# Compressed XTR

Masaaki Shirase[1], Dong-Guk Han[2], Yasushi Hibino[3],
and Ho Won Kim[2], Tsuyoshi Takagi[1]

[1] FUTURE UNIVERSITY-HAKODATE, JAPAN
shirasem@yahoo.co.jp, takagi@fun.ac.jp
[2] Electronics and Telecommunications Research Institute(ETRI), KOREA
{christa,khw}@etri.re.kr
[3] Japan Advanced Institute of Science and Technology(JAIST), JAPAN
hibino@jaist.ac.jp

**Abstract.** XTR public key system was introduced at Crypto 2000, which is based on a method to present elements of a subgroup of a multiplicative group of a finite field. Its application in cryptographic protocols leads to substantial savings both in communication and computational overhead without compromising security. It was shown how the use of finite extension fields and subgroups can be combined in such a way that the number of bits to be exchanged is reduced by a factor 3.
In this paper we show how to more compress the communication overhead. The compressed XTR leads to a factor 6 reduction in the representation size compared to the traditional representation and achieves as twice compactness as XTR. The computational overhead of it is a little worse than that of XTR, however the compressed XTR requires only about additional 6% computational effort. If finding 4-th roots of unity is pre-computed, then the computational overhead is only 1% compared to that of original XTR. Furthermore, the required size of public key data of it reduces about 26% from that of XTR.

## 1 Introduction

In the classical Diffie-Hellman (DH) key exchange scheme, two system parameters are fixed: a large prime number $q$ and a generator $g$ of the multiplicative group of the basic prime field $\mathbb{F}_q$. In the basic DH scheme the two parties each send a random power of $g$ to the other party. Assuming both parties know $q$ and $g$, each party transmits about $\log_2(q)$ bits to the other party.

In [4], ElGamal suggested that finite extension fields can be used instead of prime fields, but no direct computational or communication advantages where implied. In [10], Schnorr proposed a variant of the classical Diffie-Hellman scheme, in which $g$ does not generate the whole multiplicative group of the prime field $\mathbb{F}_q$, but only a small subgroup of which the order contain relatively small compared to $q$. This considerably reduces the computational cost of the DH scheme, but has no effect on the number of bits to be exchanged.

After that, it has tried to make use of traces to represent and calculate powers of elements of a subgroup of a finite field to achieve efficient and compact subgroup representation. The LUC cryptosystem uses the trace over $\mathbb{F}_q$ to represent elements of the order $q+1$ subgroup of $\mathbb{F}_{q^2}^*$ [12]. Compared to the traditional representation LUC leads to a a factor 2 reduction in the representation size. The variant described in [5] uses the subgroup of order $q^2+q+1$ of $\mathbb{F}_{q^3}^*$ instead, but as a result sizes are

reduced by only a factor 1.5. In [2], Brouwer et al. introduced for the first time how the use of finite extension fields and subgroups can be combined in such a way that the number of bits to be exchanged is reduced by a factor 3. More specifically, it was shown that elements of an order $p$ subgroup of $\mathbb{F}_{q^6}^*$ can be represented using $2\log_2(q)$ bits if $p$ divides $q^2 - q + 1$. Despite its communication efficiency, the method of it is rather troublesome and computationally not particularly efficient.

In 2000 Lenstra-Verheul introduced XTR [7], a cryptosystem using the trace over $\mathbb{F}_{q^2}$ to represent elements of the order $q^2 - q + 1$ subgroup of $\mathbb{F}_{q^6}^*$, there by achieving a factor 3 size reduction. Also, the resulting calculations are appreciably faster than using the standard representation. XTR of security equivalent to 1024-bit RSA achieves speed comparable to cryptosystems based on random elliptic curves over random prime fields (ECC) of equivalent security. The corresponding XTR public keys are only about twice as large as ECC keys, assuming global system parameters - without the last requirement the sizes of XTR and ECC public keys are the same. Furthermore, parameter initialization from scratch for XTR takes a negligible amount of computing time, unlike RSA and ECC. Combined with its very easy programmability, this makes XTR an excellent public key system for a very wide variety of environments, ranging from smart cards to web servers.

In this paper we present a greatly improved version of XTR that leads to a factor 6 reduction in the representation size compared to the traditional representation. That is to say, we achieve a factor 2 reduction compared to the original XTR. We show that if the characteristic of $q$ is three, i.e. $q = 3^{2k-1}$ for some integer $k$, then we can use the trace over $\mathbb{F}_q$ to represent elements of the order $q - \sqrt{3q} + 1$ subgroup of $\mathbb{F}_{q^6}^*$. Also, the resulting calculations such as exponentiations are as faster as that of XTR. Given $Tr_{(q^6,q)}(g)$ and $n$, $Tr_{(q^6,q)}(g^n)$ takes about 1381 multiplications in $\mathbb{F}_q$, which is only about 6% increase compared to the cost of computation of $Tr_{(q^6,q^2)}(h^n)$ for given $Tr_{(q^6,q^2)}(h)$ and $n$, where the size of $n$ is 160 bits. If $q$ is fixed, then finding square root of $-1$ can be pre-computed. In this case, the computational overhead is only 1% compared to that of original XTR. Furthermore, the required size of public key data of it reduces about 26% from that of the original XTR.

In Section 2 we describe XTR, and in Section 3 we introduce XTR over characteristic three, which achieves a factor 2 reduction in the representation size compared to XTR. Section 4 shows efficient calculations of XTR exponentiation over characteristic three. Applications and comparisons to the original XTR are given in Section 5.

## 2 XTR

### 2.1 Description of XTR

XTR uses a subgroup of prime order $p$ of the order $q^2 - q + 1$ subgroup of $\mathbb{F}_{q^6}^*$. The latter group is referred to as the *XTR supergroup* denoted as $G_{q^2-q+1}$ and the order $p$ subgroup $G_p$ is referred to as the *XTR group*. The XTR supergroup $G_{q^2-q+1}$ is not contained in any proper subfield of $\mathbb{F}_{q^6}$ due to the following fact.

