Comparative Analysis of Calculations in Cryptographic Protocols Using a Combination of Different Bases of Finite Fields

Sergey Gashkov[1(✉)] and Alexander Frolov[2]

[1] Faculty of Mechanics and Mathematics, Lomonosov Moscow State University - MSU, GSP-1, 1 Leninskiye Gory, Main Building, Moscow 119991, Russia
sbgashkov@gmail.com

[2] National Research University Moscow Power Engineering Institute, Krasnokazarmennaya, 14, Moscow 111250, Russian Federation
abfrolov@gmail.com

Abstract. The chapter introduces a comparative analysis of the complexity of the Tate pairing operation on a supersingular elliptic curve and the complexity of the final exponentiation in the tripartite key agreement cryptographic protocol. The analysis takes into account a possibility of using different bases of finite fields in combination. Operations of multiplication and multiple squaring in the field $GF(2^n)$ and its 4-degree extension, of Tate pairing on supersingular elliptic curve and of final exponentiation are considered separately and in combination. We conclude that the best complexity bound for the pairing and the final exponentiation in the cryptographically significant field $GF(2^{191})$ is provided by the combination of the polynomial basis of this field and 1-type optimal basis of the field expansion.

Keywords: Finite field · Extension of finite field · Optimal normal basis · Combination of bases · Supersingular elliptic curve · Tate pairing · Algorithm with square root extraction · Algorithm without square root extraction · Final exponentiation

1 Introduction

The idea of combining bases to accelerate computations in finite fields was first introduced in [1] on the basis of the estimates of complexity of transformations of bases in the fields possessing the 2- or 3-type optimal normal basis (o.n.b.) [2]. In [3–5], a number of modifications of the multiplication in these bases have been proposed. In particular, in [5] multiplication algorithm in the so-called optimal polynomial basis of type 2 (in the terminology of [1] - almost polynomial basis (a.p.b)) using the multiplication operations in the ring $GF(2)[X]$ is described and estimated. The product is converted into a.p.b. using a permutated o.n.b., i.e. by means of operations without reduction modulo an irreducible polynomial.

Recall that 1-type o.n.b. in the field $GF(2^n)$ occurs if $p = n+1$ is a prime number, and 2 is a primitive root mod p, 2-type o.n.b. or 3-type o.n.b. arise when $p = 2n + 1$ is a prime number, and the field characteristic 2 is a primitive root modulo p. If $p \equiv 3$ (mod 4), and 2 is a quadratic residue, we have 3-type o.n.b, otherwise 2-type o.n.b.

As used in this paper, the field $GF(2^{191})$, has 3-type o.n.b. and its 4-th degree extension has 1-type o.n.b.. Availability o.n.b. allowing to speed up the operation of raising to a power equal to the power of the field characteristics, or the scalar multiplication of the elliptic curve point by the power of the field characteristics, as well as the operation of the square root extracting.

Polynomial basis p.b. of these fields has generators, which are the roots of the irreducible trinomials that simplifies the implementation of the operations of multiplication in these bases. Thus, there is reason to explore the possibility of sharing o.n.b. and p.b. in the implementation of the various stages of cryptographic protocols.

In this paper, we concretize the idea of using combinations of bases in relation to the implementation of the tripartite key agreement protocol [6] in arithmetic supersingular elliptic curve over a cryptographically significant field $GF(2^{191})$: tacking into account security parameter $k = 4$ for supersingular elliptic curve over this field, security of discrete logarithm problem in group of elliptic curve points is equivalent to security of this problem in multiplicative group of order $2^{764} - 1$ [7]. Recall that this protocol is one round. System parameter is a point P of supersingular elliptic curve over the ground field $GF(2^n)$.

Each of the three participants A, B and C selects a private key a, b and c, computes and publishes the public key $KA = aP$, $KB = bP$ and $KC = cP$. Then each of them receives the public keys of other participants, calculates an element $e(bP,cP)$, $e(aP, cP)$ or $e(bP, aP)$ of the field $GF(2^{n\times4})$ performing the Tate pairing operation e with two points of an elliptic curve and then the operation of the final exponentiation (raising to a power equal to the quotient of the order of the group $GF(2^{n\times4})^*$ on the order of the elliptic curve). The final step is to calculate the shared secret key by exponentiation of the result to the power a, b and c respectively. The chapter provides upper bounds on the number of elementary operations in the pairing and the final exponentiation phases of the said cryptographic protocol. The rest part of chapter contains Sect. 2 were operation of multiplication and multi squaring in distinct bases of the field $GF(2^{191})$ are considered, Sect. 3 that is devoted to these operations in 4-degree extension of this field comparing their complexity for distinct combinations of bases of basic field and its extension. In Sect. 4 we compare complexity of pairing and final exponentiation operations separately and totally for distinct combinations of bases. In conclusion the comparison results are summarized.

