We are IntechOpen, the world’s leading publisher of Open Access books
Built by scientists, for scientists

3,800
Open access books available

116,000
International authors and editors

120M
Downloads

154
Countries delivered to

TOP 1%
Our authors are among the most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com
Biometric Encryption Using Co-Z Divisor Addition Formulae in Weighted Representation of Jacobean Genus 2 Hyperelliptic Curves over Prime Fields

Robert Brumnik¹, Vladislav Kovtun², Sergii Kavun³ and Iztok Podbregar⁴

¹University of Maribor, Faculty of Criminal Justice and Security,
²National Aviation University, Kiev,
³National University of Economic, Kharkov,
⁴University of Maribor, Faculty of Criminal Justice and Security,
¹⁴Slovenia
²³Ukraine

1. Introduction

Security in biometry is a prime concern of modern society. Identity theft is a growing problem in today’s interconnected world. To ensure a safe and secure environment biometrics is used today in many commercial, government and forensic applications. To ensure a high level of security of a biometric system we use Cryptographic algorithms. Though a number of bio-crypto algorithms have been proposed, they have limited practical applicability due to the trade-off between recognition performance and security of the template. Overall, these are very secure, however, they do have a weak point in terms of the procedure and storage of the crypto keys.

Biometric authentication systems should have many exploitable crypto secure points that can be used to compromise the identification system within the optimization process. Biometric encryption with Jacobean Genus 2 Hyperelliptic curves is a security scheme that combines strong cryptographic algorithms with biometric authentication to provide enhanced security. This paper discusses the simple implementation Co-Z divisor addition formulae in a weighted representation of encryption systems for biometric software application.

In this article the authors show a newly developed Co-Z approach to divisor scalar multiplication in Jacobean of Genus 2 Hyperelliptic curves over fields with odd characteristics in weighted coordinates for application in biometric-based authentication systems. We assess the performance of these biometric generation algorithms using Co-Z divisor. This approach is based upon improved additional formulae of weight 2 divisors in weighted divisor representation which, in the most frequent cases are well suited to exponentiation algorithms based on Euclidean addition chains.

2. Cryptographic applications

The progress of civilization and the constantly increasing role of various technologies in human day-to-day activities has lead to the permanent development of access control
security systems for information and physical objects. A number of researchers have studied
the interaction between biometrics and cryptography, two potentially complementary
security technologies (Hao et al., 2005). For such systems, the importance of authentication
and identification subsystems is undeniable. One of the approaches to the implementation
of authentication and identification subsystems is biometric systems. The process of creating
biometric systems of authentication and identification has solved a large variety of problems among which the assurance of confidentiality and integrity of biometric information, which is by no means unimportant. As a rule, cryptographic
transformations are used – encryption and electronic digital signature (EDS). The analysis,
carried out by the authors, has shown the prospectivity of algebraic curves theory for the
implementation of cryptographic transformations.

2.1 Elliptic curves
The usage of elliptic curves (EC) for cryptographic purposes was first suggested by (Koblitz,
1987) and (Miller, 1985). Basiri et al. (2004) present two algorithms for the arithmetic of cubic curves with a totally
ramified prime at infinity. The first algorithm, inspired by Cantor's reduction for
hyperelliptic curves, is easily implemented with a few lines of code, making use of a
polynomial arithmetic package.
In his research, Koblitz (1989) has proved the possibility of usage of hyperelliptic curves
(HEC) in cryptographic applications. Their difference from EC is that for HEC the group
(Jacobian) of more complex structures should be considered – divisors instead of curve
points. It is known that HEC have a variety of advantages over EC: being a richer source of
the Abelian groups (Koblitz, 1989; Menezes and Wu, 1998) they also use a base field of a
smaller size (from the Abelian group, the size of which is defined by product of the base
field size by a curve genus).

2.2 Hyperelliptic curves
Hyperelliptic curve cryptosystems (HCC for short) is a generalization of ECC. It has been
drawing the attention of more and more researchers in recent years. The problem of how to
decrease the amount of addition and scalar multiplication on the Jacobians of hyperelliptic
curves so that the implementation speed can be improved is very important for the practical
use of HCC (Zhang et al., 2001). During the time in which HEC cryptosystems were restricted to academic interest only, they
had no practical application due to the high complexity of software and hardware
implementation, low performance, absence of in-depth studies in the field of cryptoanalysis
of such cryptosystems and the absence of comprehensible algorithms of cryptosystem
parameters generation. Active review research of papers (Koblitz, 1989; Menezes et al.,
1998; Lange 2002c; Matsuo et al., 2001; Miyamoto et al., 2002; Takahashi et al., 2002; Sugizaki
et al. 2002, etc) has allowed us to overcome the majority of the described difficulties. The
authors of publications, offer a variety of approaches which increase the performance of
HEC cryptosystems essentially, and bring them close to the EC cryptosystems.