**Fact 1** [8] *Let $p$ be a prime factor of $\Phi_m(q)$, where $m$-th cyclotomic polynomial for a positive integer $m$ not divisible by $q$. Then the subgroup $G_p$ of $\mathbb{F}_{q^m}^*$ is not contained in any proper subfield of $\mathbb{F}_{q^m}$.*

Combined with the choice of $p$ it follows that computing discrete logarithms in $G_p$ is as hard, in general, as it is in $\mathbb{F}_{q^6}^*$ [7].

Before describing XTR more detail, we introduce two definitions about optimal normal basis.

**Definition 1.** *Type I Optimal Normal Basis (Type-I ONB)*
*If $m+1$ is a prime and $q$ is a generator of $\mathbb{F}_{m+1}^*$, then the set $\{\omega^m, \omega^{m-1}, \cdots, \omega^2, \omega\}$ forms an optimal normal basis of type I in $\mathbb{F}_{q^m}$ and called Type-I ONB. Here, $\omega$ is the primitive $(m+1)$-th root of unity.*

**Definition 2.** *Type II Optimal Normal Basis (Type-II ONB)*
*If $2m+1$ is a prime and either of the following two conditions holds,*

- *$q$ is a primitive root module $2m+1$,*
- *$q$ is a quadratic residue module $2m+1$ and $q \not\equiv 1$ mod $(2m+1)$,*

*then the set $\{\beta^m, \beta^{m-1}, \cdots, \beta^2, \beta\}$ forms an optimal normal basis of type II in $\mathbb{F}_{q^m}$ and called Type-II ONB. Here, $\beta = \gamma + \gamma^{-1}$ and $\gamma$ is the primitive $(2m+1)$-th root of unity.*

XTR uses $\mathbb{F}_{q^2}$ arithmetic to achieve $\mathbb{F}_{q^6}$ security, without requiring explicit construction of $\mathbb{F}_{q^6}$. Let $q$ be a prime that is 2 mod 3. It follows that $(X^3-1)/(X-1) = X^2+X+1$ is irreducible over $\mathbb{F}_q$ and the zeros $\alpha$ and $\alpha^q$ of it form an Type-I ONB for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$. In XTR elements of $G_p$ are represented by their trace over $\mathbb{F}_{q^2}$. For $h \in \mathbb{F}_{q^6}^*$ the trace $Tr_{(q^6,q^2)}(h)$ over $\mathbb{F}_{q^2}$ is defined as the sum of the conjugates over $\mathbb{F}_{q^2}$ of $h$, i.e. $Tr_{(q^6,q^2)}(h) = h + h^{q^2} + h^{q^4} \in \mathbb{F}_{q^2}$. Let $p$ and $q$ be primes with $p$ dividing $q^2-q+1$. Also let $h$ be a generate of $G_p$ and let $c = Tr_{(q^6,q^2)}(h)$. Suggested lengths to provide adequate levels of security are $\log_2(q) \approx 170$ and $\log_2(p) \approx 160$.

$c_n$ denotes $Tr_{(q^6,q^2)}(h^n) \in \mathbb{F}_{q^2}$, for some $q$ and $h$ of order $p$ dividing $q^2-q+1$ as above. Efficient computation of $c_n$ given $q, p$ and $c$ depends on the recurrence relation

$$c_{u+v} = c_u c_v - c_v^q c_{u-v} + c_{u-2v}, \tag{1}$$

for $u, v \in Z$. It simplifies for $u = v$ to

$$c_{2u} = c_u^2 - 2c_u^q. \tag{2}$$

In [7], Lenstra and Verheul proved that computing $c_{u+v}$ and $c_{2u}$ take four and two multiplications in $\mathbb{F}_q$ respectively, when $c_u, c_v, c_{u-v}$, and $c_{u-2v}$ are given.

### 2.2 XTR Exponentiation

In XTR, an algorithm for computing $Tr_{(q^6,q^2)}(h^n)$ given $Tr_{(q^6,q^2)}(h)$ and a scalar $n \in Z$ is needed like the algorithm for computing $h^n$ in public key system based on discrete logarithm problem. By using two formula (1),(2) above, we define the following two functions called as XTR addition and XTR doubling respectively;

$$A[u, v, w, z] = u \cdot v - v^q \cdot w + z,$$
$$D[u] = u^2 - 2u^q.$$

**Theorem 1.** ([7], *Theorem* 2.3.8) *Let $c$ and a positive integer $n$ be given. Computing the sum $c_n$ of the $n^{th}$ powers of the roots takes $8\log_2(n)$ multiplications in $\mathbb{F}_q$.*

Thus, given the representation $Tr_{(q^6,q^2)}(h) \in \mathbb{F}_{q^2}$ of the conjugates of $h$, the representation $Tr_{(q^6,q^2)}(h^n) \in \mathbb{F}_{q^2}$ of the conjugates of the $n^{th}$ power of $h$ can be computed at the cost of $8\log_2(n)$ multiplications in $\mathbb{F}_q$, for any integer $n$.

Denote the above XTR exponentiation with input $c$ and $n$ outputs $c_n$ as

$$\mathsf{XTR\_Exp}[c, n] = c_n.$$

**XTR Exponentiation ([7], Algorithm 2.3.7)**

INPUT: $c$ and $n$ where $n > 2$
OUTPUT: $c_n$

1. Compute initial values:
   1.1. $C_3 \leftarrow c$, $C_0 \leftarrow D[C_3]$, $C_1 \leftarrow A[C_0, C_3, C_3, 3]$, and $C_2 \leftarrow D[C_0]$
   1.2. If $n$ is even, $n$ replace $n-1$.
   Let $n = 2m+1$ and $m = \sum_{j=0}^{l} m_j 2^j$ with $m_j \in \{0,1\}$ and $m_l = 1$.
2. for $j = l-1$ down to 0
   2.1. $T_1 \leftarrow D[C_{m_j}]$
   2.2. $T_2 \leftarrow D[C_{1+m_j}]$
   2.3. if $(m_j = 0)$ then $T_3 \leftarrow A[C_0, C_1, C_3^q, C_2^q]$
   if $(m_j = 1)$ then $T_3 \leftarrow A[C_2, C_1, C_3, C_0^q]$
   2.4. $C_0 \leftarrow T_1$
   2.5. $C_1 \leftarrow T_3$
   2.6. $C_2 \leftarrow T_2$
3. If $n$ is odd then return $C_1$
   else return $C_2$

### 2.3 XTR-DH Key Agreement

XTR can be used in any cryptosystem that relies on the discrete logarithm problem. This section contains a description of an application of XTR that provides confidentiality service, for example Diffie-Hellman key agreement.

