International Mathematical Forum, Vol. 8, 2013, no. 39, 1939 - 1953
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/imf.2013.310187

Are Matrices Useful in Public-Key Cryptography?[1]

Ayan Mahalanobis

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road, Pashan, Pune 411008, India
ayan.mahalanobis@gmail.com

Abstract. The discrete logarithm problem is the most prolific cryptographic primitive in use. Though the most important ones are the Diffie-Hellman problem and the decision Diffie-Hellman problem. In this paper, we discuss the discrete logarithm problem in circulant matrices – providing many particular secure instances. We compare the discrete logarithm problem in circulant matrices with that of the discrete logarithm problem in finite fields and with the discrete logarithm problem in the group of rational points of an elliptic curve.

Mathematics Subject Classification: 94A60, 20G40

Keywords: The discrete logarithm problem, circulant matrices, elliptic curve cryptosystems

1. Introduction

The purpose of this paper is to raise the question, "can one effectively use matrices in public-key cryptography"? Public-key cryptography uses many kinds of cryptographic primitives, the one that is of interest in this paper is the *discrete logarithm problem*. Phrased differently, my question is, can one use the discrete logarithm problem in matrices effectively in public-key cryptography? This paper is not to promote matrices, but to start a honest discussion on this topic. Our principal example of useful matrices in this paper is the *circulant matrices*, which we define later. Before we go into the details,

[1]Research supported by a NBHM research grant.

let me itemize the objections to the use of matrices that I have heard over the years.

- Matrix multiplication is too expensive.
- Matrices offer security advantage, i.e., the embedding degree, that is tied to the size of the matrix. In the case of elliptic curves, this embedding degree is very high in most cases. So, use elliptic curves instead of matrices.
- Computing the inverse of a matrix is too expensive.

Let us talk about the first and the third point together. It is true that in the worst case, i.e., multiplying two arbitrary matrices is hard. However, there are matrices for which multiplication is easy, for example, circulant matrices. Furthermore, in using the discrete logarithm problem, one mostly involves exponentiation. For exponentiation, there is an useful algorithm by Leedham-Green.

1.1. **Computing matrix exponent.** We introduce the reader with an amazing algorithm by Leedham-Green [7, Section 10] to compute A^m for some $A \in \mathrm{GL}(d, q)$.

Algorithm 1 (Leedham-Green)**.**
Input: a matrix A of size d over a finite field $\mathbb{F}_q$ and a positive integer m.
Output: A^m

: *Find a matrix P such that $B = P^{-1}AP$ is in the Frobenius normal form.*
: *Determine the minimal polynomial $\mathfrak{m}(x)$ of B. Since the Smith normal form is sparse, it is easy to compute the minimal polynomial – it takes $O(d^2)$ field multiplications.*
: *Compute $t^m \bmod \mathfrak{m}(t)$ in $F[t]/\mathfrak{m}(t)$ as $\mathfrak{l}(t)$.*
: *Compute $C = \mathfrak{l}(B)$*
: *Return PCP^{-1}.*

The objective of the original algorithm was to compute the power of any non-singular matrix. For our purpose this is not the case, we can choose our matrix. One way to choose that matrix is to find an irreducible polynomial $\mathfrak{m}$ of degree d over $\mathbb{F}_q$. Then choose A to be the companion matrix for that polynomial $\mathfrak{m}$. In this case, the first two steps and the last step in the above algorithm becomes redundant.

If $\mathfrak{m}$ is irreducible, the quotient $\mathbb{F}[t]/\mathfrak{m}(t)$ is a field. So the third step is an exponentiation in the field $\mathbb{F}_{q^d}$. So apart from computing the C in the above algorithm, exponentiation of a matrix with irreducible characteristic polynomial is the same as exponentiation in the finite field $\mathbb{F}_{q^d}$.

It is true that for most matrix, inverting is hard. However, there are matrices, like the *orthogonal matrices*, which satisfies the condition, $A^TA = AA^T = 1$. In this case, A^T, the transpose of A, is the inverse of A and is easy to compute.

Let us now talk about the *embedding degree*, or the *security advantage*. The concept of embedding degree has its genesis in the MOV attack [9] on the elliptic curve discrete logarithm problem. The advantage is as follows: consider an elliptic curve over $\mathbb{F}_q$, using the MOV attack one can reduce the discrete logarithm problem in the points on the curve to a discrete logarithm problem over $\mathbb{F}_{q^d}$. This d is called the security advantage, or the embedding degree. It is helpful to understand the effect of this, to run an elliptic curve cryptosystem over $\mathbb{F}_q$, the operations of the elliptic curve are actually field operations in $\mathbb{F}_q$. However, to break this discrete logarithm problem, one has to work in $\mathbb{F}_{q^d}$, for a large d this provides an obvious security advantage. The above argument is particularly relevant in the case of *index calculus attacks*. Where for a large d the index calculus even becomes exponential. However, a very large d is not really that important.

