Identity-Based Encryption from the Tate Pairing on Genus Two Curves

Mohammed Zitouni, Sylvain Guilley, Farid Mokrane

Identity-Based Encryption from the Tate Pairing on Genus Two Curves

Mohammed Zitouni[1], Sylvain Guilley[2,3], and Farid Mokrane[4]

[1] Laboratory Analysis, Geometry and Applications CNRS (UMR 7539), Sorbonne Paris Nord University, Paris, France.
zitounimohammmed@gmail.com
[2] Secure-IC S.A.S., France.
sylvain.guilley@secure-ic.com
[3] Télécom Paris, Institut polytechnique de Paris, France.
sylvain.guilley@telecom-paristech.fr
[4] Paris VIII university, Paris, France.
mokrane@math.univ-paris13.fr

Abstract

Identity Based Encryption is an approach to link the public key to an identity. It is an extremely useful asymmetric cryptography type in which public and private keys are computed from a known identifier such as an email address instead of being generated randomly. This allows more flexibility in managing ad-hoc public key encryption and ensuring secure communications. The aim of this work is to improve IBE scheme using the bilinear Tate pairing on genus two curves with ordinary Jacobian over large prime fields. We present a full description of functional IBE scheme using the optimization of the Tate pairing computations. The proposed application answers a question of Boneh and Franklin [2] about the possibility of using the Tate pairing in IBE schemes and represents the first IBE exploiting pairings in genus two. We provide a full description of a functional IBE scheme using the optimization of the Tate pairing computations.

1 Introduction

Pairings were used for the first time in 1991 to attack cryptosystems based on supersingular elliptic curves [7, 5]. Recently, bilinear pairing over elliptic and hyperelliptic curves such as Weil, Tate or optimal Ate pairing have found applications in design of cryptographic protocols.

In 1984, Adi Shamir [10] introduced a new type of cryptographic scheme called Identity-Based Encryption (IBE). This encryption scheme permits the authentication and signature between users on communication. The original motivation of the IBE is to simplify certificate management in email systems. It allows the generation of the public key that permits the recipient of the encrypted email to authenticate himself, obtaining his private key and reading his email. This scheme can be used successfully since it requires less time to generate the public key and better protects the cryptosystem devices against hardware attacks.

Dan Boneh and Matthew Franklin [2] proposed a description of the IBE scheme using Weil pairing on elliptic curves. Their proposed scheme is a useful protocol and provides a cryptosystem that is fully resistant to known hardware attacks. The rising question is whether it is possible to extend this protocol over higher genus curves. Can we find more resistant cryptosystems against hardware attacks? and which require less time and memory to generate parameters. Moreover, how can we improve the Diffie-Hellman problem [4] using the Tate pairing on genus two curves with ordinary Jacobian over a large prime field?

In this paper, we exploit the Tate pairing over genus two hyperelliptic curves with ordinary Jacobian over a large prime field as a bilinear map to improve the Diffie-Hellman assumption in

IBE scheme. This work is a part of the study has been published in my PhD thesis. First, we briefly recall the definition of pairing, Miller algorithm for computing pairing and the description of genus two pairing-friendly curves of the type $y^2 = x^5 + ax$ with ordinary Jacobian [6]. Second, we provide a full description of a concrete IBE algorithm and the timing of implementation results in Python, including curve generation, divisor generation, key generation, encryption and decryption. Then, we give an analysis of the performance and key sizes chosen for a desired security level. Finally, we discuss the correctness of the Tate pairing-based protocol and gave a positive answer of Boneh and Franklin question on the possible use of Tate pairing over high genus curve.

2 Preliminaries

2.1 Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two additive Abelian groups of order ℓ, where ℓ is a prime number and $\mathbb{G}_3$ is a multiplicative Abelian group of order also ℓ.