**Public Parameters :** $q$, $p$, $c = Tr_{(q^6,q^2)}(h)$

If Alice and Bob want to agree on a secret key $K$ they do the following.

1. Alice selects at random $a \in Z_p$, uses $\mathsf{XTR\_Exp}[c, a] = c_a \in \mathbb{F}_{q^2}$, and sends $c_a$ to Bob.
2. Bob receives $c_a$ from Alice, selects at random $b \in Z_p$, uses $\mathsf{XTR\_Exp}[c, b] = c_b \in \mathbb{F}_{q^2}$, and sends $c_b$ to Alice.
3. Alice receives $c_b$ from Bob, computes $\mathsf{XTR\_Exp}[c_b, a] = c_{ba}$, and determines $K$ based on $c_{ba} := Tr_{(q^6,q^2)}(h^{ba})$.
4. Bob uses $\mathsf{XTR\_Exp}[c_a, b] = c_{ab}$, and determines $K$ based on $c_{ab} := Tr_{(q^6,q^2)}(h^{ab})$.

## 3 XTR over Characteristic Three

The original XTR uses the trace over $\mathbb{F}_{q^2}$ to represent elements of the order $q^2 - q + 1$ subgroup of $\mathbb{F}_{q^6}^*$, thereby achieving a factor 3 size reduction. This section shows that if $q$ is 3 to the odd power then elements in $G_{q^2-q+1}$ can be represented as elements in $\mathbb{F}_q$ using the trace over $\mathbb{F}_q$. It achieves a factor 6 size reduction, which is the half size reduction compared to the original XTR representation.

### 3.1 New XTR Group

We assume that $q = 3^t$ for any odd integer $t$, say $t = 2k-1$. Then, $\sqrt{3q} = 3^k$ is an integer and $q^2 - q + 1$ is factorized as

$$q^2 - q + 1 = (q + \sqrt{3q} + 1)(q - \sqrt{3q} + 1).$$

In this section, we define a new XTR group $G_p =< g >$ which is a subgroup of $G_{q-\sqrt{3q}+1}$, namely, XTR uses a subgroup of prime order $p$ of the order $q - \sqrt{3q} + 1$ subgroup of $\mathbb{F}_{q^6}^*$. The order $p$ subgroup $< g >$ generated by $g$ is referred as the *New XTR group*. Since $p$ does not divide any $q^s - 1$ for $s = 1, 2, 3$, the new XTR group $G_p$

generated by $g$ cannot be embedded in the multiplicative group of any true subfield of $\mathbb{F}_{q^6}$. Combined with the choice of $p$ it follows that computing discrete logarithms in $G_p$ is as hard, in general, as it is in $\mathbb{F}_{q^6}^*$ (cf. [7], Section 5).

$$G_p =< g > \quad \lhd \quad G_{q-\sqrt{3q}+1} \quad \lhd \quad G_{q^2-q+1} \quad \lhd \quad G_{q^3-1} \tag{3}$$

Here, $A \lhd B$ denotes $A$ is a subgroup of $B$.

From $q = 3^t$ and $t$ is odd it follows $q$ is a generator of $\mathbb{F}_5^*$, so that $\{\omega + \omega^{-1}, \omega^2 + \omega^{-2}\}$ form an Type-II ONB for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$, where $\omega$ is a root of the polynomial $(X^5 - 1)/(X - 1) = X^4 + X^3 + X^2 + X + 1$. For the simplicity, we denote $x = x_1 \cdot (\omega + \omega^{-1}) + x_2 \cdot (\omega^2 + \omega^{-2}) \in \mathbb{F}_{q^2}$ as $(x_1, x_2)$.

**Lemma 1.** *Let $x, y, z \in \mathbb{F}_{q^2}$ with $q = 3^t$ and $t$ is odd.*

*i. Computing $x^q$ is for free.*
*ii. Computing $x^2$ takes two multiplications in $\mathbb{F}_q$.*
*iii. Computing $x * z - y * z^q$ takes four multiplications in $\mathbb{F}_q$.*

*Proof.* Let $x = (x_1, x_2)$, $y = (y_1, y_2)$ and $z = (z_1, z_2) \in \mathbb{F}_{q^2}$ for $x_i,\ y_i,\ z_i \in \mathbb{F}_q$, $i \in \{1, 2\}$. From $(\omega + \omega^{-1})^q = \omega^2 + \omega^{-2}$ and $(\omega^2 + \omega^{-2})^q = \omega + \omega^{-1}$, $x^q = (x_2, x_1)$. It follows that $q^{th}$ powering in $\mathbb{F}_{q^2}$ does not require arithmetic operations and can thus be considered to be for free.

From $x^2 = \big((x_1+x_2)(x_1-x_2)+2x_1x_2, 2(x_1+x_2)(x_1-x_2)+2x_1x_2\big)$, $x^2$ is obtained from two multiplications in $\mathbb{F}_q$.