Another use of the elliptic curves are the pairing based cryptosystems, it uses the same reduction the MOV attack uses. However, if one uses an elliptic curve with high embedding degree then these cryptosystems become useless [4]. Given the fact that pairing is an important research direction in modern public-key cryptography, it is clear that very high embedding degree is not necessary [4, Section 1.1]. Moreover, the discrete logarithm problem is exponential or sub-exponential is mostly of an academic interest. At the end of the day, what is of most importance is the fact – the security of the discrete logarithm problem is the security of the discrete logarithm problem in the field $\mathbb{F}_{q^d}$. Now if one can trust the security in that field, he uses that discrete logarithm problem, otherwise move on to a different one. That is the right attitude about security. Let me now present the Menezes-Wu algorithm [10].

1.2. **The discrete logarithm problem in matrices.** The discrete logarithm problem is to find m, from A and A^m where $A \in \mathrm{GL}(d, q)$. Here $\mathrm{GL}(d, q)$ is the group of all nonsingular matrices of size d over the finite field $\mathbb{F}_q$. We present the work of Menezes & Wu [10], the best known algorithm to solve the discrete logarithm problem in matrices. This algorithm reduces the discrete logarithm problem in $\mathrm{GL}(d, q)$ to a finite (possibly trivial) extension of $\mathbb{F}_q$.

1.3. **The Menezes-Wu algorithm.**

: Input: A and A^m.
: Output: m.
: From A, compute the characteristic polynomial χ_A of A.
: From A^m, compute the characteristic polynomial χ_{A^m} of A^m.

Once the characteristic polynomials χ_A and χ_{A^m} are computed, the algorithm is as follows: find the smallest extension of $\mathbb{F}_q$ where χ_A splits. It is not hard to show that χ_{A^m} splits in that extension as well. Let $\alpha_1, \alpha_2, \ldots, \alpha_d$ be the roots of χ_A counting multiplicities, ordered in some way. Let $\beta_1, \beta_2, \ldots, \beta_d$ be the roots of χ_{A^m} counting multiplicities. Though there is no canonical ordering of the characteristic roots, following Menezes & Wu we assume that the lack of

ordering is not going to add much to the complexity of the algorithm. So we assume that there is an ordering, corresponding to that $\alpha_i^{m'_i} = \beta_i$ where $m'_i = m \bmod o(\alpha_i)$, where $o(\alpha_i)$ is the multiplicative order of α_i, for $i = 1, 2, \ldots, d$. Once this is done, we can solve the discrete logarithm problem in A by solving for m'_i and using the Chinese remainder theorem.

The obvious question is, for which A do we get the most security? It is clear from above that the Menezes & Wu algorithm reduces the discrete logarithm problem in A to the discrete logarithm problem in an extension of $\mathbb{F}_q$. That extension is the largest possible, when χ_A is irreducible. When the characteristic polynomial is irreducible, the discrete logarithm problem is effectively reduced to a discrete logarithm problem in $\mathbb{F}_{q^d}$. This is the best case scenario from the security standpoint. However, as we will soon see, in the case of circulant matrices this is not attainable. In that case, we should find A, such that χ_A has the largest possible irreducible component.

We now turn to a very special matrix, the **circulant matrices** and concentrate on those for the rest of the paper. It is known [8, 14] that the group of *circulant matrices* offers the same security of a finite field of about the same size, with **half the computational cost**. The other interesting fact about circulant matrices is the **size of the field** for a secure implementation. The arithmetic of the circulant matrices is implemented over a finite field, very similar to the case of elliptic curves, where the arithmetic is also implemented over a finite field. In the case of circulants, the size of the field can be smaller than the one used for elliptic curves. This is extensively studied in Section 5, and the results are tabulated in Table 2. To sum it up, the advantage of circulants is that it **uses smaller field and is faster**.

In this paper, we denote the group of non-singular circulant matrices of size d by $\mathrm{C}(d, q)$ and the group of special circulant matrices, i.e., circulant matrices with determinant 1, by $\mathrm{SC}(d, q)$ respectively.