2 Bases and Operations in $GF(2^n)$

Consider a sequence $\beta_i = \alpha^i + \alpha^{-i} \in GF(2^n)$, $n = 191, i \in Z$. The set $\{1, \beta_1, \ldots, \beta_1^i, \ldots, \beta_1^{n-1}\}$ is called a polynomial base (p.b) of $GF(2^n)$, the set $\{\beta_1, \ldots, \beta_1^i, \ldots, \beta_1^n\}$ $(\{\beta_1, \ldots, \beta_1^i, \ldots, \beta_1^{2n}\})$ forms an almost p.b (a.p.b) of $GF(2^n)$ (doubled a.p.b). The set $\{\xi_1, \ldots, \xi_i, \ldots, \xi_n\} = \left\{\beta_1^{2^0}, \ldots, \beta_1^{2^{i-1}}, \ldots, \beta_1^{2^{n-1}}\right\}$ is an optimal normal base (o.n.b) of $GF(2^n)$. The set $\{\beta_1,\ldots,\beta_i,\ldots,\beta_n\}$ (or $\{\beta_1,\ldots,\beta_i,\ldots,\beta_{2n}\}$), $\beta_i = \xi_{\pi(i)}, i = 1,\ldots n$, where π is a permutation

$$\pi(i) = \begin{cases} 2^i \bmod p \text{ if } 2^i \bmod p \le n, \\ (p-2^i) \bmod p \text{ if } 2^i \bmod p > n, \end{cases}$$

$p = 2n+1$, is called a permuted o.n.b. (p.o.n.b) (or doubled p.o.n.b).

Let T (T^{-1}) denote the operation of the conversion from a.p.b to p.o.n.b (from p.o.n.b to a.p.b). If $2^k < n < 2^{k+1}$, then the conversion complexity (number of xor-operations) satisfies the recurrent inequality $L_T(n) \le L_T(n-2^k) + L_T(2^k) + n - 2^k$ with the initial value $L_T(2) = 0$. This recurrence can be solved as $L_T(2^k) \le 2^{k-1}(k-2)+1$ [5]. Note, that the weaker bound $L_T(2^k) \le 2^{k-1}(k)+1$ was derived in [1] due to the overestimated initial value $L_T(2) = 2$. From this inequality one can obtain estimations $L_T(191) = 513$, $L_T(382) = 1227$. Trivially, the complexity of the operation D of the conversion from d. p.o.n.b to p.o.n.b. is n-1.

Following [5], we multiply elements of $GF(2^n)$ represented in a.p.b as elements of the ring $GF(2)[X]$ getting the product in d.a.p.b. Denote this operation $\times_R$. Also following [5] we denote *Bottom*(**a**) and *Top*(**a**) the lower half of product and product after replacing of its lower half with zeros). We implement two multiplication operations in a.p.b:

$\mathbf{y} \times_{apbP} \mathbf{z}$ with result in a.p.b,

$\mathbf{y} \times_{apbN} \mathbf{z}$ with result in p.o.n.b,:

$\mathbf{y} \times_{apbP} \mathbf{z} = Bottom(\mathbf{c}) + T^{-1}(D(T(Top(c))))$,

$\mathbf{y} \times_{apbN} \mathbf{z} = T(Bottom(\mathbf{c})) + D(T(Top(c)))$,

where $\mathbf{c} = \mathbf{y} \times_R \mathbf{z}$, «+» is n-bit xor.

It follows that complexity of each of these operations (number of logical operations) $L(\times_{apbP}) = L(\times_{apbN}) = M(n) + L_T(2n) + 2n$ where $M(n)$ is complexity of operation $\times_R$ (transformation D in this case is performed without "xor" - operations).

Implementing $\mathbf{c} = T(\mathbf{y}) \times_R T(\mathbf{z})$ instead of $\mathbf{c} = \mathbf{y} \times_R z$ we obtain also two multiplication operations in p.o.n.b:

$\mathbf{y} \times_{ponbP} \mathbf{z}$ with result in a.p.b,

$\mathbf{y} \times_{ponbN} \mathbf{z}$ with result in p.o.n.b.

Complexity of each of these operations $L(\times_{ponbP}) = L(\times_{ponbN}) = M(n) + 2 * L_T(n) + L_T(2n) + n$.

Denote $\times_{pbN}$ multiplication in p.b. in the field $GF(2^n)$ with minimal polynomial $P_{\text{onb}}(X)$ that root generates o.n.b. It can be performed converting operands to a.p.b., executing $\times_{apbP}$ and converting the product back to p.b. Mentioned converting's are of complexity n. Hence complexity of multiplication $\times_{pbN}$ is $L(\times_{apbP}) + 3n$.

P.b. is organized using the root of trinomial $P_{\text{pb}}(X)$ instead $P_{\text{onb}}(X)$. Let $R(\mathbf{x})$ be the modulo trinomial reduction of complexity $2n$. Then multiplication in p.b. of $GF(2^n)$ is denoted and described as $\mathbf{y} \times_R \mathbf{z} = R(\mathbf{x} \times_R \mathbf{y})$. Its complexity is limited to $M(n) + 2n$. Squaring in p.b. is performed by an algorithm that directly take into account the reducing of the vector-result. Below we denote this operation $z_{pb}^{(1)} = \boldsymbol{Z}^2$. Its complexity is limited to n, but symbolically synthesis for n = 191 gave the squaring program with only 99 xor's.

