end

2.2 Obtaining the matrices A_i

The equation $a(t)t^i \pmod{f(t)}$ is reduced in the Finite Field by the action $R \cdot \varphi(a(t)t^i)$ and the columns from R that are involved in the linear combinations are $c_j : j \in [i, n+1]$

$$A_i = R_{[i,n+1]}$$

Thus every matrix A_i is indeed a submatrix of R, but why?. This can be a tricky question to answer but it's easily explained looking at the following example:

$$R \cdot \varphi(a(t)t^0) = R \cdot \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = I \cdot \overrightarrow{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Which leaves the coefficient tuple $\varphi(a(t)) = \overrightarrow{a}$ invariant, here $A_i = I_n$ is the submatrix from column 1 to n in R, which is the $n \times n$ Identity matrix.

Next, the polynomial $a(t)t^1$ comes in in the *right hand side or RHS*:

$$R \cdot \varphi(a(t)t^1) = R \cdot \begin{bmatrix} 0 \\ a_0 \\ \vdots \\ a_{n-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = A_2 \cdot \overrightarrow{a} = \begin{bmatrix} a_0 + \sum a_i \\ \vdots \\ a_{n-1} + \sum a_i \end{bmatrix}$$

Notice how the first coefficient from the RHS vector in this case is a 0. This zero doesn't select the first column, in fact, the colums that are selected are those ranging from 2 to $n+1$, so $A_2 = R_{[2,n+1]}$. This is the mathematical pattern that proves the claim shown in this paper.

This approach only needs an invocation of the polynomial remainder function and a single call to the function that retrieves the linear transformation i.e: CoefficientArrays. The speedup versus the naive procedure is enormous, yet allows the computation of matrices A_i as submatrices of R which is a very fast procedure. Following with the next section, it's now explained how to compute the Quadratic Forms of an HFE Polynomial by using these matrices A_i.

3 Fast computation of Quadratic Forms of an HFE Polynomial

In this paper, special kind of HFE Polynomials are studied to generate the Quadratic Forms resulting from the Public Key. Therefore define an HFE Polynomial $\mathcal{F}(X)$ [3] by:

$$\mathcal{F}(X) = \sum_{i=1}^{n} \sum_{i}^{n} \alpha_{i,j} X^{q^i + q^j} + \sum_{i=1}^{n} \beta_i X^{q^i} + \gamma \quad \alpha_{i,j}, \beta_i, \gamma \in F_q$$

which results in a polynomial in the univariate variable t having quadratic symbolic coefficients over F_p as the input variable is the polynomial $X = \sum_{i=0}^{n-1} x_i t^i$.

3.1 HFE Polynomials that are the product of two Linearised Polynomials

Moreover, some HFE Polynomials can be written as the product of two *Linearised Polynomials* over F_q. A *Linearised Polynomial* over F_q maps an input polynomial in F_q to another polynomial in F_q. A *Linearised Permutation Polynomial* maps every input polynomial to a distinct one, thus being an Automorphism in F_q.

$$\mathcal{L}(X) = \sum_{i=0}^{n-1} \alpha_i X^{q^i} \quad \alpha_i \in F_q$$

It is known that *Linearised Polynomials* define a linear transformation on the coefficient vector of the input polynomial. From now, consider $M_{\mathcal{L}}$ as the matrix representation of the *Linearised Polynomial* $\mathcal{L}$. This is:

$$\mathcal{L}(X) = Y \rightarrow M_{\mathcal{L}} \cdot \overrightarrow{x} = \overrightarrow{y}$$

Now define an HFE Polynomial $\mathcal{F}(X)$ as the multiplication of two *Linearised Polynomials*:

$$\mathcal{F}(X) = \mathcal{L}_1(X)\mathcal{L}_2(X)$$

The matrix representation seen in subsection 1.1 now changes a bit as both *Linearised Polynomials* change the structure of M_a.