2. Circulant Matrices

Definition 1 (Circulant matrix $\mathrm{C}(d, q)$)**.** *A $d \times d$ matrix over a field F is called a circulant matrix, if every row except the first row, is a right circular shift of the row above that. So a circulant matrix is defined by its first row. One can define a circulant matrix similarly using columns.*

A matrix is a two dimensional object, but a circulant matrix behaves like a one dimensional object – given by the first row or the first column. We will denote a circulant matrix C of size d, with the first row $c_0, c_1, \ldots, c_{d-1}$, by $C = \mathrm{circ}\,(c_0, c_1, c_2, \ldots, c_{d-1})$. An example of a circulant 5×5 matrix is:

$$\begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 \\ c_4 & c_0 & c_1 & c_2 & c_3 \\ c_3 & c_4 & c_0 & c_1 & c_2 \\ c_2 & c_3 & c_4 & c_0 & c_1 \\ c_1 & c_2 & c_3 & c_4 & c_0 \end{pmatrix}$$

One can define a *representer polynomial* corresponding to the circulant matrix C as $\phi_C = c_0 + c_1 x + c_2 x^2 + \ldots + c_{d-1} x^{d-1}$. The circulants form a commutative ring under matrix multiplication and matrix addition and is isomorphic to (the isomorphism being circulant matrix to the representer polynomial) $\mathcal{R} = \frac{F[x]}{x^d - 1}$. For more on circulant matrices, see [3].

We will study the discrete logarithm problem in $\mathrm{SC}(d, q)$, the special circulant matrix. It is fairly straightforward to see that one can develop a Diffie-Hellman key exchange protocol or the ElGamal cryptosystem from this discrete logarithm problem. The ElGamal cryptosystem over $\mathrm{SL}(d, q)$, *the special linear group* of size d over $\mathbb{F}_q$ is described below. Since the special circulant matrix is contained in the special linear group, this description of the ElGamal cryptosystem works for $\mathrm{SC}(d, q)$ as well.

All **fields considered** from now on are of characteristic 2.

3. The ElGamal over $\mathrm{SL}(d, q)$

Private Key: m, $m \in \mathbb{N}$.
Public Key: A and A^m, where $A \in \mathrm{SL}(d, q)$.

Encryption.

a: To send a message (plaintext) $\mathbf{v} \in \mathbb{F}_q^d$, Bob computes A^r and A^{mr} for an arbitrary $r \in \mathbb{N}$.
b: The ciphertext is $\left(A^r, A^{mr}\mathbf{v}^{\mathrm{T}}\right)$, where $\mathbf{v}^{\mathrm{T}}$ is the transpose of $\mathbf{v}$.

Decryption.

a: Alice knows m, when she receives the ciphertext $\left(A^r, A^{mr}\mathbf{v}^{\mathrm{T}}\right)$, she computes A^{mr} from A^r, then A^{-mr} and then computes $\mathbf{v}$ from $A^{mr}\mathbf{v}^{\mathrm{T}}$.

We show that the security of the ElGamal cryptosystem over $\mathrm{SL}(d, q)$, is equivalent to the Diffie-Hellman problem in $\mathrm{SL}(d, q)$. Since $\mathrm{SC}(d, q)$ is contained in $\mathrm{SL}(d, q)$, this proves that the security of ElGamal cryptosystem is equivalent to the Diffie-Hellman problem in $\mathrm{SC}(d, q)$.

Assume that Eve can solve the Diffie-Hellman problem, then from the public information, she knows A^m. From a ciphertext $\left(A^r, A^{rm}\mathbf{v}^T\right)$ she gets A^r. Since she can solve the Diffie-Hellman problem, she computes A^{rm} and can decrypt the ciphertext. The converse follows from the following theorem.

Theorem 1. *Suppose Eve has access to an oracle that can decrypt arbitrary ciphertext of the above cryptosystem for any private key, then she can solve the Diffie-Hellman problem in SL(d, q).*

Proof. Let $g = A^a$ and $h = A^b$. Eve takes an arbitrary element $\mathbf{v}$ in the vector space of dimension d on which $\mathrm{SL}(d, q)$ acts. We use the same basis used for the representation of $\mathrm{SL}(d, q)$. Then $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d)$ where $\mathbf{v}_i \in \mathbb{F}_q^\times$. Let $\widehat{\mathbf{v}}_i = (0, \ldots, \mathbf{v}_i, \ldots, 0)$ and $c = \widehat{\mathbf{v}}_i^T$. She pretends that A and A^a is a public key. Sends that information to the oracle. Then asks the oracle to decrypt (h, c). Oracle sends back to Eve, $h^{-a}c$. Eve knowing $\mathbf{v}$, computes the i^{th} column

of A^{-ab} from $h^{-a}c$. In d tries A^{ab} is found. This solves the Diffie-Hellman problem. □

4. Security of the proposed ElGamal cryptosystem

This paper is primarily focused on the discrete logarithm problem in the automorphism group of a vector space over a finite field. There are two kinds of attack on the discrete logarithm problem.

(i) The "so called" generic attacks, like the *Pollard's rho* algorithm. These attacks use a *black box* group algorithm. The time complexity of these algorithms is about the same as the square-root of the size of the group.
(ii) The other one is an *index calculus* attack. These attacks do not work in any group.