Denote $z_{ponb}^{(j)} = z^{2^j}$ a raising to power 2^j operation implemented to element $\mathbf{z}$ in p.o.n.b. with result in a.p.b.:

```
for $i = (1, n)$:
$\mathbf{b}[i] = \mathbf{a}[\pi[\pi^{-1}(i) - \mathrm{j}]]$
$\mathbf{c} = T(\mathbf{b})$.
```

Its complexity equals 0, because logical operations absent.

Operation $z_{apb}^{(j)} = T^{-1}(T(z)^{2^j})$ is implemented to element $\mathbf{z}$ in a.p.b. with result in a.p.b. Its complexity is $2L_T$ (n).

Operation $z_{apbN}^{(j)} = T(z)^{2^j}$ is implemented to element $\mathbf{z}$ in a.p.b. with result in p.o.n.b. Its complexity is L_T.

Operation $z_{pbN}^{(j)} = T(z)^{2^j}$ is implemented to element $\mathbf{z}$ in p.b. (for minimal polynomial $P_{\text{onb}}(X)$) with result in p.b. Its complexity is $2L_T(n) + n)$.

Denote $z_{pb}^{(j)}$ a raising to power 2^j operation to element $\mathbf{z}$ in p. b. (for minimal polynomial $P_{\text{pb}}(X)$) with result in p.b.:

```
$\mathbf{c} = \mathbf{z}$
for $i = (1,j)$:
        $\mathbf{c} = \mathbf{c}_{pb}^{(1)}$
```

Its complexity is bounded by nj (for n = 191 one can take estimate $99j$).

In Table 1 there are represented the numbers of logical operations "xor" and "and" (denoted $\oplus$ and &) and the total numbers of these operations in rows $\{\oplus, \&\}$ for multiplication and squaring in distinct bases of $GF(2^{191})$. Here and below we assume implementation the fastest of the stated algorithms for multiplication in the ring $GF(2)[X]$ [8]. Here and below in tables there are represented data derived from estimates of the complexity of operations and confirmed by computer experiment. Column "Δ over the ring" contains numbers of operations without operations X_R of multiplication in the ring.

3 Multiplication and Squaring in $GF(2^{191\times 4})$

The field $GF((2^{191})^4)$ contains a 1-type o.n.b. over the subfield $GF(2^{191})$. Operations of addition, multiplication, and squaring in these bases can be implemented using operations in $GF(2^{191})$ in p.b, a.p.b, or p.o.n.b of this basic field. It follows that there are 6 combinations of bases of basic field and its extension, and each of them can be chosen to implement operations of Tate pairing, final exponentiation, and operation of secret working key computing. Together with operations considered in the second

Table 1. Comparison of Operations in $GF(2^{191})$

Bases of $GF(2^{191})$	Minimal polynomial	Logical operations	Squaring	j-Times squaring	Multiplication	Δ over the ring
a.p.b.	Ponb(X)	$\oplus$	1026	1026	15798	1418
		&	0	0	5724	0
		{$\oplus$,&}	1026	1026	21522	1418
p.o.n.b.	Ponb(X)	$\oplus$	0	0	16311	1931
		&	0	0	5724	0
		{$\oplus$,&}	0	0	22035	1931
p.b	Ponb(X)	$\oplus$	1408	1408	16371	1991
		&	0	0	5724	0
		{$\oplus$,&}	1408	1408	22095	1991
p.b	Ppb(X)	$\oplus$	99	$99j$	14758	378
		&	0	0	5724	0
		{$\oplus$,&}	99	$99j$	20482	378

section, algorithms of these operations use operations of multiplication in 4-degree extension of the field $GF(2^n)$ and operation of raising to power 2^j.

Let $\mathbf{a}^{(j)}_{apb\_4nP}$ denote the operation of raising an element $\mathbf{a}$ to the power 2^j using operation $z^{(j)}_{apb}$ of basic field and p.b. of its extension.

Let $\times_{apb\_4nP}$ be a multiplication using a.p.b. of basic field with multiplication $\times_{apb}$ and p.b. of its extension.

Analogous notation $\mathbf{a}^{(j)}_{apb\_4nN}$, $\mathbf{a}^{(j)}_{ponb\_4nP}$, $\mathbf{a}^{(j)}_{ponb\_4nN}$, $\times_{apb\_4nN}$, $\times_{ponb\_4nP}$, $\times_{ponb\_4nN}$, $\times_{pb\_4nP}$, $\times_{pb\_4nN}$ are used for operations in other combinations of field extension and basic field.

Denote $+_{4n}$ an addition in field extension in any of its basis and any basis of basic field, its complexity equals $4n$.