3.1.1 Obtaining the map $\mathcal{F}(X)$ using Matrix Representation

In terms of matrix representation, if we represent $M_{\mathcal{L}_1(a(t))}$ we notice that the matrix $M_a = \{I \cdot \overrightarrow{a}, \cdots, A_n \cdot \overrightarrow{a}\}$ is transformed to $M_{\mathcal{L}_1(a(t))} = \{I \cdot \mathcal{M}_{\mathcal{L}1} \cdot \overrightarrow{a}, \cdots, A_n \cdot \mathcal{M}_{\mathcal{L}1} \cdot \overrightarrow{a}\}$. Hence, the HFE Polynomial can be stated as a product of matrices and a column vector:

$$\mathcal{F}(X) = \mathcal{L}_1(X)\mathcal{L}_2(X) = M_{\mathcal{L}_1(a(t))} \cdot M_{\mathcal{L}_2} \overrightarrow{a}$$

This is a linear combination of the columns of $M_{\mathcal{L}_1(a(t))}$ with the values of $\overrightarrow{a}' = M_{\mathcal{L}_2} \overrightarrow{a}$. As the columns of $M_{\mathcal{L}_1(a(t))}$ come from matrix product by a column vector, a finite sum of these yields the following expression:

$$M_{\mathcal{L}_1(a(t))} \cdot M_{\mathcal{L}_2} \overrightarrow{a} = M_{\mathcal{L}_1(a(t))} \cdot \overrightarrow{a}' = \sum_{i=1}^{n} a_i'(A_i M_{\mathcal{L}_1} \overrightarrow{a}) = (\sum_{i=1}^{n} a_i' A_i M_{\mathcal{L}_1}) \overrightarrow{a} = M_{\mathcal{F}} \overrightarrow{a}$$

Then matrix equation $M_{\mathcal{F}} \cdot \overrightarrow{a}$ results in n quadratic polynomials in n variables and represents the map $\varphi(\mathcal{L}_1(a(t))\mathcal{L}_2(a(t)))$. We've seen how to compute the symbolic matrix representation $M_{\mathcal{F}}$ of an HFE Polynomial, but it's of great interest to determine an approach to compute its Quadratic Forms, which have purely integral values.

3.1.2 Obtaining the Quadratic Forms

In the previous representation, each polynomial $f_i(X) \in \mathcal{F}(X)$ is the dot product of each row of $M_{\mathcal{F}}$ and the column vector $\overrightarrow{a}$. Notice that the dot product can be encoded or represented by a Quadratic Form since each row of $M_{\mathcal{F}}$ has symbolic coefficients on the variables $(a_0, \ldots, a_{n-1})$. This is, the dot product of the i-th row of $M_{\mathcal{F}}$ and the column vector $\overrightarrow{a}$ is $f_i =< Q_i \overrightarrow{a}, \overrightarrow{a}^T >$ which gives the Quadratic Form $\overrightarrow{a}^T Q_i \overrightarrow{a}$. Hence,

$$\mathcal{F}(X) = (a^T Q_1 a, \ldots, a^T Q_n a)$$

$$Q_i = \sum_{j=1}^{n} (A_j M_{\mathcal{L}_1})^T G_{i,j} M_{\mathcal{L}_2} = M_{\mathcal{L}_1}^T \cdot (\sum_{j=1}^{n} A_j^T G_{i,j}) \cdot M_{\mathcal{L}_2}$$

Linear Algebra allows us to express the sum of the i-th row of matrices A_j scaled by the value a_j'. To do that, notice how the transpose of any matrix A_j^T puts the i-th row in the i-th column and as the matrix $G_{i,j}$ transforms the vector a' into a zero vector where the unique non-zero value is a_j' is in the i-th position. Then to find Q_i define $G_{i,j}$ as a *Zero Matrix* such that $(G_{i,j})_{i,j} = 1$. The product $A_j^T G_{i,j}$ outputs a *Zero Matrix* where the j-th column is the i-th column of A_j^T. Mathematically, this means that the central matrix of the equation that involves Q_i is a matrix where the j-th column comes from the i-th column of A_j^T. As the matrices A_j as seen in subsection 2.2 are submatrices of the matrix R, conclude that the Quadratic Forms of an HFE Polynomial are entirely dependent on the selected irreducible polynomial $f(t)$.