Black box group algorithms work in any group, hence they will work in $\mathrm{SC}(d, q)$ as well. The most efficient way to a use black box attack on the discrete logarithm problem, is to use the Pohlig-Hellman algorithm [15, Section 6.2.3] first. This reduces the discrete logarithm problem to the prime divisors of the order of the element (the base for the discrete logarithm) and then use the Chinese remainder theorem to construct a solution for the original discrete logarithm problem. One can use the Pollard's rho algorithm to solve the discrete logarithm problem in the prime divisors. So the whole process can be summarized as follows: the security of the discrete logarithm against generic attacks, is the security of the discrete logarithm in the largest prime divisor of the order. We cannot prevent these attacks. These generic attacks are of exponential time complexity and are not of much concern.

The biggest threat to any cryptosystem using the discrete logarithm problem is a subexponential attack like the index calculus attack [12]. It is often argued [6, 13] that there is no index calculus algorithm for most elliptic curve cryptosystems that has subexponential time complexity. This fact is often used to promote elliptic curve cryptosystem over a finite field cryptosystem [6]. So, the best we can hope from the discrete logarithm problem in $\mathrm{SC}(d, q)$ is, there is no index calculus attack or the index calculus attack becomes exponential.

The expected asymptotic complexity of the index calculus algorithm in $\mathbb{F}_{q^k}$ is

$$\exp\left((c + o(1))(\log q^k)^{\frac{1}{3}}(\log\log q^k)^{\frac{2}{3}}\right),$$

where c is a constant, see [12] and [6, Section 4]. If the degree of the extension, k, is greater than $\log^2 q$ then the asymptotic time complexity of the index calculus algorithm becomes exponential. In our case this means, if $d > \log^2 q$, the asymptotic complexity of the index calculus algorithm on circulant matrices of size d becomes exponential in $\log q$.

On the other hand, in the proposed cryptosystem, encryption and decryption works in $\mathbb{F}_q$ and breaking the cryptosystem depends on solving a discrete logarithm problem in $\mathbb{F}_{q^{d-1}}$. Implementing the index calculus attack becomes harder as the field gets bigger.

5. Is the ElGamal cryptosystem over $\mathrm{SC}(d,q)$ really useful?

For a circulant matrix over a field of even characteristic, squaring is fast. It is shown [8, Theorem 2.2] that, if $A = \mathrm{circ}\,(a_0, a_1, \ldots, a_{d-1})$, then $A^2 = \mathrm{circ}\left(a^2_{\pi(0)}, a^2_{\pi(1)}, \ldots, a^2_{\pi(d-1)}\right)$. Here π is a permutation of $\{0, 1, 2, \ldots, d-1\}$. Now the a_is belong to the underlying field $\mathbb{F}_q$ of characteristic 2. In this field, squaring is just a cyclic shift using a *normal basis* [11, Chapter 4] representation of the field elements.

It was shown [8], that if five conditions are satisfied, then the security of the discrete logarithm problem for circulant matrices of size d over $\mathbb{F}_q$ is the same as the discrete logarithm problem in $\mathbb{F}_{q^{d-1}}$.

The five conditions are:

a. The circulant matrix should be of determinant 1.
b. The matrix A should have row-sum 1.
c. The integer d is prime.
d. The polynomial $\dfrac{\chi_A}{x-1}$ is irreducible.
e. q is primitive mod d.

In short, the argument for these five conditions are the following:

Let $A = \mathrm{circ}\,(a_0, a_1, \ldots, a_{d-1})$ and let χ_A be the characteristic polynomial of A. It is easy to see that the row-sum, $a_0 + a_1 + \cdots + a_{d-1}$, sum of all elements in a row, is constant for a circulant matrix. This row-sum, α is an eigenvalue of A and belongs to $\mathbb{F}_q$. Clearly, α^m is an eigenvalue of A^m. This α and α^m can reduce a part of the discrete logarithm problem in A, to a discrete logarithm problem in the field $\mathbb{F}_q$. If the row-sum is 1, then there is no such issue. This is the reason behind the condition, the row-sum is 1.

Now assume that $\dfrac{\chi_A}{x-1} = f_1^{e_1} f_2^{e_2} \ldots f_n^{e_n}$, where each f_i is an irreducible polynomial and e_is are positive integers[2]. Then it follows, the discrete logarithm problem in A, can be reduced to discrete logarithm problems in $\dfrac{\mathbb{F}_q[x]}{f_i}$, for each i. Then one can solve the individual discrete logarithms in extensions of $\mathbb{F}_q$, put those solutions together using the Chinese remainder theorem and solve the discrete logarithm problem in A. The degree of these extensions, the size of which provides us with the better security, is maximized when $\dfrac{\chi_A}{x-1}$ is irreducible. This is the reason for $\dfrac{\chi_A}{x-1}$ is irreducible.