In can be shown that operations $\times_{apb\_4nN}$, $\times_{ponb\_4nN}$, $\times_{pb\_4nN}$ can be implemented performing 9 multiplications an 22 additions in the field $GF(2^n)$.

So complexity of $\times_{apb\_4nN}$ equals 4 $L(\times_{apbN}) + 22n$. Complexity of operations $\times_{apb\_4nP}$, $\times_{ponb\_4nP}$, $\times_{pb\_4nP}$ exceed these values of $6n$ accordingly numbers of n-bit additions for converting between o.n.b. and p.b. of field extension.

Complexity of multiple squaring is estimated analogously.

Numbers of logical operation for multiplication in distinct bases of the field $GF(2^{191})$ and its extension are presented in Table 2.

Table 2. Comparison of Operations in $GF(2^{191\times 4})$

Base of $GF(2^n)$	Minimal polynomial	Logical and n-bit operations	Numbers of logical operations if there are used the bases of $GF(2^{n\times 4})$ over $GF(2^n)$, $n = 191$:					
			o.n.b.			p.b		
			$\times$	squaring	661-times squaring	$\times$	squaring	661-times squaring
a.p.b	Ponb(X)	⊕	146384	4104	1026	147530	5250	5250
		&	51516	0	0	51516	0	0
		{⊕,&}	197900	4104	1026	199046	5250	5250
p.o.n.b	Ponb(X)	⊕	151001	0	0	152147	1146	1146
		&	51516	0	0	51516	0	0
		{⊕,&}	202517	0	0	203663	1146	1146
p.b	Ppb(X)	⊕	137024	396	65439	138170	1542	261756
		&	51516	0	0	51516	0	0
		{⊕,&}	188540	396	65439	189686	1542	261756

4 Tate Pairing and Final Exponentiation Operations

Let us consider four variants of Tate pairing computing with root extraction on supersingular elliptic curve $Y^2 = X^3 + X + \mathbf{b}$ [9].

(a) A.p.b. of the field $GF(2^{191})$, o.n.b of its extension.

Algorithm $Pairing\_apb\_onb(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{apb\_onb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{apb\_onb}$ (an element of the extension field with all coefficients being 0 s and 1 s of the field $GF(2^{191})$), $\mathbf{b} = 1_{apb}$ (identity element represented in a.p.b of $GF(2^n)$).

```
$\mathbf{C} = [1_{apb}, 1_{apb}, 1_{apb}, 1_{apb}]$
$\mathbf{t} = \mathbf{t}_{apb\_onb}$
$\mathbf{s} = \mathbf{t}^{(1)}_{apbP\_4nN}$
for $i = (1,n)$:
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{apbP}{}^{(1)}$
    $\boldsymbol{\beta} = \boldsymbol{\alpha}_{apbP}{}^{(1)}$
    $\mathbf{z} = \boldsymbol{\alpha}+\mathbf{x}$
    $\mathbf{v} = \boldsymbol{\alpha}\times_{apbP}\mathbf{x}$
    $\mathbf{w} = \mathbf{z} + \mathbf{v}+\boldsymbol{\beta}+\mathbf{y}+1_{apb}$
    $\mathbf{u} = [\mathbf{z}\times_{apbP}\mathbf{t}[0]),\ \mathbf{z}\times_{apbP}\mathbf{t}[1]),\ \mathbf{z}\times_{apbP}\mathbf{t}[2]),\ \mathbf{z}\times_{apbP}\mathbf{t}[3]),),$
    $\mathbf{v} = \mathbf{z}+1_{apb}$
    $\mathbf{r} = [\mathbf{v}\times_{apbP}\mathbf{t}_{apb}\mathbf{s}\ [0],\ \mathbf{v}\times_{apbP}\mathbf{s}[1],\ \mathbf{v}\times_{apbP}\mathbf{s}[2],\ \mathbf{v}\times_{apbP}\mathbf{s}[3]]$
    $\mathbf{v} = [\mathbf{w},\mathbf{w},\mathbf{w},\mathbf{w}] +_{4n} \mathbf{u} +_{4n} \mathbf{r}$
    $\mathbf{C} = \mathbf{C}\times_{apbP\_4nN}\boldsymbol{v}$
    $\mathbf{x} = \mathbf{x}^{(n-1)}_{apb}$
    $\mathbf{y} = \mathbf{y}^{(n-1)}_{apb}$
```

Complexity of this algorithm estimated accordingly numbers if multiplication, addition and squaring operations in them:

$$\mathrm{L}_{Pairing\_apb\_onb}(n) = 191(2L(\mathbf{z}_{apb}^{(1)}) + 2\mathrm{L}(\mathbf{z}_{apb}^{(190)}) + L(\times_{apbP}) + L(\times_{apb\_4nN}) + 15L(+)).$$

Remark that here and below multiplication with multiples **t** or **s** containing trivial elements are not taken into account in assessing complexity of pairing, $L(+)$ is complexity of addition in $GF(2^{191})$.

(b) P.o.n.b. of the field $GF(2^{191})$, o.n.b of its extension.