Definition 1. *A bilinear pairing on* $(\mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_3)$ *is a map:*

$$e : (\mathbb{G}_1, +)\ \times\ (\mathbb{G}_2, +)\ \longrightarrow\ (\mathbb{G}_3, \times)$$

that satisfies the three following requirements:

1. *Bilinearity :* $\forall D_1, D_1^{'} \in \mathbb{G}_1, \forall D_2, D_2^{'} \in \mathbb{G}_2$,
   - $e(D_1 + D_1^{'}, D_2) = e(D_1, D_2)\ e(D_1^{'}, D_2)$,
   - $e(D_1, D_2 + D_2^{'}) = e(D_1, D_2)\ e(D_1, D_2^{'})$,
   - $e(a\ D_1, D_2) = e(D_1, a\ D_2) = e(D_1, D_2)^a, a \in \mathbb{N}^*$.
2. *Non-degeneracy:*
   - $\forall D_1 \in \mathbb{G}_1 - \{0\},\ \exists D_2 \in \mathbb{G}_2\ :\ e(D_1, D_2)\ \neq\ 1$,
   - $\forall D_2 \in \mathbb{G}_2 - \{0\},\ \exists D_1 \in \mathbb{G}_1\ :\ e(D_1, D_2)\ \neq\ 1$.
3. *Easily and efficiently calculable.*

There are four main cryptographic pairings, we briefly recall those that are effectively used in cryptography applications. Let N be the order of the Jacobian group and ℓ be the largest prime factor of N. Let k be the embedding degree (the smallest integer such that ℓ divides $p^k - 1$).

Let $Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell]$ be the ℓ-torsion subgroup of the $Jac_{\mathcal{C}}(\mathbb{F}_{q^k})$. An element D in $Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell]$ verifies $\ell\ D = \mathcal{O}$. Let μ_ℓ be the group of ℓ-th roots of unity in $\mathbb{F}_{q^k}$.

Weil pairing: The Weil pairing is given by

$$\begin{aligned} W : Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell] \times Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell]\ &\longrightarrow\ \mu_\ell \\ (D_1, D_2)\ &\longmapsto\ W(D_1, D_2) = (-1)^\ell \frac{f_{\ell,D_1}(D_2)}{f_{\ell,D_2}(D_1)}. \end{aligned}$$

Where f_{ℓ,D_i}, $i = \{1, 2\}$ is a function given by the divisor $\ell\ D_i\ = div(f)$. To compute pairing we need to evaluate this two rational functions f_{ℓ,D_1} at the divisor D_2 (respectively f_{ℓ,D_2} at the divisor D_1). Efficient algorithms will be given to do this computation.

Tate pairing: The Tate-Lichtenbaum pairing is a bilinear and non-degeneracy map defined by:

$$\begin{aligned} T_\ell : & Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell] \times Jac_{\mathcal{C}}(\mathbb{F}_{q^k})/\ell\, Jac_{\mathcal{C}}(\mathbb{F}_{q^k}) \longrightarrow \mathbb{F}_{q^k}^{\times}/(\mathbb{F}_{q^k}^{\times})^{\ell} \\ & (D_1, D_2) \longmapsto T_\ell(D_1, D_2) = f_{\ell,D_1}(D_2)^{(q^k-1)/\ell}. \end{aligned}$$

Computing the Tate pairing requires two divisors D_1 in the ℓ-torsion subgroup of the Jacobian of the curve over $\mathbb{F}_{q^k}$ and D_2 in the quotient of the Jacobian over ℓ times the subgroup of the Jacobian. Here we need just one evaluation of the rational function f_{ℓ,D_1} at the point D_2. The final exponentiation step are crucial to have a non-degeneracy map that we need to work in it very seriously when we do the implementation.

Ate pairing: To define the Ate paring, the situation is somewhat different because it depends on the family of hyperelliptic curves chosen for the pairing, but in general,we assume that ℓ is approximately near to the order of the Jacobian of the curve over the prime field $\mathbb{F}_q$. Let ϕ be the p^{th} power Frobenius automorphism $\phi : \mathcal{C} \longmapsto \mathcal{C}$. We set $J[\ell] = Jac_{\mathcal{C}}(\mathbb{F}_{q^k})[\ell]$.

$$\begin{aligned} A : & J[\ell] \cap (Ker(\phi - [q])) \times J[\ell] \cap (Ker(\phi - [1])) \longrightarrow \mu_\ell \\ & (\bar{D_1}, \bar{D_2}) \longmapsto A(\bar{D_1}, \bar{D_2}) = f_{q,D_2}(D_1). \end{aligned}$$

With D_1 is a divisor in class of divisors $\bar{D_1}$ in $J[\ell] \cap (Ker(\phi - [1]))$ and D_2 a reduced of the class of divisors $\bar{D_2}$ in $J[\ell] \cap (Ker(\phi - [p]))$ such that D_1 and D_2 have a disjoint support. We can have as a result a shorter algorithm to compute the evaluation of the function by using p and not ℓ for some pairing-friendly curves.