The ring of circulant matrices is isomorphic to $\dfrac{\mathbb{F}_q[x]}{x^d-1}$, moreover $\dfrac{\mathbb{F}_q[x]}{x^d-1}$ is isomorphic to $\dfrac{\mathbb{F}_q[x]}{x-1} \times \dfrac{\mathbb{F}_q[x]}{\Phi(x)}$, where $\Phi(x) = \dfrac{x^d-1}{x-1}$ is the d^{th} cyclotomic polynomial. If d is prime and q is primitive modulo d, then the cyclotomic

[2]Condition c. ensures that $e_i = 1$ for all i.

polynomial $\Phi(x)$ is irreducible. In this case, the discrete logarithm problem in circulant matrices reduce to the discrete logarithm problem in $\mathbb{F}_{q^{d-1}}$.

5.1. **What are the advantages of using circulant matrices?** The advantages of using circulant matrices are:

- Multiplying circulant matrices of size d over $\mathbb{F}_q$ is twice as fast compared to multiplication in the field of size $\mathbb{F}_{q^d}$ using optimal normal basis.
- Computing the inverse of a circulant matrix is easy.

Since any circulant matrix A can be represented as a polynomial of the form $f(x) = c_0 + c_1 x + \ldots + c_{d-1}x^{d-1}$. This polynomial is invertible, implies that, $\gcd\big(f(x), x^d - 1\big) = 1$. Then one can use the extended Euclid's algorithm to find the inverse. In our cryptosystem, we need to find that inverse, and it is easily computable.

We now compare the following three cryptosystems for security and speed. We do not compare the key sizes and the size of the ciphertext, as these can be decided easily.

1. The ElGamal cryptosystem using the circulant matrices of size d over $\mathbb{F}_q$.
2. The ElGamal cryptosystem using the group of an elliptic curve.
3. The ElGamal cryptosystem over $\mathbb{F}_{q^d}$.

5.2. **ElGamal over $\mathbb{F}_{q^d}$ vs. the circulants of size d over $\mathbb{F}_q$.** Clearly the circulants are the winner in this case. The circulants provide almost the same security as the ElGamal over the finite field $\mathbb{F}_{q^d}$, but multiplication in the circulants is twice as fast compared to the multiplication in the finite field $\mathbb{F}_{q^d}$.

To understand the difference, we need to understand the standard field multiplication. A field $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$, an extension of degree d, is a commutative algebra of dimension d over $\mathbb{F}_q$. Let $\alpha_0, \alpha_1, \ldots, \alpha_{d-1}$ be a basis of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$. Let $A := (a_0\alpha_0 + a_1\alpha_1 + \cdots + a_{d-1}\alpha_{d-1})$, $B := (b_0\alpha_0 + b_1\alpha_1 + \cdots + b_{d-1}\alpha_{d-1})$ and $C := A \cdot B = (c_0\alpha_0 + c_1\alpha_1 + \cdots + c_{d-1}\alpha_{d-1})$ be elements of $\mathbb{F}_{q^d}$.

The objective of multiplication is to find c_k for $k = 0, 1, \ldots, (d-1)$. Now notice that, if

$$\alpha_i\alpha_j = \sum_{k=0}^{d-1} t_{ij}^k \alpha_k,$$

we can define a $d \times d$ matrix T_k as $\{t_{ij}^k\}_{ij}$. It follows that $c_k = AT_kB^t$. The number of nonzero entries in the matrix T_k, which is constant over k, is called the *complexity* of the field multiplication [11, Chapter 5]. The following theorem is well known [11, Theorem 5.1]:

Theorem 2. *For any normal basis N of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$, the complexity of multiplication is at least $2d - 1$.*

Note that in an implementation of a field exponentiation, one can use a normal basis to use the square and multiply algorithm.

In our case, circulants of size d over a finite field $\mathbb{F}_q$, the situation is much different. We need a normal basis implementation for $\mathbb{F}_q$. However, to implement multiplication of two circulants, i.e., multiplication in $\mathcal{R} = \dfrac{\mathbb{F}_q[x]}{x^d - 1}$ we can use the basis $\left\{1, x, x^2, \ldots, x^{d-1}\right\}$.

In a very similar way as before, if $A := a_0 + a_1 x + \ldots + a_{d-1}x^{d-1}$ and $B := b_0 + b_1 x + \ldots b_{d-1}x^{d-1}$ then $C := A \cdot B = c_0 + c_1 x + \ldots + c_{d-1}x^{d-1}$. Our job is to compute c_k for $k = 0, 1, \ldots, d-1$. It follows that

$$c_k = \sum_{i=0}^{d-1} a_i b_j \text{ where } i + j = k \mod d \text{ and } 0 \leq i, j \leq d-1 \tag{1}$$

It is now clear that the complexity of the multiplication is d. Compare this to the best case situation for the *optimal normal basis* [11, Chapter 5], in which case it is $2d - 1$. So multiplying circulants take about half the time that of finite fields.