Algorithm $Pairing\_ponb\_onb(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{ponb\_onb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{ponb\_onb}$ when $\mathbf{b} = 1_{ponb}$ (that is, the identity element represented in p.o.n.b. of $GF(2^n)$) differs from just considered only in the type used in operations in the notation of which "apb" is replaced by "ponb", 1_{apb} is replaced by 1_{ponb}.

Hence complexity of this pairing operation is represented by formula

$$L_{Pairing\_ponb\_onb}(n) = 191(2L(\mathbf{z}_{ponb}^{(1)}) + 2L\mathbf{z}_{ponb}^{(190)} + L(\times_{ponbP}) + L(\times_{ponb\_4nN}) + 15L(+)).$$

(c) A.p.b. of the field $GF(2^{191})$, and p.b. of its extension.

Algorithm $Pairing\_apb\_pb(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{apb\_pb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{apb\_pb}$ when $\mathbf{b} = 1_{apb}$.

```
C = [$1_{apb}$, $0_{apb}$, $0_{apb}$, $0_{apb}$]
t = $\mathbf{t}_{apbpb}$
s = $\mathbf{t}_{apbpbapbP\_4nPb}{}^{(1)}$
for $i$ = (1,$n$):
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{apbP}{}^{(1)}$
    $\boldsymbol{\beta} = \boldsymbol{\beta}_{apbP}{}^{(1)}$
    $\mathbf{z} = \boldsymbol{\alpha}+\mathbf{x}$
    $\mathbf{v} = \boldsymbol{\alpha}\times_{apbP}\mathbf{x}$
    $\mathbf{w} = \mathbf{z} + \mathbf{v} + \boldsymbol{\beta} + \mathbf{y} + 1_{apb}$
    $\mathbf{u}$ = [$\mathbf{z}\times_{apbP}\mathbf{t}[0]$, $\mathbf{z}\times_{apbP}\mathbf{t}[1]$, $\mathbf{z}\times_{apbP}\mathbf{t}[2]$, $\mathbf{z}\times_{apbP}\mathbf{t}[3]$]
    $\mathbf{v} = \mathbf{z}+1_{apb}$
    $\mathbf{r}$ = [$\mathbf{v}\times_{apbP}\mathbf{s}[0]$,$\mathbf{v}\times_{apbP}\mathbf{s}[1]$,$\mathbf{v}\times_{apbP}\mathbf{s}[2]$, $\mathbf{v}\times_{apbP}\mathbf{s}[3]$]
    $\mathbf{v}$ = [$\mathbf{w}$, $0_{apb}$, $0_{apb}$, $0_{apb}$] + ${}_{4n}\mathbf{u}$ + ${}_{4n}\mathbf{r}$
    $\mathbf{C} = \mathbf{C}\times_{apbP\_4nPb}\mathbf{v}$
    $\mathbf{x} = \mathbf{x}_{apbP}^{(n-1)}$
    $\mathbf{y} = \mathbf{y}_{apbP}^{(n-1)}$
```

Its complexity is the following:

$$\begin{aligned} L_{Pairing\_apb\_pb}(n) = 191(2L(\mathbf{z}_{apb}^{(1)}) + 2L(\mathbf{z}_{apb}^{(190)}) \\ + L(\times_{apbP}) + \mathrm{L}(\times_{apb\_4nP}) + 11\mathrm{L}(+)). \end{aligned}$$

(d) P.o.n.b. of the field $GF(2^{191})$, p.b. of its extension.

Algorithm $\mathit{Pairing\_ponb\_pb}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{ponb\_pb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{ponb\_pb}$ when $\mathbf{b} = 1_{ponb\_pb}$ can be obtained from the considered algorithm $\mathit{Pairing\_apb\_pb}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{apb\_pb}, \mathbf{b})$ via substitution of indices of operations, pairing parameter, and the field identity element. Its complexity is estimated by formula

$$\mathrm{L}_{Pairing\_ponb\_pb}(n) = 191(2L(\mathbf{z}^{(1)}_{ponb}) + 2L(\mathbf{z}^{(190)}_{aonb}) \\ + L(\times_{ponbP}) + L(\times_{ponb\_4nP}) + 11L(+)).$$

Now consider two variants of Tate pairing computing without root extraction on supersingular elliptic curve $Y^2 = X^3 + X + \mathbf{b}$ [9].

(a) P.b. of the field $GF(2^{191})$, o.n.b. of its extension.