Eta-pairing: This type of pairing was introduced by Barreto et al in 2007 for supersingular curves, for every s integer, the Eta pairing denoted η_s is given by

$$\begin{aligned} \eta_s : & J[\ell] \cap (Ker(\phi - [1])) \times J[\ell] \cap (Ker(\phi - [1])) \longrightarrow \mu_\ell \\ & (D_1, D_2) \longmapsto \eta_s(D_1, D_2) = f_{s,D_1}(\psi(D_2))^{(q^k-1)/\ell}. \end{aligned}$$

The final exponentiation allows to the *eta* pairing to have a unique value in μ_ℓ, ψ is a distortion map permits to have the x-coordinates of points in $\psi(D_2)$ lie in the subfield of $\mathbb{F}_q$.

2.2 Pairing computations

We recall the Miller algorithm [8] used mainly to compute the pairing for higher genus curves. Let D_1 and D_2 be two reduced divisors on the Jacobian. Let f_{ℓ,D_1} be a rational Weil function corresponding to a divisor D_1 and the integer ℓ. Let $f_{\ell,D_1}(D_2)$ be the evaluation of the rational function f at the divisor D_2. Let h be the function that derives from the group law on the Jacobian between the two divisors $\rho([\ell]D_1)$ and $\rho([\ell + i]D_1)$ such that i is integer.
The evaluation of the Miller function f_{ℓ,D_1} at the point D_2 leads immediately to the following algorithm:

Algorithm 1 Miller's Algorithm for hyperelliptic curves

Require: $\ell \in \mathbb{N}$ *and* $D_1, D_2 \in Jac_{\mathcal{C}}$, reduced-divisors with disjoint support.
Ensure: $f_{\ell,D_1}(D_2)$
Write ℓ in binary form: $\ell = \Sigma_{j=0}^{s} \ell_j 2^j$, with $\ell_j \in \{0,1\}$ and $\ell_s = 1$
$D \leftarrow D_1$
$f \leftarrow 1$
for (j from $s-1$ to 0) **do**
    Compute $D \leftarrow [2]\, D$ and extract $h_{(D,D)}$
    $f \leftarrow f^2 \cdot h_{(D,D)}(D_2)$
    if ($\ell_j == 1$) **then**
        Compute $D \leftarrow D \oplus D_1$ and extract $h_{(D,D_1)}$
        $f \leftarrow f \cdot h_{(D,D_1)}(D_2)$
    end if
end for
return f

We note that the Miller algorithm cost over the Jacobian group of genus $g \geq 2$ curve is much higher than the case of the Abelian group of points of elliptic curve. However, it remains more practical and a good efficiency candidate for computing pairings.

2.3 Pairing-friendly curves

The curves are chosen such that certain conditions should be satisfied to ensure an asymmetric bilinear pairing. We use genus two Kawazoe-Takahishi hyperelliptic curves of the type $y^2 = x^5 + ax$ with an ordinary Jacobian over a large prime number. The curves are generated using the analogous Cocks-Pinch method presented by [6].
We give two examples of curves with embedding degree $k = 10$ and the parameters: the characteristic of the field p, the prime factor of the Jacobian order ℓ and the $\rho_{value} = 2\, log(p)/log(\ell)$ of the curve.

Example 1. *The curve is* $\mathcal{C}$*:* $y^2 = x^5 + 5x$ *over* $\mathbb{F}_q$ *such that:*
$p = 3563198463393137462206554962310726289906481729891585337179455123732340206995 8/$
$4217744030295935575430536019296464529224387708105818762032518049120907068165554 0/$
$010143795766036580111929 7.$
$q = p^k.$
$\ell = 4331317295229299123996606097779726808684376874887092837694990654326747650095 4/$
$3511172407295145437370569227602880232125000 01.$
$\rho_{value} = 2.985$
The size of p *is (**604** bits) and the size of* ℓ *is (**405** bits).*