2.3 Genus 2 HEC cryptosystems over prime fields
The given research is devoted to the development of the approach (Cohen et al., 1998) and to
the improved efficiency of genus 2 HEC cryptosystems over prime fields.
Scalar multiplication operation is used in encryption, decryption and electronic digital signature based on HEC. These computations are relatively expensive when implemented on low-power devices. A widely used standard method is the left-to-right binary method. In accordance with (Koblitz, 1989; Menezes et al., 1998; Lange 2002c; Matsuo et al., 2001; Miyamoto et al., 2002; Takahashi et al., 2002; Sugizaki et al. 2002; Lange, 2002; Kovtun and Zbitnev, 2004) the power consumption traces of divisor addition and doubling are not the same, they can easily be distinguished between these operations and derive the bit of scalar. The first method proposed, with resistance to the side channel attacks (SCA), is Coron’s dummy addition (CDA) (Coron, 1999).

Several SCA-resistant scalar multiplication algorithms have been proposed that are faster than the CDA method. There are three basic approaches with SCA resistance:

- The first is to use indistinguishable additions and doubling algorithms in scalar multiplication (Clavier and Joye, 2001). For example, Jacobi form and Hesse form of EC. However, this requires specially chosen EC and HEC curves and does not work for the standardized curves.

- The second is the double-and-always-add approach. The CDA method is the simplest algorithm of this type. In paper (Okeya and Sakuri, 2000), the authors proposed to use the Montgomery form of EC and extended it to general curves (Brier and Joye, 2002).

- The third approach is to use a special addition chain with a sequence of additions and doublings that does not depend on the bit information of the scalar (Izu and Takagi, 2002).

In this paper, we are interested in scalar multiplication algorithms that do not require specially chosen curves and based on approach (Meloni, 2007) for genus 2 HEC over prime fields.

### 3. Biometric cryptosystems

In a generic cryptographic system the user authentication is possession based. That is, possession of the decrypting key is a sufficient evidence to establish user authenticity. Because cryptographic keys are long and random, (e.g., 128 bits for the advanced encryption standard (AES) (NIST, 2008; Stallings, 2003), they are difficult to memorize. As a result, the cryptographic keys are stored somewhere (for example, on a computer or a smart card) and released based on some alternative authentication (e.g., password) mechanism, that is, upon assuring that they are being released to the authorized users only. Most passwords are so simple that they can be easily guessed (especially based on social engineering methods) or broken by simple dictionary attacks (Klein, 1990).

It is not surprising that the most commonly used password is the word “password”? Thus, the multimedia protected by the cryptographic algorithm is only as secure as the passwords (weakest link) used for user authentication that release the correct decrypting key(s). Simple passwords are easy to crack and, thus, compromise security; complex passwords are difficult to remember and, thus, are expensive to maintain. Users also have the tendency to write down complex passwords in easily accessible locations. Further, most people use the same password across different applications and, thus, if a single password is compromised, it may open many doors. Finally, passwords are unable to provide nonrepudiation; that is, when a password is shared with a friend, there is no way to know who the actual user is. This may eliminate the feasibility of countermeasures such as holding conniving legitimate users accountable in a court of law. Many of these limitations of the traditional passwords can be ameliorated by incorporation of better methods of user authentication. Biometric authentication (Jain et al., 1999; Maltoni et al., 2003) refers to verifying individuals based on
their physiological and behavioural characteristics such as face, fingerprint, hand geometry, iris, keystroke, signature, voice, etc. It is inherently more reliable than password-based authentication, as biometric characteristics cannot be lost or forgotten (cf. passwords being lost or forgotten); they are extremely difficult to copy, share, and distribute (cf. passwords being announced in hacker websites) and require the person being authenticated to be present.