Algorithm $\mathit{Pairing\_pb\_onb}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{pb\_onb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{pb\_onb}$ when $\mathbf{b} = 1_{pb}$.

```
$\mathbf{C} = [1_{pb}, 1_{pb}, 1_{pb}, 1_{pb}]$
$\mathbf{t} = \mathbf{t}_{pbonb}$
$\mathbf{s} = \mathbf{t}_{pb\_4nN}{}^{(1)}$
$\mathbf{u} = \mathbf{x}^{(1)}_{\mathrm{pb}}$
$\mathbf{v} = \mathbf{u}$
$\mathbf{y} = \mathbf{y}^{(1)}_{\mathrm{pb}}$
for $i = (1,n)$:
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{(4)}_{\mathrm{pb}}$
    $\boldsymbol{\beta} = \boldsymbol{\beta}^{(4)}_{\mathrm{pb}}$
    $\mathbf{w} = \boldsymbol{\alpha}\times_{\mathrm{bp}}(\mathbf{v} + 1_{\mathrm{pb}}) + \mathbf{u}+\mathbf{y} + \mathbf{b}+((n\text{-}1)/2)_{\mathrm{pb}}$
    $\mathbf{v} = \boldsymbol{\alpha}+\mathbf{v}$
    $\mathbf{r} = \mathbf{v}+1_{pb}$
    $\mathbf{a} = [\mathbf{w} + \mathbf{v}\times_{pb}\mathbf{t}\,[0] + \mathbf{r}\times_{pb}\mathbf{s}[1], \mathbf{w} + \mathbf{v}\times_{pb}\mathbf{t}_{pb}[2] + \mathbf{r}\times_{pb}\mathbf{s}[3],$
    $\mathbf{w} + \mathbf{v}\times_{pb}\mathbf{s}\,[0] + \mathbf{r}\times_{pb}\mathbf{s}[1], \mathbf{w} + \mathbf{v}\times_{pb}\mathbf{s}[2] + \mathbf{r}\times_{pb}\mathbf{s}[3]]$
    $\mathbf{C} = \mathbf{C}^{(1)}_{\mathrm{pb}\_4nN}\times_{pb\_4nN}\mathbf{a}$
    $\mathbf{u} = \mathbf{u} + \mathbf{v}+1_{pb}$
    $\mathbf{v} = \mathbf{v} + 1_{pb}$
```

Its complexity is estimated as follows:

$$L_{Pairing\_pb\_onb}(n) = 191(2L(\mathbf{z}^{(1)}_{pb}) + 2\mathrm{L}(\mathbf{z}^{(4)}_{pb}) + 2L(\mathbf{z}^{(190)}_{aonb}) \\ + L(\times_{pb}) + 2\mathrm{L}(\times_{\mathrm{pb}\_4nN}) + \mathrm{L}\left(\mathbf{a}^{(1)}_{ab\_4nN}\right) + 16\mathrm{L}(+)).$$

(b) P.b. of the field $GF(2^{191})$, p.b. of its extension.

Algorithm $\mathit{Pairing\_pb\_pb}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{x}, \mathbf{y}, \mathbf{t}_{pb\_pb}, \mathbf{b})$ for pairing of points $P = (\boldsymbol{\alpha}, \boldsymbol{\beta})$, $Q = (\mathbf{x}, \mathbf{y})$ using pairing parameter $\mathbf{t}_{pb}$ when $\mathbf{b} = 1_{pb}$ differs from just considered in four rows:

$\boldsymbol{C} = [1_{pb}, 0_{pb}, 0_{pb}, 0_{pb}]$
$\mathbf{t} = \mathbf{t}_{pb\_pb}$
$\mathbf{s} = \mathbf{t}^{(1)}_{pb\_4nP}$
$\mathbf{C} = \mathbf{C}^{(1)}_{\mathrm{pb}\_4nP} \times_{pb\_4nP}$

Its complexity is represented by formula

$$\mathrm{L}_{Pairing\_pb\_onb}(n) = 191(2L(z^{(1)}_{pb}) + 2L(z^{(4)}_{pb}) + 2L(z^{(190)}_{aonb}) + L(\times_{pb}) + 2L(\times_{\mathrm{pb}\_4nP}) + L\left(\mathbf{a}^{(1)}_{ab\_4nP}\right) + 16L(+)).$$

Table 3 presents data on the number of logical operations executed considered pairing algorithms on supersingular elliptic curve $Y^2 = X^3 + X + 1$ over $GF(2^{191})$ (1 corresponds to 29910607 “xor”, or 10875600 “and” or 43094757 of both these operations). In the tables below we also provide better bounds (given in parentheses) obtained via conversion to a basis with faster implementation of the corresponding operation.

Table 3. Comparison of complexity of pairing algorithms

Base of $GF(2^n)$	Minimal polynomial	Logical and n-bit operations	Relative numbers of logical operations if there are used the bases of $GF(2^{n\times4})$ over $GF(2^n)$, $n = 191$	
			o.n.b.	p.b
			Algorithms with root extraction	
a.p.b	Ponb(X)	⊕	≈1.0753	≈1.0826 (≈1.0753)
		&	≈1.0053	≈1.0053
		{⊕,&}	≈1.0551	≈1.0605 (≈1.0551)
p.o.n.b	Ponb(X)	⊕	≈1.0826 (≈1.0753)	≈1.0928 (≈1.0754)
		&	≈1.0053	≈1.0053
		{⊕,&}	≈1.0590 (≈1.0551)	≈1.0680 (≈1.0551)
			Algorithms without root extraction	
p.b	Ppb(X)	⊕	1	≈1.0122(1)
		&	1	≈1
		{⊕,&}	1	≈1.0089(1)