Example 2. *The curve is* $\mathcal{C}$*:* $y^2 = x^5 + 3x$ *over* $\mathbb{F}_q$ *such that:*
$p = 3820073953072486836991634724533268868070604635095013679075381981018885799654 4/$
$4282032540867528192243628341074423888083548748538184700767314625016739951095964 0/$
$9088786444013296031670040837917770331165532248022832898839490474788020710094154 9/$
$0574068957599412908632723409820511617481 7.$
$q = p^k.$
$\ell = 4537059816305826341615501375870572589990656903595079119827632620738944776296 4/$
$8483966205774838740034915502069742254838141042980960826434734210360843301691197 6/$

124308917619155308960673924 81.
$\rho_{value} = 2.990$
*The size of p is (**932** bits) and the size of ℓ is (**617** bits) .*

3 A concrete IBE system using the Tate pairing

The identity-based encryption scheme has four main algorithms: `Setup`, `Extract`, `Encrypt` and `Decrypt`. In the following, the description of these algorithms, alongside with their input and output parameters, used in this work:

1. `Setup:` this step produces a system parameters noted (params) and the master-key by using the secret element k.

2. `Extract:` this algorithm requires three parameters, namely an arbitrary $ID \in \{0,1\}^*$, some params and master-key. It gives as output a private key d. It's called `Extract` algorithm because it extracts a private key d from the given public key.

3. `Encrypt:` in this step, the algorithm encrypts a message $M \in \mathcal{M}$ to a ciphertext $C \in \mathbf{C}$ by using parameters params and the arbitrary parameter ID.

4. `Decrypt:` this algorithm is the reverse step of the algorithm `Encrypt`, it helps to decrypt the ciphertext C. It requires parameters params and the private key d.

$$\forall M \in \mathcal{M} : \texttt{Decrypt}(params, C, d) = M$$

$$where \quad C = \texttt{Encrypt}(params, ID, M)$$

Let $\mathcal{C}$ be a hyperelliptic curve defined by the affine equation $y^2 = x^5 + ax$ over the finite field $\mathbb{F}_q$ of characteristic the prime number p and the embedding degree k. Let $Jac_{\mathcal{C}}(\mathbb{F}_q)$ be the Jacobian group of the curve $\mathcal{C}$ and $Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$ be the ℓ-torsion subgroup. A divisor $D \in Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$ is of the order ℓ such that $\ell D = 0$. We need here to construct two points D and W of order ℓ in the subgroup $Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$ expressed by their Mumford representation. we recall here an algorithm that we called `SelectDivisor`. This algorithm works as follows:

1. Let $x \in \mathbb{F}_{\text{!}}$ be an input, we compute $y = (x^5 + a\ x)^{1/2}$ such that $P = (x, y) \in \mathcal{H}$. (we need to determine at most two points).

2. Compute the Jacobian on this set of points $d = Jac([P_1, P_2]) \in Jac_{\mathcal{C}}(\mathbb{F}_q)$.

3. Let $\#Jac_{\mathcal{C}}(\mathbb{F}_q) = h\ \ell$ and set $d_h = h\ d$

4. Let $D = \texttt{DivReduction}(d_h)$ a reduced divisor of order ℓ.

5. Output $\texttt{SelectDivisor}(x) = D$

The function `DivReduction` is an algorithm of reduction of a divisor represented by its Mumford representation $D = [u(x), v(x)]$, it can be summarized by the following algorithm:

Algorithm 2 `DivReduction`

1: **Require:** $D = [u(x), v(x)]$, semi-reduced divisor.
2: **Ensure:** $D_r = [u_r(x), v_r(x)]$ reduced with $D_r \sim D$.
3: Compute: $u_r(x) \leftarrow (f(x) - v(x)\ h(x) - v(x)^2)\ /\ u(x)$
4: Compute: $v_r(x) \leftarrow (-h(x) - v(x))\ mod\ u_r(x)$
5: **if** $(deg(u_r(x)) > g)$ **then**
6: $u(x) \leftarrow u_r(x)$,
7: $v(x) \leftarrow v_r(x)$
8: Go to step **3:**
9: **end if**
10: Make $u_r(x)$ monic.
11: **return** $[u_r(x), v_r(x)]$.