3.1.3 Complete Algorithm

Note that, in subsubsection 2.1.1 the algorithm to compute the matrix R has been provided. The presented algorithm outputs the set of Quadratic Forms related to the Public Key $P(X) = T \circ \mathcal{F} \circ S(X)$ where $\mathcal{F}(X) = \mathcal{L}_1(X)\mathcal{L}_2(X)$. First computes the matrices A_j^T and using these, computes each Quadratic Form associated to the equation $f_i(X) \in \mathcal{F}(X)$.

The information needed is:

- The pair of *Linearised Polynomials* given as square matrices $\mathcal{M}_{\mathcal{L}_1}, \mathcal{M}_{\mathcal{L}_2}$ over F_2
- The pair of invertible linear transformations (S, T)
- An irreducible polynomial $f(t)$ *modulo* 2 of the type $f(t) = t^a + t^b + 1$
- The extension degree n

Input: The extension degree n, the R matrix, the matrix S and *Linearised Polynomials* $\mathcal{L}_1(X), \mathcal{L}_2(X)$
Output: The set Qs containing the Quadratic Forms of $\mathcal{F}(X)$
Function *QuadForms*$(n, R, S, \mathcal{M}_{\mathcal{L}_1}, \mathcal{M}_{\mathcal{L}_2})$ **is**
  $Qs \leftarrow$ Array[0,n]
  $As \leftarrow$ Array[0,n]
  //Precompute two matrix products
  $H_1 \leftarrow S^T \cdot \mathcal{M}_{\mathcal{L}_1}$
  $H_2 \leftarrow \mathcal{M}_{\mathcal{L}_2} \cdot S$
  for $i = 1, i \leq n, i++$ **do**
    As[i] $\leftarrow R^T_{[i,n+i]}$
  end
  for $i = 1, i \leq n, i++$ **do**
    mat $\leftarrow$ Matrix(0,n,n)
    for $j = 1, j \leq n, j++$ **do**
      mat[Col, j] $\leftarrow$ (As[j])[Col,i]
    end
    Qs[i] $\leftarrow H_1 \cdot mat \cdot H_2$
  end
  return Qs
end

Input: The extension degree n and the R matrix
Output: The set of Quadratic Forms of the Public Key $\mathcal{P}(X)$
Function *GenHFE*$(\mathcal{M}_{\mathcal{L}_1}, \mathcal{M}_{\mathcal{L}_2}, f(t), n)$ **is**
  $R \leftarrow$ *ComputeR*$(n, f(t))$
  $Qs \leftarrow$ *QuadForms*$(n, R, S, \mathcal{M}_{\mathcal{L}_1}, \mathcal{M}_{\mathcal{L}_2})$
  $P \leftarrow T \cdot Qs$
  return P
end

In the next section parametrizations are analyzed and some Public Key generation examples using both random *Linerised Polynomials* and pairs (S, T) are given.

In a nuthsell Quadratic Forms are obtained without using symbolic matrices-expressions once the matrix R is computed since any product of polynomials in F_q can be tracked down to a set of Quadratic Forms by using pure Linear Algebra.