For the considered supersingular elliptic curve over the field $GF(2^{191})$, the final exponent is

$d = 3091630018413806675756281512823633589197041669549687929671602408959$
$84012937857959440293752760129934932222669494907787798498735918079301$
$8784436808613949303377539749281529855.$

Taking into account that in binary expansion of this number the units take the places 0–95, 97–190, 192–381, 478, and 573 one can represent this exponent as

$$\begin{aligned} d &= (((2^{95}+1)2^{286} + \left(2^{190}-1\right))2^{95} + \left(2^{94}-1\right))2^{97} + \left(2^{96}-1\right) \\ &= \left(\left(a_0 2^{286} + a_1\right)2^{95} + a_2\right)2^{97} + a_3. \end{aligned}$$

As a corollary, final exponentiation algorithm corresponds to the formula $\mathbf{x}^d = (((\mathbf{y}_0)^{2^{286}}\mathbf{y}_1)^{2^{95}}\mathbf{y}_2)^{2^{97}}\mathbf{y}_3$, where $\mathbf{y}_0 = \mathbf{x}^{a_0} = \mathbf{x}^{95}\mathbf{x}$ and the remaining elements $\mathbf{y}_1 = \mathbf{x}^{a_1}, \mathbf{y}_2 = \mathbf{x}^{a_2}, \mathbf{y}_3 = \mathbf{x}^{a_3}$, can be computed by the additive chain 1,2,4,8,10,14,20,40,80,94,96,160,180,190 of lengths 13.

This allows obtaining the following program of final exponentiation using a.p.b. of basic field and p.o.n.b. of its extension.

x = a;	$\mathbf{v} = \mathbf{x}^{(1)}_{apbP\_4N}$;	$\mathbf{z}_1 = \mathbf{v}\times_{apbN\_4N}\mathbf{x}$;	$\mathbf{z} = \mathbf{v}\times_{apbN\_4N}\mathbf{z}_1$;
$\mathrm{v} = \mathrm{z}^{(2)}_{1ponbP\_4N}$;	$\mathbf{v} = \mathbf{z}^{(4)}_{ponbP\_4N}$;	$\mathbf{z} = \mathbf{y}\times_{apbN\_4N}\mathbf{z}$;	$\mathbf{v}_2 = \mathbf{x}^{(2)}_{apbP\_4N}$;
$\mathbf{v} = \mathbf{z}\times_{apbP\_4nN}\mathbf{v}$;	$\mathbf{z}_2 = \mathbf{v}\times_{apbP\_4nN}\mathbf{z}_1$;	$\mathbf{v} = \mathbf{x}^{(4)}_{apbP\_4N}$;	$\mathbf{z} = \mathbf{v}\times_{apbP\_4nN}\mathbf{z}_2$;
$\mathbf{v}_{10} = \mathbf{x}^{(10)}_{apbP\_4N}$;	$\mathbf{z}_3 = \mathbf{z}_2\times_{apbP\_4nN}\mathbf{z}_2$;	$\mathbf{z}_4 = \mathbf{z}_2\times_{apbN\_4nN}\mathbf{z}_2$;	$\mathbf{v}_3 = \mathbf{x}^{(20)}_{apbP\_4N}$;
$\mathbf{z} = \mathbf{v}_3\times_{apbP\_4nN}\mathbf{z}_3$;	$\mathbf{z}_5 = \mathbf{v}\times_{apbN\_4nN}\mathbf{z}_4$;	$\mathbf{x}^{(40)}_{apbP\_4N}$;	$\mathbf{z} = \mathbf{v}\times_{apbP\_4nN}\mathbf{z}_5$;
$\mathbf{z}_6 = \mathbf{z}_5\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{v} = \mathbf{x}^{(14)}_{apbP\_4N}$;	$\mathbf{z} = \mathbf{v}\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{y}_2 = \mathbf{v}_1\times_{apbP\_4nN}\mathbf{z}$;
$\mathbf{z} = \mathbf{v}_2\times_{apbP\_4nN}\mathbf{y}_2$;	$\mathbf{y}_3 = \mathbf{z}_1\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{v} = \mathbf{x}^{(80)}_{apbP\_4N}$;	$\mathbf{z} = \mathbf{z}_6\times\mathbf{v}$;
$\mathbf{z} = \mathbf{v}_6\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{z} = \mathbf{v}_3\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{z} = \mathbf{v}_3\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{z} = \mathbf{v}_3\times_{apbP\_4nN}\mathbf{z}_4$;
$\mathbf{z} = \mathbf{v}_{10}\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{y}_1 = \mathbf{z}_2\times_{apbP\_4nN}\mathbf{z}$;	$\mathbf{z} = \mathbf{x}^{(95)}_{ponbP\_4N}$;	$\mathbf{y}_0 = \mathbf{z}\times_{apbP\_4nN}\mathbf{x}$;
$\mathbf{z} = \mathbf{y}^{(286)}_{0\ ponbP\_4N}$;	$\mathbf{z} = \mathbf{z}\times_{apbN\_4nN}\mathbf{y}_1$;	$\mathbf{z} = z^{(95)}_{ponbP\_4N}$;	$\mathbf{z} = \mathbf{z}\times_{apbP\_4nN}\mathbf{y}_2$;
$\mathbf{z} = \mathbf{z}^{(97)}_{apbP\_4N}$;	$\mathbf{z} = \mathbf{z}\times_{apbP\_4nN}\mathbf{y}_3$;	$\boldsymbol{c} = z$.	