We apply the same procedure to construct another element $W \in Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$. We obtain as finally D and W two points of order ℓ in Jacobian subgroup $Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$. D will be the first input of the Tate pairing as the point in the group of ℓ-torsion divisors in $\mathbb{F}_q$. However, the second input is an element in $Jac_{\mathcal{C}}(\mathbb{F}_q)/\ell\ Jac_{\mathcal{C}}(\mathbb{F}_q)$.
Therefore, the second input to the Tate pairing can be taken to be an element of the ℓ-torsion subgroup of the Jacobian. However, to ensure a non-trivial pairing value, the divisors D and W must have disjoint support. We note $\mathbb{G}_1$ the ℓ-torsion group $Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$, $\mathbb{G}_2$ the subgroup of $\mathbb{F}_q^{\times}$ of order ℓ. The Tate pairing will be the mapping $T : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ as defined before. It is a non-degenerate map and also a bilinear application such that for all $D,\ W \in \mathbb{G}_1$ and for all $a,\ b \in \mathbb{Z}$ we have $T(a\ D, b\ W) = T(D, W)^{ab}$.
Here we describe in detail the scheme of the identity based encryption and the main contribution of using the Tate pairing. Further, the discussion of the chosen data of the four algorithms: `Setup`, `Extract`, `Encrypt` and `Decrypt`.
The first step `Setup` can be done into two algorithms `Setup1` that allows us to check the version number, if it is right, then execute the second algorithm `Setup2` as follows:

Algorithm 3 `Setup1`

1: **Require:** *ver* an integer version number, n security parameter.
2: **Ensure:** *params* public parameters and s master secret.
3: Determine the selected version *Sver*
4: **if** $(ver = Sver)$ **then**
5: Go to algorithm `Setup2`.
6: **else**
7: Go to step **3:**
8: **end if**
9: **return** *params* and s.

The following algorithm `Setup2` is the triplet of three sub-algorithms: `SelectSecurityParams`, `ConstructCurve` and the `SelectDivisor`.

The first sub-algorithm `SelectSecurityParams` determines the security parameters n_p, n_q and the hash function corresponding to the desired version and security parameter n. The size of data and the standard secure hash algorithms SHA-224, SHA-256, SHA-384, SHA-512 are specified in [3].

Example 3. *Possible outputs of the sub-algorithm* `SelectSecurityParams`*:*

1. *if* $n = 1024$*, then* $n_p = 512$*,* $n_q = 160$ *and the hash function [SHA-224]*
2. *if* $n = 2048$*, then* $n_p = 1024$*,* $n_q = 224$ *and the hash function [SHA-224]*
3. *if* $n = 3072$*, then* $n_p = 3840$*,* $n_q = 256$ *and the hash function [SHA-256]*
4. *if* $n = 15360$*, then* $n_p = 7680$*,* $n_q = 512$ *and the hash function [SHA-512]*

The second sub-algorithm `ConstructCurve` constructs, selects, generates and implements the genus two hyperelliptic curve $\mathcal{C}$ corresponding to the desired security level, n_p and n_q. This algorithm was tested for the time taken for

a) Curve generation

b) Jacobian Computation

Table 1 represents the implementation results for the `ConstructCurve` sub-algorithm for the security levels of 192 and 256.

The third sub-algorithm `SelectDivisor` selects a divisor of order ℓ corresponding to the desired size in $Jac_{\mathcal{C}}(\mathbb{F}_q)[\ell]$. In the following Table 2 the time taken for:

a) `SelectDivisor`: the selection of divisor of order ℓ

b) Computation of the public parameter D_{pub}

Finally, the outputs of the first algorithm can be summarized into:

1. Curve parameters: p, ℓ and let $\mathbb{G}_1$ and $\mathbb{G}_2$ two groups of order ℓ
2. Divisor of order ℓ
3. $hash_1$ and $hash_2$ hash functions such that:
   $hash_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1^*$ and $hash_2 : \mathbb{G}_2 \longrightarrow \{0,1\}^n$
4. $D_{pub} = s\ D$, s master secret.

Let T be a bilinear map represented by the Tate pairing define as: $T : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$.
Let $params = \{\ell, \mathbb{G}_1, \mathbb{G}_2, T, n, D, D_{pub}, hash_1, hash_2\}$ be the system parameters. The performance analysis is represented in the Table 3.