A brief comparison of some of the biometric identifiers based on seven factors is provided (Wayman, 2001):

- universality (do all people have it?),
- distinctiveness (can people be distinguished based on an identifier?),
- permanence (how permanent is the identifier?), and
- collectability (how well can the identifier be captured and quantified?) are properties of biometric identifiers.
- performance (speed and accuracy), acceptability (willingness of people to use), and circumvention (foolproof) are attributes of biometric systems.

4. Background

Let’s observe basic concepts of cryptosystems on HEC. More detailed information can be obtained from (Koblitz, 1989; Menezes and Wu, 1998).

Let $K$ be a field and $\overline{K}$ be the algebraic closure of $K$. Hyperelliptic curve $C$ of genus $g \geq 1$ over $K$ is a set of points $(u, v)$ that satisfy the equation:

$$C : v^2 + h(u)v = f(u), \quad k[u,v]$$

and there are no solutions $(u, v) \in \overline{K} \times \overline{K}$ which simultaneously satisfy the equation (1) and the partial derivative equations with corresponding variables.

$$2u + h(u) = 0, \quad h'(u)v - f'(u) = 0$$

In case of genus 2 HEC, polynomials $f(u)$ and $h(u)$ will be represented as:

$$f(u) = u^5 + f_4u^4 + f_3u^3 + f_2u^2 + f_1u + f_0$$

and

$$h(u) = h_2u^2 + h_1u + h_0, \quad h_i, f_i \in K.$$  

Divisor $D$ is a formal sum of points in $C$:

$$D = \sum_{P \in C} m_P P, \quad m_P \in \mathbb{Z}$$

where only a finite number of the $m_P$ are non-zero.

Divisor $D \in \mathbb{D}^0$ is a principal divisor, if $D = \text{div}(R)$ for some rational function $R \in \overline{K}(C)^*$. The set of all principal divisors, denotes $P_C(\overline{K}) = \{ \text{div}(F) : F \in \overline{K}(C) \}$, in curve $C$ over $\overline{K}$, moreover $P_C(\overline{K})$ is a subgroup of $\mathbb{D}^0$. Generally $P(C) = P_C(\overline{K})$ is called a group of principal divisors of curve $C$. The quotient group $I_C(\overline{K}) = \text{div}^0_C(\overline{K})/P_C(\overline{K})$ is called the Jacobian of the
curve $C$ over $\bar{K}$. The quotient group $J(C) = \text{div}^0(C)/\text{P}(C)$ is called Jacobian of the curve $C$.

Furthermore, we will operate with divisors in the Mumford representation (Menezes, 1998):

$$D = \left(x^2 + u_1x + u_0, v_1x + v_0\right), \quad \deg v < \deg u \leq 2, \quad u \mid f(u) - h(u)v - v^2$$

where $\forall D_i \in J(C), \text{weight}(D_i) = 2, \ i = 1, 2$ \hspace{1cm} (6)

The result $D_3 = D_1 + D_2$ will have a weight$(D_3) = 2$, which helps to avoid consideration of alternative addition methods for divisors of different weight and with intersecting support (containing intersecting sets of points) (Lange, 2003; Wollinger, 2004).

HECC uses a divisor scalar multiplication operation:

$$D + D + \cdots + D = k \cdot D$$ \hspace{1cm} (7)

At the intermediate computation phase of scalar divisor multiplication (scalar multiplier in binary notation) the binary algorithm performs the divisor addition and doubling operation. The addition and doubling algorithms use field $\text{GF}(p)$ multiplicative inversion, which is the most computationally intensive and space critical operation. Projective divisor representation (Miyamoto et al., 2002; Takahashi, 2002; Lange, 2002c) is one of the most popular approaches which allows saving of a field inversion.

In her work Lange (2002c), suggested a weighted divisor representation, being the development of a projective approach.

In weighted representation, the divisor $D$ we can present as Lange (2002c):

$$D = \left(x^2 + u_1x + u_0, v_1x + v_0\right)$$ \hspace{1cm} (8)

which is of the form of:

$$D = \left[U_1, U_0, V_1, V_0, Z_1, Z_2, Z_2^2\right]$$ \hspace{1cm} (9)

while

$$D = \left[x^2 + U_1/Z_1^2x + U_0/Z_1^2, V_1/Z_1^2Z_2x + V_0/Z_1^2Z_2\right]$$ \hspace{1cm} (10)

Note, that arithmetic in Jacobian genus 2 HEC in weighted representation (Lange, 2002c) is the most efficient (Kovtun and Zbitnev, 2004).