Programs for other combination of fields bases differ only by operation notation. These programs contain 17 multiplication and 14 multi squaring's in the field $GF(2^{191\times 4})$. It is easy to write formula of complexity of these operations and compute their values that are given in Table 4 (1 corresponds to 3421756 logical operations "xor" and "and"). In each case 374 additions, 153 multiplications and 2644 squaring's in $GF(2^{191})$ are executed.

Table 4. Final exponentiation, n = 191

Base of $GF(2^n)$	Minimal polynomial	Logical and n-bit operations	Bases of the field $GF(2^{n\times 4})$ over $GF(2^n)$, $n = 191$:	
			o.n.b.	p.b.
a.p.b.	Ponb	{⊕,&}	1	≈1.0103(≈1.0004)
p.o.n.b.	Ponb	{⊕,&}	≈1.0061(≈1.003)	≈1.0165 (≈1.0005)
p.b.	Ppb	{⊕,&}	≈1.0134	≈1.0192(≈1.0135)

In three partite key agreement protocol, final exponentiation is performed after pairing operation. In Table 5 there are represented total numbers of logical operations for implementations of this composition in distinct combinations of bases (1 corresponds to 44310956 logical operations “xor” and “and”).

Table 5. Comparison of composition of pairing and final exponentiation, n = 191

Bases of $GF(2^n)$	Minimal polynomial	Bases of the field $GF(2^{n\times 4})$ over $GF(2^n)$, $n = 191$:	
		o.n.b.	p.b.
a.p.b	Ponb(X)	≈1.0498	≈1.0555(≈1.0498)
p.o.n.b	Ponb(X)	≈1.05381(≈1.0498)	≈1.0629(≈1.0499)
		Algorithms without root extraction and final exponentiation	
p.b	Ppb(X)	1	≈1.0087(≈1)

5 Conclusion

In this chapter, implementation of algebraic operations in finite fields possessing 2-type or 3- type optimal normal basis and in its 4-degree extension has been comparatively considered taking into account possibility of using distinct combination of bases. Comparative data were also obtained on the complexity of the implementation of pairing and final exponentiation operations in a three-partite key agreement protocol. Based on these data, we can conclude that although for final exponentiation the best is combination of almost polynomial basic of the base field and optimal normal basis of its extension, pairing and final exponentiation are performed faster in polynomial basis of $GF(2^{191})$ and optimal normal basis of its extension. At the same time, it can be noted that the differences in the complexity of implementation with the use of different combinations of bases are not so significant. The advantage of a polynomial basis of the base field is a consequence of the peculiarities of the pairing algorithm without root extraction.

Acknowledgements. This research was supported by the Russian Foundation for Basic Research, project 17-01-00485a. The authors are grateful to Igor Sergeev for editing and anonymous reviewers for comments.

References

1. Bolotov, A.A., Gashkov, S.B.: On quick multiplication in normal bases of finite fields. Discrete Math. Appl. **11**(4), 327–356 (2001)
2. Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A., Wilson, R.M.: Optimal normal bases in *GF*(p^n). Discrete Appl. Math. **22**, 149–161 (1988/1989)

3. Shokrollahi, J.: Efficient implementation of elliptic curve cryptography on FPGA. PhD thesis, Universität Bonn (2007)
4. von zur Gathen, J., Shokrollahi, A., Shokrollahi, J.: Efficient multiplication using type 2 optimal normal bases. In: WAIFI 2007. LNCS, pp. 55–68 (2007)
5. Bernstein, D.J., Lange, T.: Type-II optimal polynomial bases. In: Arithmetic Finite Fields, Proceedings. LNCS, vol. 6087, pp. 41–61 (2010)
6. Joux, A.: A one round protocol for tripartite Diffie-Hellman. In: ANTS 2000. LNCS, vol. 1838, pp. 385–394 (2000)
7. Menezes, A.J., Vanstone, S., Okamoto, T.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inform. Th. **IT-39**, 1639–1646 (1993)
8. Bernstein, D.J.: Minimum number of bit operations for multiplication. http://binary.cr.yp.to/m.html, (Accessed 2009)
9. Kwon, S.: Efficient tate pairing computation for supersingular elliptic curves over binary fields. Cryptology ePrint archive, Report 2004/303 (2004)