The second step of IBE is the algorithm `Extract` that returns a private key from *params*, s and an identity string ID. This algorithm takes as input a string $ID \in \{0,1\}^*$, computes $W_{ID} = hash_1(ID)$ and sets the private key $d_{ID} = s\ W_{ID}$. This operation is shown in Figure 1. The timing of implementation is shown in Table 4.

The third step of IBE is the encryption of a session key using the public parameters *params* and an identity ID. It is represented by `Encrypt` algorithm. Briefly, the description of the encryption operation is as follows: let $M \in \mathcal{M}$ be a random symmetric message. The algorithm takes the message M, *params* and ID as inputs. It computes the Tate pairing between W_{ID} and D_{pub} and the result is denoted g_{ID} such that $g_{ID} = T(W_{ID}, D_{pub}) \in \mathbb{G}_2^*$. Then, it selects a random integer $r \in \mathbb{Z}_\ell^*$. The output is the ciphertext C which can be represented by the following operation:

$$C = \texttt{Encrypt}(r\ D, M \oplus hash_2(g_{ID}^r)).$$

Table 1: `ConstructCurve` for each security level.

Sub-algorithm: `ConstructCurve-192`	
Security level	192
Hyperelliptic curve equation	$\mathcal{C}$: $y^2 = x^5 + 5x$ $k = 10$ $p = 14831053227820099020851702242827202301123849999819087/$ $98384618570024315076753171532737218888271626681165643471/$ $7697.$ (**373** bits) $\ell = 24145784518520028448286918486097375436360607723491093/$ $78671891649698743040001.$ (**251** bits) $\rho_{value} = 2.976$
Curve generation	Time (Milliseconds): 1.207
Jacobian generation	Time (Microseconds): 33.620
Sub-algorithm: `ConstructCurve-256`	
Security level	256
Hyperelliptic curve equation	$\mathcal{C}$: $y^2 = x^5 + 10x$ $k = 10$ $p = 35548828300835427236598842441009859003569076731124 63/$ $85600841674938197094297832735569882204882345992310138375/$ $64610935788620397522545657388494794369575954134557609951.$ $132252641778247371.$ (**604** bits) $\ell = 43245758482417031182453013163117686895486016836902168/$ $31744214612885361606918203877499165759265532524989350472/$ $4833344340961.$ (**405** bits) $\rho_{value} = 2.985$
Curve generation	Time (Milliseconds): 1.246
Jacobian generation	Time (Microseconds): 38.850

Table 2: `SelectDivisor` for each security level.

Sub-algorithm: `SelectDivisor-192`	
`SelectDivisor`	Time (Milliseconds): 21032.529
public parameter D_{pub}	Time (Milliseconds): 28.837
Sub-algorithm: `SelectDivisor-256`	
`SelectDivisor`	Time (Milliseconds): 46921.726
public parameter D_{pub}	Time (Milliseconds): 39.523

The last step of IBE is decryption represented by the algorithm `Decrypt`. This algorithm decrypts the ciphertext C the output of the encryption operation. It takes three inputs the public parameters *params*, the private key d_{ID} (see 3 in `Extract` algorithm) and the ciphertext C. Let $V = M \oplus hash_2(g_{ID}^r)$ and $U = r\ D$. The operation used to retrieve the decrypted message M is

$$M = (V \oplus hash_2(T(d_{ID}, U))$$

Table 3: Performance analysis of the `Setup` algorithm.

Algorithm: `Setup`		
Security level	192	256
Curve parameters	Time (Milliseconds): 1.210	Time (Milliseconds): 1.250
Divisor	Time (Milliseconds): 21032.529	Time (Milliseconds): 46921.726
D_{pub}	Time (Milliseconds): 28.837	Time (Milliseconds): 39.523