It is clear that the keysizes will be the same for both these cryptosystems.

5.3. **The elliptic curve ElGamal vs. the circulants of size d.** In this case there is no clear winner. On one hand, take the case of embedding degree. For most elliptic curves the embedding degree is very large. The embedding degree, that we refer to as the security advantage, for a circulant is tied up with the size of the matrix. For a matrix of size d, it is $d - 1$. So with circulants, it is hard to get very large embedding degree, without blowing up the size of the matrix. On the other hand, a very large embedding degree is not always necessary.

On the other hand, in elliptic curves, the order of the group is about the same as the size of the field. For 80-bit security, we must take the field to be around 2^{160}, to defend against any square-root algorithms. In the case of circulants, the order of a circulant matrix can be large. This enables us to **use smaller field for the same security**. In circulants, one can use the extended Euclid's algorithm to compute the inverse.

So, as we said before, we are not in a position to declare a clear winner in this case. However, if the size of the field is important in the implementation, and a moderate embedding degree suffices for security, then circulants are a **little ahead in the game**. We explain this by some examples in the next section.

5.3.1. *An interesting comparison.* We compare an elliptic curve cryptosystem on a non-supersingular curve $\mathrm{E}(\mathbb{F}_q)$, where we assume that q is approximately equal to 2^{160}, with circulant matrices of size 13 over $\mathbb{F}_{2^{89}}$. For instance, one can take $\mathbb{F}_q = \mathbb{F}_{2^{163}}$, the NIST recommended field. Circulant matrices of size 13 provides adequate security, see Table 2. Koblitz et. al. [6, Section 5] has done similar comparison and we borrow heavily from them. The comparison is between these two situations:

- kP where P in $\mathrm{E}(\mathbb{F}_q)$, E is a non-supersingular curve, and k is a 160-bit integer; and
- A^k, where A is a non-singular circulant matrix of size 13 over a field $\mathbb{F}_{2^{89}}$, again k is a random 160 bit integer.

We count the number of field multiplications necessary and ignore field additions. To compute the number of field multiplications necessary, we count the number of field multiplications in a square-and-multiply algorithm. One would expect that a 160-bit random integer k, will have as many ones as zeros. So to compute kP one expects about 160 elliptic curve doubling and 80 elliptic curve additions. Same is the case with circulant matrices, to compute A^k one would need about 160 squaring and 80 multiplications. Now squaring is free in circulants, however in elliptic curve doubling and addition requires 1 field inversion and 2 field multiplications in affine coordinates. There are many coordinate systems that one can use for elliptic curves. In this paper, to compare with the elliptic curves, we use the projective coordinates. Bernstein and Lange [1, Table 2.1 and Table 2.2] lists that using the projective coordinates addition needs 12 field multiplication and doubling needs 5 field multiplication. From that we estimate that to compute kP one would need 1760 field multiplications, 160×5 from doubling and 80×12 from addition.

In the case of circulant matrices, squaring is free and so there are 80 multiplications, the total is $80 \times (13)^2 = 13520$ field multiplications. However notice that the size of the field is $\mathbb{F}_{2^{89}}$ for circulants compared to $\mathbb{F}_{2^{163}}$ for the elliptic curve. It is easy to see that a field multiplication in the field $\mathbb{F}_{2^l}$ uses l^2 bit operations. So the field operation for circulant will run about 3.35 times faster than that of the elliptic curve. So 13520 circulant multiplications is equivalent to $\frac{13520}{3.35} = 4035$ field multiplications for the elliptic curve. So the circulants are slower by a factor of 2.3. However, circulants use smaller fields which is useful for lightweight cryptography.

A similar comparison with circulants of size 11 over a field of size 2^{97}, reveals that it runs neck-and-neck with elliptic curve. The complexity of multiplication in circulants depend on the size of the matrix. If there are better ways to multiply circulants or a higher extension field is chosen, that can allow us to choose smaller size for circulants and the complexity will come down dramatically.

It is clear that the keysize for circulant matrices will be larger than that of the elliptic curve cryptosystem, both satisfying the following:

1: Security of 80 bits or more from generic algorithms.
2: Security from index-calculus comparable to the field $\mathbb{F}_{2^{1000}}$, i.e., **index calculus security** of 1000 bits.

6. An algorithm

Recall that $\mathrm{C}(d,q)$ is isomorphic to $\dfrac{\mathbb{F}_q[x]}{x-1} \times \dfrac{\mathbb{F}_q[x]}{\Phi(x)}$. We now describe an algorithm to find a circulant matrix satisfying the above five conditions.

Algorithm 2 (Construct a circulant matrix satisfying five conditions)**.**

Input q, d.