Finally, to compute $x * z - y * z^q$ four multiplications in $\mathbb{F}_q$ suffice, because it is easily verified that

$$\begin{aligned} x * z - y * z^q = &\big((x_2 - 2x_1 + y_2 - y_1) * z_1 + (x_1 - x_2 + 2y_1 - y_2) * z_2\big) * (\omega + \omega^{-1}) \\ &+ \big((x_2 - x_1 + 2y_2 - y_1) * z_1 + (x_1 - 2x_2 + y_1 - y_2) * z_2\big) * (\omega^2 + \omega^{-2}). \end{aligned}$$

□

### 3.2 Compression and Restoration

For some $q$ and $g$ of order $p$ dividing $q-\sqrt{3q}+1$, define $d$ and $e$ as the trace $Tr_{(q^6,q^2)}(g)$ over $\mathbb{F}_{q^2}$ and the trace $Tr_{(q^6,q)}(g)$ over $\mathbb{F}_q$, respectively. We use the shorthand $d_n = Tr_{(q^6,q^2)}(g^n)$ and $e_n = Tr_{(q^6,q)}(g^n)$, i.e. $e_n$ and $d_n$ are the sum of the conjugates over $\mathbb{F}_{q^2}$ and $\mathbb{F}_q$ of $g^n$ respectively. Immediately, $d = d_1$ and $e = e_1$.

$$\begin{aligned} d_n &= Tr_{(q^6,q^2)}(g^n) = g^n + g^{nq^2} + g^{nq^4} \in \mathbb{F}_{q^2} \\ e_n &= Tr_{(q^6,q)}(g^n) = g^n + g^{nq} + g^{nq^2} + g^{nq^3} + g^{nq^4} + g^{nq^5} \in \mathbb{F}_q. \end{aligned}$$

**Compression** From the definition of $d_n$ and $e_n$, $e_n$ can be easily derived from $d_n$ due to the following equation

$$e_n = d_n + d_n^q. \tag{4}$$

For any $d_n = x(\omega + \omega^{-1}) + y(\omega^2 + \omega^{-2}) \in \mathbb{F}_{q^2}$, we have that $e_n = (x + y) * (\omega + \omega^{-1}) + (x + y) * (\omega^2 + \omega^{-2}) \in \mathbb{F}_q$ because of Lemma 1-*i*. Note that as $d_n \in \mathbb{F}_{q^2}$ and $d_n \notin \mathbb{F}_q$, $x \neq y$.

Define a compression function with input an element of $\mathbb{F}_{q^2}$ represented by two elements of $\mathbb{F}_q$, say $(x, y)$, outputs an element of $\mathbb{F}_q$.

$$\mathsf{Compression}[x, y] = x + y$$

**Restoration** Contrary to the compression from $d_n$ to $e_n$, this section explains how to get $d_n$ from $e_n$, called it as restoration in this paper.

**Lemma 2.** *The roots of* $X^2 - e_n X + e_n^{\sqrt{3q}} \in \mathbb{F}_q[x]$ *are* $d_n$ *and* $d_n^q$.

*Proof.* It is sufficient to prove $d_n * d_n^q = e_n^{\sqrt{3q}}$ because $d_n + d_n^q = e_n$ from equation (4). For simplicity, we prove $d_n * d_n^q = e_n^{\sqrt{3q}}$ when $n = 1$.

$$\begin{aligned} d * d^q &= (g + g^{q^2} + g^{q^4}) \cdot (g^q + g^{q^3} + g^{q^5}) \\ &= g^{1+q} + g^{1+q^3} + g^{1+q^5} + g^{q^2+q} + g^{q^2+q^3} + g^{q^2+q^5} + g^{q^4+q} + g^{q^4+q^3} + g^{q^4+q^5} \\ &\overset{\checkmark}{=} g^{\sqrt{3q}} + (g^q)^{\sqrt{3q}} + (g^{q^2})^{\sqrt{3q}} + (g^{q^3})^{\sqrt{3q}} + (g^{q^4})^{\sqrt{3q}} + (g^{q^5})^{\sqrt{3q}} + 3 \\ &= (g+g^q+g^{q^2}+g^{q^3}+g^{q^4}+g^{q^5})^{\sqrt{3q}} \\ &= e^{\sqrt{3q}}. \end{aligned}$$

The third equality is derived from the series of subgroups in equation (3), that is to say, $g^{q+1} = g^{\sqrt{3q}}$ (from $<g> \lhd G_{q-\sqrt{3q}+1}$) and $g^{q^3+1} = 1$ (from $<g> \lhd G_{q^3-1}$). □

From Lemma 2, we can find two roots $d_n$ and $d_n^q$ by solving the quadratic formula, which are

$$\{d_n,\ d_n^q\} = \frac{e_n \pm \sqrt{e_n^2 - 4e_n^{\sqrt{3q}}}}{2}. \tag{5}$$

Let $e_n = z \in \mathbb{F}_q$ and the roots of the quadratic equation be $\{(x, y), (y, x)\}$, where $x, y \in \mathbb{F}_q$ and $x \neq y$. Actually, $z = x + y$. Define a restoration function with input $e_n$, outputs $\{d_n, d_n^q\} \in \mathbb{F}_{q^2}$.

$$\mathsf{Restoration}[e_n] = \{d_n, d_n^q\}.$$

## 4 Efficient Method of Restoration - finding $d_n$ and $d_n^q$ from $e_n$

As we have looked around at the previous section, we need to solve the quadratic formula described in Lemma 2 to extract $d_n$ and $d_n^q$ from $e_n$. In other words, we have to compute the square root extraction $\sqrt{e_n^2 - 4e_n^{\sqrt{3q}}}$.

In a finite field $\mathbb{F}_{r^s}$ where $r \equiv 3\ (mod\ 4)$ and odd $s$, the best algorithm known [3, 9] to compute a squire root executes $O(s \log_2 r)$ multiplications in $\mathbb{F}_{r^s}$. By that method, a solution of $X^2 = A$ is given by $X = A^{\frac{r^s+1}{4}}$, assume that $A$ is a quadratic residue. Recently, Barreto et al. (c.f. [1], Section 4) presented an improvement to it. The complexity is reduced to $O(\log_2 s + \log_2 r)$ multiplications in $\mathbb{F}_{r^s}$. If the characteristic $r$ is fixed and small compared to $s$, the complexity is simply $O(\log_2 s)$.

### 4.1 Square Root Extraction

Let $R = e_n^2 - 4e_n^{\sqrt{3q}} \in \mathbb{F}_q$. Here, $q = 3^{2k-1}$ for any inter $k$. As $d_n$ or $d_n^q \notin \mathbb{F}_q$ $\sqrt{R}$ is not an element of $\mathbb{F}_q$. Thus, we can not utilize Barreto et al.'s method directly to compute square root of $R$ even if $q \equiv 3\ (mod\ 4)$.