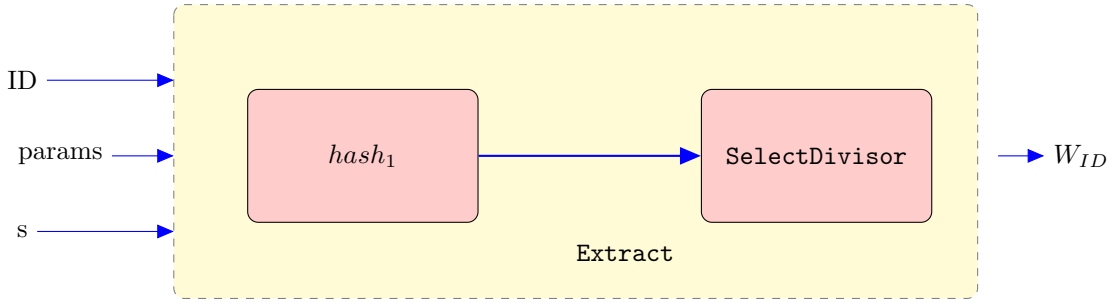

Figure 1: `Extract` algorithm

In the above expressions, the operation on the divisors should be interpreted in the context of arithmetic on the Jacobian group of ℓ-torsion.

4 Performance analysis

4.1 Key-sizes and security levels

The choice of key size is a major issue in this work, precisely for the selection of the curve parameters. On the one hand, the prime order of the subgroup, ℓ, should be large enough to prevent from the known Pollard rho-attack [9]. More precisely, this means that for a security level of n bits, ℓ should be at least of size $2n$ bits. On the other hand, the pairings should be resist against the best discrete logarithm for genus 2 which has an exponential running time and the size of the finite field $\mathbb{F}_{p^k}$ should be large enough to prevent from Kim-Barbulescu new variant of NFS [1]. Thus, to achieve the same security level, we start by selecting suitable

Table 4: Timing of the private key extraction. `Extract` algorithm.

Algorithm: `Extract`		
Security level	192	256
W_{ID}	Time (Milliseconds): 21377.054	Time (Milliseconds): 47138.520
d_{ID}	Time (Milliseconds): 29.154	Time (Milliseconds): 39.236

Table 5: Timing of the `Encrypt` and `Decrypt` algorithms of a message $M \in \mathcal{M}$.

Security level	192	256
Encryption	Time (Milliseconds): 21062.576	Time (Milliseconds): 46962.499
Decryption	Time (Milliseconds): 21406.208	Time (Milliseconds): 47177.756

Table 6: Selected genus two hyperelliptic curves required to obtain desired security levels.

Security level (bits)	**Embedding degree (k)** $2 \leq \rho_{value} \leq 3$	**Subgroup size** ℓ (bits)	**Extension field** q^k (bits)
128	6-10	256	3000-5000
192	10-18	384	7000-9000
256	18-30	512	14000-16000

curves as given in Sec. 2.3. The following table 6 gives the security level, the embedding degree of the curve, ρ_{value} and size of the parameters of the curves. Moreover, the field size depends not only on the security level (192 and 256 bits), but also on the embedding degree k and the polynomial parameterizing p and ℓ ($p(z)$ and $\ell(z)$) in the case of the Kachisa family. In this work, the embedding degree is 10 is considered and the security estimation is improved according to [1]. We give here some workaround options for the recent estimate of the performances of Kim–Barbulescu NFS in $\mathbb{F}_{p^k}$ in the case of embedding degree $k = 10$. In $\mathbb{F}_{p^{10}}$, the polynomials $p(z)$ and $\ell(z)$ have degrees 24 and 16 respectively, the smallest size of p targeting the 128-bit security level is at least 380 bits long (and ℓ is at least 256 bits long, $\rho_{value} \approx 3$). We can conclude the size of p and ℓ needed for 192-bit and 256-bit as shown in the Table 5.

4.2 Resistant to hardware attacks

Early IBE studies focused mostly on the Weil pairing over genus 1 curve, which is based on two so-called Miller loops, but it quickly became evident that variations of the Tate pairing are more efficient and the calculation of one Miller loop and one final exponentiation is common to all of these variations. Later the subject is to optimize software and hardware implementation, to reduce the number of finite-field operations and to execute finite-field and big-integer operations as efficiently as feasible given the machine instructions of a particular target architecture.

In order to decrypt a message, identity-based protocols compute a pairing involving the private key and the plaintext. The pairing used for the Boneh-Franklin protocol was the Weil pairing that accepts two elliptic curve points as inputs and compute the pairing between these points as output. However, the Weil pairing is vulnerable to a fault attack such as side channel attacks (SCA) because we know the Miller algorithm so we know the physical implementation and the number of iterations. The attacker needs to modify the number of iterations when executing Miller's algorithm to find two execution results for consecutive iterations and the ratio between these two results gives a fraction that gives information about the secret point. That's why IBE based on the Weil pairing has become vulnerable. The solution is the use of the Tate pairing as an alternative bilinear map which allows us to remedy this problem because

it requires a final exponentiation which makes it difficult or impossible to find the secret point.