- *construct* $\mathbb{F}_q$.
- $\tau(x) \leftarrow$ *A primitive polynomial of degree* $d-1$ *over* $\mathbb{F}_q$.
- *order* $\leftarrow$ *Order of the determinant of the companion matrix of* $\tau(x)$.
- *Use Chinese remainder theorem to find* $\psi(x)$ *such that* $\psi(x) = 1 \mod (x-1)$ *and* $\psi(x) = \tau(x) \mod \Phi(x)$.
- $\psi(x) \leftarrow \psi(x) \mod (x^d - 1)$.
- $A \leftarrow$ *The circulant matrix with the first row* $\psi(x)$.
- $A \leftarrow A^{order}$.

Output A.

Using Magma [2] and Algorithm 2, we were able to compute several circulant matrices over many different fields of characteristic 2. We produce part of that data in Table 1. The row with q is the size of the field extension and the row with d is the size of the circulant matrix over that field extension.

To construct the table, we considered all possible field extensions of size q, where q varies from 2^{40} to 2^{100}. For each such extension, we took all the primes, d, from 11 to 50. We then checked and tabulated the ones for which q is primitive modulo d. For every extension q and for all primes d, satisfying the primitivity condition, Algorithm 2 was used and the output matrix was checked for all the five conditions and moreover the order of the matrix A was found to be at least q^{d-3}. So, if q is primitive modulo d, our algorithm produces the desired matrix A, satisfying all five conditions. The computation was fast on a standard workstation.

So now it is clear, that there are a lot of choices for parameters for the ElGamal cryptosystem over circulant matrices. We describe our findings with some arbitrary examples. For more data see Table 2.

In the case, $\mathbf{q = 2^{89}, d = 13}$, we found the largest prime factor of the order of A to be

$$799336446517079299871633769103325135089545331 3.$$

The base two logarithm of this prime is 152.5. So even if we use the Pohlig-Hellman algorithm to reduce the discrete logarithm in A, to the discrete logarithm problem in the prime factors of the order of A, we still have the security very close to the 80-bit security from generic attacks. The security against the index calculus is the same as in $\mathbb{F}_{2^{1068}}$.

q	2^{41}	2^{43}	2^{47}	2^{49}	2^{53}	2^{55}
d	11, 13, 19, 29, 37	11, 13, 19, 29, 37	11, 13, 19, 37	11, 13, 19, 37	11, 13, 19, 29, 37	13, 19, 29, 37
q	2^{59}	2^{61}	2^{65}	2^{67}	2^{71}	2^{73}
d	11, 13, 19, 29, 37	11, 13, 19, 29, 37	13, 19, 29, 37	11, 13, 19, 29, 37	11, 13, 19, 29, 37	11, 13, 19, 29, 37
q	2^{77}	2^{79}	2^{83}	2^{85}	2^{89}	2^{95}
d	11, 13, 19, 37	11, 13, 19, 29, 37	11, 13, 19, 37	11, 13, 19, 29, 37	11, 13, 19, 29, 37	13, 19, 29, 37

TABLE 1. Fields from size 2^{40} to 2^{100} and matrices from size 11 to 50 that satisfy those five conditions.

In case of $\mathbf{q = 2^{39}, d = 29}$, the largest prime factor of A was

$$319475398781398849939742864389565 9569.$$

The logarithm base 2 of which is about 120. So from generic attack, the security is about 2^{60} or sixty bit security. From index calculus the security is the same as the security of a field of size $\mathbb{F}_{2^{1092}}$.

In the case of $\mathbf{q = 2^{45}, d = 29}$, the largest prime factor of the order of A is

$$1516917399755786418486789540081363901842 1$$

with more than 60 bit security. The security against the index calculus is equivalent to $\mathbb{F}_{2^{1260}}$.

In the case of $\mathbf{q = 2^{97}, d = 11}$, the largest prime divisor of A is

$$509968433928053143130332521088536688309634722937437691410695755991556 1,$$

the logarithm base 2 is 231. Security from generic attacks is 115 bits and from index calculus is equivalent to the field $\mathbb{F}_{2^{970}}$, i.e., 970 bits security.

In the case of $\mathbf{q = 2^{43}, d = 29}$, the largest prime factor of the order is

$$1597133026914484603924687622599912490649282490944114185598138955039971493534 9,$$

the logarithm of that is 253. So this has about 125 bit security from the generic attacks and 1204 bit security from index calculus attack.

In the case of $\mathbf{q = 2^{29}, d = 37}$, the largest prime factor is

$$3280170250141029234499886637529600808865114129658 81,$$

with logarithm 167, i.e., security of more than 80 bits from generic attacks and 1044 bits from index calculus.

Using GAP [5], we created Table 2. In this table, all extensions q, q from 2^{45} to 2^{90} and all primes from 10 to 20 are considered. For those extensions and primes, it was checked if q is primitive mod d. If that was so, then the circulant matrix A was constructed and both the generic and the index calculus security was tabulated.