**Fact 2** $-1$ *has a square root in* $\mathbb{F}_q$ *if and only if* $q \equiv 1\ (mod\ 4)$.

As $q \equiv 3\ (mod\ 4)$, $\sqrt{-1} \notin \mathbb{F}_q$, but in $\mathbb{F}_{q^2}$.

**Lemma 3.** *$\sqrt{-R} \in \mathbb{F}_q$, where $-R = 2e_n^2 + e_n^{\sqrt{3q}}$.*

*Proof.* Let $\mathbb{F}_q^* = \langle g_1 \rangle$. $\sqrt{g_1^n} = g_1^{n/2} \in \mathbb{F}_q$ if $n$ is even and $\sqrt{g_1^n}$ is not in $\mathbb{F}_q$ if $n$ is odd for $g_1^n \in \mathbb{F}_q^*$. From $(g_1^{(q-1)/2})^2 = 1$ and $g_1$ is a generator of $\mathbb{F}_q$, $g_1^{(q-1)/2} = -1$. We confirm easily that $(q-1)/2$ is odd if $q = 3^{2k-1}$. Then we see that $R = g_1^{n_1}$ for some odd $n_1$ since $\sqrt{R} \notin \mathbb{F}_q$. Hence $-R = R \cdot (-1) = g_1^{n_1+(q-1)/2}$ and $n_1 + (q-1)/2$ is even. Therefore, $\sqrt{-R} \in \mathbb{F}_q$. □

From Lemma 3, one of $\sqrt{-R}$ is $(-R)^{\frac{q+1}{4}}$ and it is efficiently computed by using the idea of Barreto et al. [1]. The basic idea is as follows.

They noticed that, if $q = 3^{2k-1}$ for some $k$:

$$\frac{q+1}{4} = \frac{3^{2k-1}+1}{4} = 6 \cdot \sum_{i=0}^{k-2} (3^2)^i + 1,$$

so that

$$\big(-R\big)^{(q+1)/4} = \big[\big((-R)^2\big)^{\sum_{i=0}^{k-2}(3^2)^i}\big]^3 \cdot (-R).$$

The quantity $\big((-R)^2\big)^{\sum_{i=0}^{k-2}(3^2)^i}$ is efficiently computed in an analogues fashion to Itoh-Teechai-Tsujii inversion [6], based on the Frobenius map in characteristic three. Let $A \in \mathbb{F}_q$. Then, one can compute $A^{\sum_{i=0}^{k-2}(3^2)^i}$ with no more than $\lfloor \log_2(k-1) \rfloor + HW(k-1) - 1$ multiplications in $\mathbb{F}_q$. Here, $\lfloor \cdot \rfloor$and $HW(\cdot)$ denote the maximum integer less than its operand and the Hamming weight of its operand respectively. Thus, we need at most $\lfloor \log_2(k-1) \rfloor + HW(k-1) + 1$ multiplications in $\mathbb{F}_q$ to compute $(-R)^{(q+1)/4}$ in total.

Next we must find $\sqrt{-1} \in \mathbb{F}_{q^2}$ to compute $\sqrt{R} = \sqrt{-R} \cdot \sqrt{-1}$. 4-th roots of unity in $\mathbb{F}_{q^2}$ are $\pm 1$, $\pm\sqrt{-1}$. We select an element $x$ in $\mathbb{F}_{q^2}^*$ at random then $x^{(q^2-1)/4}$ becomes any of $\pm 1, \pm\sqrt{-1}$. If $x^{(q^2-1)/4}$ is not $\pm 1$ then it is one of square root of $-1$. 4-th roots of unity are also efficiently computed just by small modification of Barreto et al.'s idea [1].

$$\frac{q^2-1}{4} = \frac{3^{4k-2}-1}{4} = 2 \cdot \sum_{i=0}^{2k-2} (3^2)^i,$$

so that

$$\big(x\big)^{(q^2-1)/4} = \big(x^2\big)^{\sum_{i=0}^{2k-2}(3^2)^i}.$$

As $x \in \mathbb{F}_{q^2}$, to find a 4-th root of unity it takes on average $2 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$ multiplications in $\mathbb{F}_{q^2}$ using the Frobenius map in characteristic three. Thus, $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$ multiplications in $\mathbb{F}_q$ as one multiplication in $\mathbb{F}_{q^2}$ takes three[1] multiplications in $\mathbb{F}_q$. Note that if $q$ is fixed, then finding square root of $-1$ can be pre-computed.

[1] Multiplication in $\mathbb{F}_{q^2}$ can be done using four multiplications in $\mathbb{F}_q$. These straightforward results can simply be improved to three multiplications by using a Karatsuba-like approach: to compute $(x_1, x_2) * (y_1, y_2)$ one computes $x_1 * y_1$, $x_2 * y_2$, and $(x_1 + x_2) * (y_1 + y_2)$, then $(x_1, x_2) * (y_1, y_2)$ becomes $\big((x_1 + x_2) * (y_1 + y_2) + x_2 * y_2, (x_1 + x_2) * (y_1 + y_2) + x_1 * y_1\big)$.

### 4.2 Computation of $d_n$ and $d_n^q$

Thanks to the equation (5) and the results of the previous section, for given $e_n \in \mathbb{F}_q$

$$\begin{aligned}\{d_n, d_n^q\} &= \frac{e_n \pm \sqrt{R}}{2} \\ &= 2 \cdot (e_n \pm \sqrt{-R} \cdot \sqrt{-1}) \\ &= 2e_n \pm (2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}. \end{aligned} \tag{6}$$

Table 1 shows the number of multiplications in $\mathbb{F}_q$ required to compute equation (6), where as customary we do not count the cost of additions and subtractions in $\mathbb{F}_q$.