4 Examples of Public Key Instantiation

The entire generation process of the Quadratic Forms associated to an HFE Polynomial of the type $\mathcal{F}(X) = \mathcal{L}_1(X)\mathcal{L}_2(X)$ is bounded by the computation of the matrix R_1 which comes from the matrix $R = [I_n \quad R_1]$ associated to the linear action that the polynomial remainder modulo $f(t)$ produces in a polynomial of degree at least $2(n-1)$. The R matrix is related to the selected irreducible polynomial $f(t)$ *modulo* 2. It's been commented in

subsubsection 2.1.1 how to compute such R matrix. Here some examples are given so the reader can familiarize with the results. Now, given an irreducible Polynomials of the form $t^n + t^{n-1} + 1$ the matrix R_1 has the form:

$$R_1 = \begin{pmatrix} 11\ldots\ldots1\ldots\ldots1 \\ 01\ldots\ldots1\ldots\ldots1 \\ 00\ldots\ldots1\ldots\ldots1 \\ \vdots \\ 00\ldots\ldots0\ldots\ldots1 \\ 11\ldots\ldots1\ldots\ldots1 \end{pmatrix}$$

So for example, take $f(t) = t^{127} + t^{126} + 1$. In Mathematica, the computation of the Quadratic Forms of the map $\mathcal{F} \circ S(X) = (\mathcal{L}_1 \circ S(X)) \times (\mathcal{L}_2 \circ S(X))$ takes roughly ≈ 0.48 *seconds* selecting two random *Linearised Polynomials* along with two random linear invertible transformations S, T.

The public key $P(X) = T \circ F \circ S(X)$ generation step takes ≈ 0.82 *seconds*, however, the global time spent computing is ≈ 1.41 *seconds* and not $\approx 0.82 + 0.48 = 1.3$ since the linear transformations are also randomly selected.

For example the familiy of irreducible polynomials of the form $t^{2^k-1} + t^{2^k-2} + 1$ share the matrix R_1 seen in this example. Anyways, the algorithms given in the previous section work with any irreducible polynomial of the type $t^a + t^b + 1$ *modulo* 2.

Consider the irreducible polynomial $f(t) = t^7 + t^6 + 1$ modulo 2, then the matrix R has the promised structure:

$$R = [I_7 \quad R_1] = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Consider $f(t) = t^5 + t^3 + 1$ irreducible modulo 2, then we have:

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

5 Further Applications

The author wants to remark that this research on computing Quadratic Forms of HFE Polynomials can be further investigated towards irreducible polynomials containing more than 3 terms in their expansion. Moreover, other characteristics than 2 can be studied. The research can be further extended to other schemes from the BigField family of schemes.

Beyond the aforementioned applications, the author has made a positive research on the inversion of HFE Polynomials with very high degree **where root finding algorithms are inefficient**. It results that the theory behind this paper is applicable, and speeds up considerably the public key generation process. Such investigation is being published soon in the same repository.

6 Mathematica Code

The code listed here works for characteristic two so $p = 2$ and an irreducible polynomial $f(t) = t^a + t^b + 1$ *modulo* 2 must be selected. After that, the reader can freely call the *GenHFE* function to retrieve the Quadratic Forms of the Public Key.

```
In[1]:= GenRndMat[p_, n_] := (
            mat = 0;
            rnk = 0;
            While[rnk < n,
                    mat = RandomInteger[{0, p - 1}, {n, n}];
                    rnk = MatrixRank[mat, Modulus -> p];
            ];
            Return[mat];
        )

In[2]:= ComputeR[p_, n_, irr_] := (
            coeffs = CoefficientList[irr, t, n];
            R1 = ConstantArray[0, {n, n - 1}];
            R1[[All, 1]] = coeffs;
            For[i = 1, i < n - 1, i++,
                    col = R1[[All, i]];
                    newcol = Array[0 &, n];
                    For[j = 1, j < n, j++,
                            If[col[[n]] == 1 && j == n - 1,
                                    newcol = BitOr[R1[[All, 1]], newcol];
                                    ,
                                    newcol[[j + 1]] = col[[j]];
                            ];
                    ];
                    R1[[All, i + 1]] = newcol;
            ];
            Return[ArrayFlatten[{{IdentityMatrix[n], R1}}]];
        )

In[3]:= QuadForms[p_, n_,R_,S_,L1_,L2_] := (
            Qs = Array[0 &, n];
            Rs = Array[0 &, n];
            H1 = Mod[Transpose[S].Transpose[L1], p];
            H2 = Mod[L2.S, p];
            For[i = 1, i <= n, i++,
                    Rs[[i]] = Transpose[R[[All, i ;; n + i - 1]]];
            ];
            For[i = 1, i <= n, i++,
                    mat = ConstantArray[0, {n, n}];
                    Table[
                    mat[[All, j]] = (Rs[[j]])[[All, i]],
                    {j, 1, n}];
                    Qs[[i]] = Mod[H1.mat.H2, p];
            ];
        )

In[4]:= GenHFE[p_, n_, irr_] := (
            L1 = GenRndMat[p, n];
            L2 = GenRndMat[p, n];
            S = GenRndMat[p, n];
            T = GenRndMat[p, n];
            R = ComputeR[p, n, irr];
            Print[AbsoluteTiming[QuadForms[p, n,R,S,L1,L2]][[1]]];
            Print[AbsoluteTiming[
            P = Table[Mod[T[[i]].Qs, p], {i, 1, n}]][[1]]];
            Return[P];
        )
```