Size of the extension q	Size of the matrix d	Logarithm of the largest prime	Index-calculus security in bits
2^{47}	11	115	470
	13	77	564
	19	207	846
2^{49}	11	157	490
	13	83	588
	19	112	882
2^{51}	11	92	510
2^{53}	11	129	530
	13	92	636
	19	312	954
2^{55}	13	80	660
	19	239	990
2^{57}	11	123	570
2^{59}	11	232	590
	13	91	708
	19	262	1062
2^{61}	11	157	610
	13	120	732
	19	294	1098
2^{63}	11	123	630
2^{65}	13	96	780
	19	131	1170
2^{67}	11	248	670
	13	106	804
	19	274	1206
2^{69}	11	176	690
2^{71}	11	242	710
	13	111	852
	19	281	1278
2^{73}	11	184	730
	13	103	876
	19	258	1314
2^{77}	11	184	770
	13	121	924
	19	359	1386
2^{79}	11	279	790
	13	140	948
	19	209	1422
2^{81}	11	143	810
2^{83}	11	284	830
	13	132	996
	19	443	1494
2^{85}	13	101	1020

	19	245	1530
2^{87}	11	151	870
2^{89}	11	227	890
	13	152	1068
	19	323	1602

Table 2: Security for q from 2^{45} to 2^{90} and d from 10 to 20

6.1. **Complexity of exponentiation of a circulant matrix of size d.** Let us assume, that the circulant matrix of size d is A and we are raising it to power m, i.e., compute A^m. We are using the square and multiply algorithm. We know that squaring of circulants is free, and multiplication of two circulant matrices of size d takes about d^2 field multiplications. The number of multiplications in the exponentiation is the same as the number of ones in the binary expansion of m. It is expected that a finite random string of zeros and ones will have about the same number of zeros and ones. So the expected number of ones in the binary expansion of m is $\frac{1}{2}\log_2 m$. So the expected number of field multiplications required to compute A^m is $\frac{d^2}{2}\log_2 m$.

6.2. **Further Research.** It seems that circulant matrices have some promise in public-key cryptography. Let us finish this article with two questions that I find interesting.

- In this paper, the comparison of speed between circulant matrices and elliptic curves is theoretical. One needs to do an actual implementation to verify our estimates.
- What is the situation with *orthogonal circulant matrices*? Can one get the same security with the orthogonal circulant matrices? This is important in the light that computing the inverse of an orthogonal circulant matrix is straightforward and involves no computation.

References

[1] Daniel J. Bernstein and Tanja Lange, *Analysis and optimization of elliptic-curve single-scalar multiplication*, Proceedings of Fq8, 2013, http://www.hyperelliptic.org/EFD/precomp.pdf.

[2] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993).

[3] Philip J. Davis, *Circulant matrices*, Chelsea, 1994.

[4] David Freeman, Michael Scott, and Edlyn Teske, *A taxonomy of pairing-friendly elliptic curves*, Journal of Cryptology **23** (2010), 224–280.

[5] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, 2007.

[6] Neal Koblitz, Alfred Menezes, and Scott Vanstone, *The state of elliptic curve cryptography*, Designs, Codes and Cryptogrpahy **19** (2000), 173–193.

[7] C.R. Leedham-Green and E.A. O'Brien, *Constructive recongnition of classical groups in odd characteristic*, Journal of Algebra **322** (2009), 833–881.

[8] Ayan Mahalanobis, *The discrete logarithm problem in the group of non-singular circulant matrices*, Groups Complexity Cryptology **2** (2010), 83–89.
[9] A.J. Menezes, T. Okamoto, and S.A. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE transactions on information theory **39** (1993), no. 5, 1639 – 1646.
[10] Alfred Menezes and Yi-Hong Wu, *The discrete logarithm problem in $GL(n, q)$*, Ars Combinatorica **47** (1997), 23–32.
[11] Alfred J. Menezes (ed.), *Applications of finite fields*, Kluwer, 1993.
[12] Oliver Schirokauer, Damian Weber, and Thomas Denny, *Discrete logarithm: the effectiveness of the index calculus method*, Algorithmic number theory (Talence, 1996), LNCS, vol. 1122, 1996, pp. 337–361.
[13] Joseph Silverman and Joe Suzuki, *Elliptic curve discrete logarithms and the index calculus*, Asiacrypt'98 (K. Ohra and D. Pei, eds.), LNCS, vol. 1514, 1998, pp. 110–125.
[14] Joseph H. Silverman, *Fast multiplication in Finite Fields $GF(2^n)$*, CHES'99, LNCS, vol. 1717, 1999, pp. 122–134.
[15] Douglas Stinson, *Cryptography theory and practice*, third ed., Chapman & Hall/CRC, 2006.

Received: October 7, 2013