**Table 1.** The number of multiplications in $\mathbb{F}_q$ for computation of $d_n$ and $d_n^q$, where $q = 3^t$ and $t = 2k - 1$ for some integer $k$

| Operation | # of multiplications in $\mathbb{F}_q$ |
|---|---|
| $e_n^2$ | 1 |
| $e_n^{\sqrt{3q}}$ | free if $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$ |
| $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}}$ | $\lfloor \log_2(k-1) \rfloor + HW(k-1) + 1$ |
| $\sqrt{-1}$ | $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big)$ |
| $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}$ | 2 |
| $2e_n \pm (2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \cdot \sqrt{-1}$ | $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) +$ $\lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ |

For efficient computation of $\sqrt{3q}$-th power of $e_n$ ($\in \mathbb{F}_q$), i.e. $e_n^{\sqrt{3q}}$, we should select $q$ such that $\mathbb{F}_q$ has optimal normal basis (ONB) over $\mathbb{F}_3$. As $q = 3^{2k-1}$, $\sqrt{3q} = 3^k$. Thus, $\sqrt{3q}$-th power is performed by shift of coefficients when $\mathbb{F}_q$ has ONB over $\mathbb{F}_3$. However, $\mathbb{F}_q$ never has Type-I ONB over $\mathbb{F}_3$ since $2k$ is not prime. Therefore, we should check whether $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$ or not for given $k$. For example, we may select $k = 55, 70, 82, 89$, and 94, which satisfy that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$.

Note that a multiplication $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} * \sqrt{-1}$ takes two multiplications in $\mathbb{F}_q$ because $(2e_n^2 + e_n^{\sqrt{3q}})^{\frac{q+1}{4}} \in \mathbb{F}_q$ and $\sqrt{-1} \in \mathbb{F}_{q^2}$.

**Theorem 2.** *Given $e_n$ for any integer $n$, computing $d_n$ and $d_n^q$ take about $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ multiplications in $\mathbb{F}_q$ under assumption that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$.*

## 5 Compressed XTR Exponentiation

In this section it is shown how $e_n$ can be computed based on $e_1$ and an arbitrary integer $n$.

Restoration - compute $d_1$ and $d_1^q$ from Restoration$[e_1]$. Between $\{d_1, d_1^q\}$ choose one of them at random, denoted $d'$.
XTR exponentiation - compute $d'_n$ from XTR_Exp$[d', n]$ described in section 2.2.

Compression - compute $\mathsf{Compression}[d'_n] = d'_n + (d'_n)^q$. Actually, $\mathsf{Compression}[d'_n] = e_n$.

At the compression step, we can easily check $d'_n + (d'_n)^q = e_n$. $d'$ is one of $\{d_1, d_1^q\}$. If $d' = d_1$ then it is trivial because of the definition of $e_n$. Otherwise, i.e. $d' = d_1^q$ then $d'_n + (d'_n)^q = d_n^q + d_n$ because $d_n \in \mathbb{F}_{q^2}$, which concludes the justification of the compression step.

Denote the above XTR exponentiation over characteristic three with input $e_1$ and $n$ outputs $e_n$ as

$$\mathsf{XTR\_Exp}_3[e_1, n] = e_n.$$

**Theorem 3.** *Let $e_1$ and a positive integer $n \in Z_p$ be given. Assume that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$. Then, computing $e_n$ takes about $8\log_2(n) + 6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ multiplications in $\mathbb{F}_q$.*

*Proof.* Immediate from Theorem 2, XTR Exponentiation algorithm ([7], Algorithm 2.3.7), and Lemma 1.

### 5.1 Application to XTR-DH

In this section we describe XTR version Diffie-Hellman key agreement over characteristic three.

**Public Parameters :** $q(= 3^{2k-1})$, $p$, $Tr_{(q^6,q)}(g) := e$

Suppose that Alice and Bob who both have access to the XTR public key data, want to agree on a shared secret key $K$. This can be done using the following XTR version.

1. Alice selects at random $a \in Z_p$, uses $\mathsf{XTR\_Exp}_3[e, a] = e_a \in \mathbb{F}_q$, and sends $e_a$ to Bob.
2. Bob receives $e_a$ from Alice, selects at random $b \in Z_p$, uses $\mathsf{XTR\_Exp}_3[e, b] = e_b \in \mathbb{F}_q$, and sends $e_b$ to Alice.
3. Alice receives $e_b$ from Bob, computes $\mathsf{XTR\_Exp}_3[e_b, a] = e_{ba}$, and determines $K$ based on $e_{ba} = Tr_{(q^6,q)}(g^{ba})$.
4. Bob uses $\mathsf{XTR\_Exp}_3[e_a, b] = e_{ab}$, and determines $K$ based on $e_{ab} = Tr_{(q^6,q)}(g^{ab})$.

### 5.2 Comparison to Original XTR

In this section, we compare XTR over characteristic three to the original XTR. Let $\mathbb{XTR}$ and $\mathbb{XTR}_3$ denote the original XTR [7] and XTR over characteristic three respectively.

#### XTR group $G_p$

$\mathbb{XTR}$ - XTR group $G_p = <h>$ is a subgroup of $G_{q^2-q+1}$, where $h \in \mathbb{F}_{q^6}^*$.

- $p$ and $q$ are prime, and $q \equiv 2(\ mod\ 3)$.

$\mathbb{XTR}_3$ - XTR group $G_p = <g>$ is a subgroup of $G_{q-\sqrt{3q}+1}$, where $g \in \mathbb{F}_{q^6}^*$.

- $p$ is prime and $q = 3^{2k-1}$.

Note that suggested lengths to provide adequate levels of security are $\log_2(q) \approx 170$ and $\log_2(p) \approx 160$.

**XTR Exponentiation**

$\mathbb{XTR}$ - For given $Tr_{(q^6,q^2)}(h)$ and $n \in Z_p$ computing $Tr_{(q^6,q^2)}(h^n)$ takes $8\log_2(n)$ multiplications in $\mathbb{F}_q$.
- $\mathbb{F}_{q^2}$ has Type-I ONB over $\mathbb{F}_q$.

$\mathbb{XTR}_3$ - For given $Tr_{(q^6,q)}(g)$ and $n \in Z_p$ computing $Tr_{(q^6,q)}(g^n)$ takes $8\log_2(n) + 6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ multiplications in $\mathbb{F}_q$.
- $\mathbb{F}_{q^2}$ has Type-II ONB over $\mathbb{F}_q$.
- $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$.