Now, if we generate a Public Key having a central HFE polynomial, the inversion should work by invocating root finding over F_q. Let's give a small example to prove that the research done works.

6.1 Brief Validation Example

Let the irreducible polynomial *modulo* 2 be $f(t) = t^5 + t^3 + 1$. Given the *Linearised Polynomials* $\mathcal{L}_1(X) = (t^4 + t^2 + 1)X^{16} + (t^3 + 1)X^8 + (t^3 + t^2 + t)X^2$ and $\mathcal{L}_2(X) = (t+1)X^8 + (t^4+t^3+1)X^4 + (t^4+t)X$. The matrices associated to $\mathcal{L}_1(X)$ and $\mathcal{L}_2(X)$ are:

$$M_{\mathcal{L}_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$M_{\mathcal{L}_2} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The pair of invertible matrices S, T is given as follows:

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

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

The R matrix is:

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

And the HFE polynomial map can be computed by the product of both *Linearised Polynomials*:

$$F(X) = (t^4+t^2+t)X^{24}+X^{20}+(t^4+t^3+t+1)X^{17}+(t^4+t^3+t+1)X^{16}+(t^3+t^2+t)X^{12}+(t^4+t)X^{10}+(t^3+t^2+t+1)X^9+(t^4+t+1)X^6+(t^3+t)X^3$$

- Modify the *GenHFE* function to select the aforementioned matrices $M_{\mathcal{L}_1}, M_{\mathcal{L}_2}, S, T$ instead of generating random transformations.
- Generate P by calling $GenHFE(2, 5, M_{\mathcal{L}_1}, M_{\mathcal{L}_2}, S, T)$.
- Let the plaintext be $x = (1, 1, 0, 1, 1)$ and the ciphertext $c = P(x) = (1, 0, 0, 0, 1)$.
- Obtain $c' = T \cdot (1, 0, 0, 0, 1) = (1, 0, 0, 0, 0)$.
- Solve $F(X) - 1 = 0$ over F_q to obtain roots $(t^3 + t^2 + 1,\ t^4 + t^2,\ t^4 + t^3 + t^2,\ t^4 + t^3 + t^2 + t + 1)$.
- Recover x as $S^{-1} \cdot (1, 0, 0, 1, 0) = (1, 1, 0, 1, 1)$ where the candidate root is $t^3 + t^2 + 1$.

References

[1] Patrick J. Morandi *Group Actions* https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.623.7733&rep=rep1&type=pdf

[2] Ted Hurley *Group rings and rings of matrices* https://www.researchgate.net/publication/228928727_Group_rings_and_rings_of_matrices

[3] Jacques Patarin. *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms* https://link.springer.com/chapter/10.1007/3-540-68339-9_4