The Tate pairing over genus 1 curves is still a good response for the question of the use of the bilinear pairing. However, we need to generate the curve equation over high characteristic field to ensure a desired security level and it is not appropriate for the newest pairing that requires small sizes of the finite field parameters and to insure a realised implementation on integrated circuits and embedded devices. The solution in this paper is to reduce the size of the keys more and the field characteristic to generate the equation of the curve, while assuring the desired level of security, we thought of the curve of genus two of ordinary Jacobian which allows to have groups of higher order while generating the curve on a field of small characteristics compared to genus 1 curves.
The table 7 summarizes the advantages of the use of the Tate pairing over genus two curves in our article for the same level of the security. On one hand, the Weil pairing benefits from a low computation time and an easy Miller algorithm compared to the Tate at the cost of a weaker resistance to attacks. On the other hand, genus 1 and 2 have comparable medium computational time while genus two remains more resistant to attacks with less exponentiation and characteristic of field compared to the genus 1.

Table 7: The impact of parameters on the choice of pairing for the same desired security level.

Pairing	**Weil**	**Tate (genus 1)**	**Tate (genus 2)**
Characteristic field (size)	very large	large	small to large
Points coordinates (size)	large	large	small
Miller algorithm	easy	low to medium	medium
Final exponentiation	/	high ++	high +
Pairing computation (time)	low	medium	medium
Resistant to attacks	-	+	++

4.3 Correctness of IBE

The correctness of the identity based encryption scheme is verified as follows:

$$\begin{aligned} T(d_{ID}, U) &= T(s\ W_{ID}, r\ D) \\ &= T(W_{ID}, D)^{sr} \\ &= T(W_{ID}, s\ D)^{r} \\ &= T(W_{ID}, D_{pub})^{r} \\ &= g_{ID}^{r}. \end{aligned}$$

5 Conclusion

In this work, we give a positive answer to Boneh and Franklin's question about the possible use of the Tate pairing on higher genus curves as a bilinear map to improve the identity-based encryption scheme. We prove that our scheme is more resistant to hardware attacks compared to genus 1 curves with small characteristic field used to generate the curve equation. We present a complete description of this scheme using the Tate pairing on genus two curves of the type $y^2 = x^5 + ax$ over a large prime field. Moreover, we provide a complete implementation of all the steps and algorithms of this scheme, specifying each time the input and output parameters. First, we give the definition of pairing, the description of pairing-friendly curves of ordinary Jacobian over a large prime field and we recall the main algorithm for computing the Tate pairing. Second, we provide the description of the Identity-Based Encryption scheme and we describe the input and output parameters of algorithms: `Setup`, `Extract`, `Encrypt` and `Decrypt`. Then, We improve the bilinear Tate pairing in the IBE scheme and we present the timing of the implementation results in Python including curve generation, divisor generation, key generation, encryption and decryption. Finally, we give the performance analysis and discuss the correctness of the Tate pairing-based protocol.
Performance results show that our proposed protocol based on the Tate pairing on genus two curves, yields higher resistance to existing attacks compared to the genus one case, while requiring similar pairing computation time, which represents a significant advantage for our proposed scheme.

References

[1] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, 2019.

[2] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[3] Quynh H Dang. Secure Hash Standard. August 4 2015. Secure Hash Standard, Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.FIPS.180-4 (Accessed September 18, 2022).

[4] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[5] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62(206):865–874, 1994.

[6] Mitsuru Kawazoe and Tetsuya Takahashi. Pairing-friendly hyperelliptic curves with ordinary jacobians of type $y^2 = x^5 + ax$. In *International Conference on Pairing-Based Cryptography*, pages 164–177. Springer, 2008.

[7] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *iEEE Transactions on information Theory*, 39(5):1639–1646, 1993.

[8] Victor Miller et al. Short programs for functions on curves. *Unpublished manuscript*, 97(101-102):44, 1986.

[9] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[10] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.