Denote by $P$ and $Q$ the sizes of the prime $p$ and $q$ to be generated, respectively. To achieve security at least equivalent to 1024-bit RSA, $6Q$ should be set to about 1024, i.e., $Q \approx 170$, and $P$ can for instance be set at 160. In $\mathbb{XTR}_3$, $k = 55$ satisfies that $\mathbb{F}_q$ has Type-II ONB over $\mathbb{F}_3$, and also the size of $Q$ ,where $k = 55$, is about 170. In the case when $k = 55$, the result of $6 \cdot \big(\lfloor \log_2(2k-1) \rfloor + HW(2k-1)\big) + \lfloor \log_2(k-1) \rfloor + HW(k-1) + 4$ is at most 101. In general, the size of $n$ is about 160-bit. Under these conditions, $Tr_{(q^6,q)}(g^n)$ takes about 1359 multiplications in $\mathbb{F}_q$, which is only about 6% increase compared to the cost of computation of $Tr_{(q^6,q^2)}(h^n)$. If $q$ is fixed, then finding square root of $-1$ can be pre-computed. Then, the computational overhead is only 1% compared to that of original XTR.

**Communication Overhead** The communication overhead of XTR-DH in $\mathbb{XTR}_3$ is about *half* of XTR-DH proposed in [7] and *one six* of traditional implementations of the Diffie-Hellman protocol that are based on subgroups of multiplicative groups of finite fields, and that achieves the same level of security.

**Size of Public Key Parameter** In $\mathbb{XTR}$, the public key data are $q$, $p$, and $Tr_{(q^6,q^2)}(h)$. Thus, the total length is $3Q + P$. However, in the case of $\mathbb{XTR}_3$, the public key data are $q(= 3^{2k-1})$, $p$, and $Tr_{(q^6,q)}(g)$, and the total length of it is $2Q+P$. If we select $Q \approx 170$ and $P \approx 160$ then the required length of public key data of $\mathbb{XTR}_3$ is reduced about 26% from that of $\mathbb{XTR}$.

## 6 Side Channel Attacks

### 6.1 Improved XTR Single Exponentiation

(cf. [11], Algorithm 5.1)

XTR-ISE is based on an adaptation of a Euclidean algorithm by Montgomery using Lucas chains. For ease of notation, we will momentarily use ordinary exponentiation in our description instead of the third order XTR recurrence. Note that the following description of XTR-ISE is based on Algorithm 3.1 without the optional steps and Algorithm 5.1 in [11].

**Improved XTR Single Exponentiation (XTR-ISE)**

Input: $c_1$ and $n$ where $n > 2$
Output: $c_n$

1. Initialization:
   1.1. Let $a$ =round($\frac{3-\sqrt{5}}{2}n$) and $b = n - a$ (where round($x$) is the integer closest to $x$).
   1.2. Let $f = 0$. As long as $a$ and $b$ are both even, replace $(a, b)$ by $(a/2, b/2)$ and $f$ by $f + 1$.
   1.3. Let $i = 1$ and $G_i := (Q_0, Q_1, Q_2, Q_3) = (c_1, c_1, 3, c_1^p)$.

2. As long as $a \neq b$
   2.1. If $b > a$
   $X_1$. if $b \leq 4a$, then $(a, b) \leftarrow (b - a, a)$
   $T_0 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$, $T_1 \leftarrow Q_0$,
   $T_2 \leftarrow Q_1$, $T_3 \leftarrow Q_2^p$.
   $X_2$. else if $b$ is even, then $(a, b) \leftarrow (a, b/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
   $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$.
   $X_3$. else if $a$ is odd, then $(a, b) \leftarrow (a, (b - a)/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
   $T_2 \leftarrow Q_2$, $T_3 \leftarrow D[Q_1]^p$.
   $X_4$. else ($a$ is even), then $(a, b) \leftarrow (b, a/2)$
   $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow Q_0$,
   $T_2 \leftarrow Q_3^p$, $T_3 \leftarrow D[Q_2]^p$.
   2.2. Else (if $a > b$)
   $Y_1$. if $a \leq 4b$, then $(a, b) \leftarrow (a - b, b)$
   $T_0 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$, $T_1 \leftarrow Q_1$,
   $T_2 \leftarrow Q_0$, $T_3 \leftarrow Q_2$.
   $Y_2$. else if $a$ is even, then $(a, b) \leftarrow (b, a/2)$
   $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow Q_0$,
   $T_2 \leftarrow Q_3^p$, $T_3 \leftarrow D[Q_2]^p$.
   $Y_3$. else if $b$ is odd, then $(a, b) \leftarrow (b, (a - b)/2)$
   $T_0 \leftarrow D[Q_1]$, $T_1 \leftarrow A[Q_0, Q_1, Q_2, Q_3]$,
   $T_2 \leftarrow Q_2^p$, $T_3 \leftarrow D[Q_0]^p$.
   $Y_4$. else ($b$ is even), then $(a, b) \leftarrow (a, b/2)$
   $T_0 \leftarrow D[Q_0]$, $T_1 \leftarrow Q_1$,
   $T_2 \leftarrow A[Q_0, Q_2, Q_1, Q_3^p]$, $T_3 \leftarrow D[Q_2]$
   2.3. $i \leftarrow i + 1$ and set $G_i = (T_0, T_1, T_2, T_3)$.
3. Compute $\tilde{c} = A[Q_0, Q_1, Q_2, Q_3] = c_{u+v}$.
4. Output $\tilde{c}_{2^f}$.
5. If $a = 1$ then return $\tilde{c}_{2^f}$
   else run Improved XTR Single Exponentiation with $c = \tilde{c}_{2^f}$ and $n = a$.

---

### 6.2 SPA-Resistant Trace Operations

From Lemma 1,

$$\begin{aligned} D[x] &= x^2 - 2x^q = (R_1, R_2), \\ R_1 &= (x_1 + x_2) \cdot (x_1 - x_2) + 2x_1x_2 - 2x_2, \\ R_2 &= 2(x_1 + x_2) \cdot (x_1 - x_2) + 2x_1x_2 - 2x_1. \end{aligned}$$

$$\begin{aligned} A[x, z, y, w] &= x \cdot z - y \cdot z^q + w = (S_1, S_2), \\ S_1 &= (x_2 + x_1 + y_2 - y_1) \cdot z_1 + (x_1 - x_2 - y_1 - y_2) \cdot z_2 + w_1, \\ S_2 &= (x_2 - x_1 - y_1 - y_2) \cdot z_1 + (x_1 + x_2 + y_1 - y_2) \cdot z_2 + w_2. \end{aligned}$$

In Table 2 we show how to make two applications of $D[\cdot]$ indistinguishable from one application of $A[\cdot]$. We may assume that modular addition and modular subtraction are indistinguishable if we represent an element $a \in \mathbb{F}_3$ as two bit-vectors $a_H$ and $a_L$ constructed from $a$ such that $a_H = a$ div 2 and $a_L = a$ mod 2. Given representation of this type, subtraction can be implemented by the same function as addition since the negation of an element $a$ simply swaps the vectors $a_H$ and $a_L$ over.

## References


1. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *Advances in Cryptology-CRYPTO 2002*, LNCS 2442, Springer, pp.354-369, 2002.

**Table 2.** Indistinguishable arithmetic

| | One $A[\cdot]$ call | Value | Operation | | Two $D[\cdot]$ calls | value |
|---|---|---|---|---|---|---|
| $t_1$ | $x_1 + x_2$ | $x_1 + x_2$ | Add | Add | $x_1 + x_2$ | $x_1 + x_2$ |
| $t_2$ | $y_2 - y_1$ | $y_2 - y_1$ | Sub | Sub | $x_1 - x_2$ | $x_1 - x_2$ |
| $t_3$ | $t_1 - t_2$ | $(x_1 + x_2 + y_1 - y_2)$ | Sub | Sub | $t_1 - t_2$ | $2x_2$ |
| $t_4$ | $t_3 \cdot z_2$ | $(x_1 + x_2 + y_1 - y_2) \cdot z_2$ | Mul | Mul | $t_1 \cdot t_2$ | $(x_1 + x_2) \cdot (x_1 - x_2)$ |
| $t_5$ | $t_3 - t_2$ | $(x_1 + x_2 - y_1 + y_2)$ | Sub | Sub | $t_4 - t_3$ | $(x_1 + x_2) \cdot (x_1 - x_2) - 2x_2$ |
| $t_6$ | $t_5 \cdot z_1$ | $(x_1 + x_2 - y_1 + y_2) \cdot z_1$ | Mul | Mul | $x_1 \cdot x_2$ | $x_1 x_2$ |
| $t_7$ | $y_1 - y_2$ | $y_1 - y_2$ | Sub | Sub | $t_5 - t_6$ | $R_1$ |
| $t_8$ | $t_7 - y_2$ | $y_1 + y_2$ | Sub | Sub | $x_1 - t_6$ | $x_1 + 2x_1 x_2$ |
| $t_9$ | $x_1 - x_2$ | $x_1 - x_2$ | Sub | Sub | $t_8 - t_4$ | $R_2$ |
| $t_{10}$ | $t_9 - t_8$ | $(x_1 - x_2 - y_1 - y_2)$ | Sub | Add | $x_1 + x_2$ | $x_1 + x_2$ |
| $t_{11}$ | $x_2 - x_1$ | $x_2 - x_1$ | Sub | Sub | $x_1 - x_2$ | $x_1 - x_2$ |
| $t_{12}$ | $t_{11} - t_8$ | $(x_2 - x_1 - y_1 - y_2)$ | Sub | Sub | $t_{10} - t_{11}$ | $2x_2$ |
| $t_{13}$ | $t_{10} \cdot z_2$ | $(x_1 - x_2 - y_1 - y_2) \cdot z_2$ | Mul | Mul | $t_{10} \cdot t_{11}$ | $(x_1 + x_2) \cdot (x_1 - x_2)$ |
| $t_{14}$ | $t_4 + w_2$ | $(x_1 + x_2 + y_1 - y_2) \cdot z_2 + w_2$ | Add | Sub | $t_{13} - t_{12}$ | $(x_1 + x_2) \cdot (x_1 - x_2) - 2x_2$ |
| $t_{15}$ | $t_{12} \cdot z_1$ | $(x_2 - x_1 - y_1 - y_2) \cdot z_1$ | Mul | Mul | $x_1 \cdot x_2$ | $x_1 x_2$ |
| $t_{16}$ | $t_6 + w_1$ | $(x_1 + x_2 - y_1 + y_2) \cdot z_1 + w_1$ | Add | Sub | $t_{14} - t_{15}$ | $R_1$ |
| $t_{17}$ | $t_{13} + t_{16}$ | $S_1$ | Add | Sub | $x_1 - t_{15}$ | $x_1 + 2x_1 x_2$ |
| $t_{18}$ | $t_{14} + t_{15}$ | $S_2$ | Add | Sub | $t_{17} - t_{13}$ | $R_2$ |


2. A.E. Brouwer, R. Pellikaan, E.R. Verheul, "Doing more with fewer bits," *Asiacrytp 1999*, LNCS 1716, pp.321-332, 1999.
3. H. Cohen, "A Course in Computational Algebraic Number Theory," Springer, 1993.
4. T.ElGamal, "A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, 31(4), pp.469-472, 1985.
5. G.Gong, L.Harn, "Public key cryptosystems based on cubic finite field extensions," *IEEE Transactions on Information Theory*, 45 (7), pp.2601-2605, 1999.
6. T. Itoh, O. Teechai and S. Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases," *Information and Computation*, 78, pp.171-177, 1988.
7. A. K. Lenstra and E. R. Verheul, "The XTR Public Key System," *Advances in Cryptology-CRYPTO 2000*, LNCS 1800, Springer, pp.1-20, 2000.
8. A. K. Lenstra , "Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields," *ACISP 1997*, LNCS 1270, Springer, pp.127-138, 1997.
9. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1997.
10. C.P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, 4, pp.161-174, 1991.
11. M. Stam and A.K. Lenstra, *Speeding Up XTR*, Proceedings of Asiacrypt 2001, LNCS 2248, (2001), 125-143.
12. P.Smith, C.Skinner, "A Public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms," *Asiacrytp 1994*, LNCS 917, pp.357-364, 1995.