5. Co-Z approach

The Co-Z approach was first suggested by Meloni (2007), in order to increase the efficiency of scalar multiplication in EC over $\text{GF}(p)$ with double-free addition chain for the resistance side channel attacks (SCA) (Goundar, 2008). It results in a fixed sequence of operations; hence attackers could not detect any information through SCA. Its principles lie in transformation of EC points in SCA resistant scalar point multiplication in projective and
modified Jacobi representation with the same denominator and further operation with points of identical \( Z \) -coordinates. Note that the Co-Z approach is applicable for algorithms, based on the Euclidian addition chains approach by Meloni (2007) and scalar in the Zeckendorf representation, in order to replace doublings by Fibonacci numbers computations refer to Algorithm A.1. Indeed the Fibonacci sequence is an optimal chain (Meloni, 2007).

The Zeckendorf number representation needs 44\% more digits in comparison with the binary representation. For example a 80-bit integer will require around 115 Fibonacci digits. However, the density of 1’s in this representation in lower (near 0.2764). This means that representing a 80-bits integer requires, an average 40 powers of 2 but only 32 Fibonacci numbers (near 115x0.2764).

More generally, for a \( n \) -bit integer, the classical double-and-add algorithm requires on average \( 1.5 \times n \) operations \( (\frac{7}{2}A + nD) \), where A -addition operation and D -doubling operation) and the Fibonacci-and-add requires \( 1.83 \times n \) operations \( (1.44 \times nF + 0.398 \times nA) \), where A -addition step and F -Fibonacci step). In other words, the Fibonacci-and-add algorithms A.1 require about 23\% more operations (Table 1). Note, that in paper (Lange, 2002c) the simplified version of HEC is used, such that \( h(x) = 0 \), \( f_4 = 0 \) (since replacement \( y \rightarrow y - h_l/2 \) is admissible for the odd field characteristic and the replacement \( x \rightarrow x - f_4/5 \) is admissible if \( p \neq 5 \)), which allowed T. Lange to save 1 multiplication at the step A.2.7. However in this paper we will consider a more general case with \( \deg(h) = 2 \), \( h_i \in F_2 \), \( i = 0, 2 \) \( \deg(f) = 5 \), \( f_5 = 1 \), \( f_i \in GF(p) \), \( i = 0, 4 \).

![Algorithm A.1 Fibonacci-and-add \((k, P)\)](attachment:image)

Table 1. Algorithm A.1 Fibonacci-and-add \((k, P)\)

We can see, in Algorithm A.1 using the Addition and Fibonacci step for the adding divisors. In common case, addition of two reduced divisors in weighted coordinates can be represented by the Algorithm A.2 (Lange, 2002c) (Table 2).

Assuming that \( Z_{11} = Z_{21} = Z_1 \) and \( Z_{12} = Z_{22} = Z_2 \) for \( D_1 \) and \( D_2 \), which allows the transformation of algorithm A.2 into algorithm A.3. Apply the approach described by Meloni (2007) to the divisor addition algorithm suggested by Lange (2002c) and weighted divisor representation. Further circumscribe the derivation of expressions in different steps of A.3 (Table 3).

![Algorithm A.2 Divisor Addition](attachment:image)
Table 2. Algorithm A.2 Addition reduced divisors

<table>
<thead>
<tr>
<th>#</th>
<th>Expression</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Precomputations: ( z_{13} = Z_{13} \cdot Z_{12} ), ( z_{23} = Z_{23} \cdot Z_{22} ), ( z_{12} = z_{11} \cdot z_{13} ), ( z_{22} = z_{21} \cdot z_{23} ), ( U_{21} = z_{11} \cdot U_{22} ), ( U_{20} = z_{11} \cdot U_{20} ), ( \Delta_{21} = z_{11} \cdot V_{21} ), ( \Delta_{20} = z_{11} \cdot V_{20} )</td>
<td>8M</td>
</tr>
<tr>
<td>2</td>
<td>Compute resultant ( r ) for ( u_{1} ) and ( u_{2} ): ( y_{1} = U_{11} \cdot z_{21} - U_{21} ), ( y_{2} = U_{10} \cdot z_{21} + U_{11} \cdot y_{1} + y_{2} \cdot z_{11} ), ( r = y_{2} \cdot y_{3} + y_{2} \cdot U_{11} \cdot z_{21} \cdot Z_{11} \cdot Z_{21} ), ( Z_{2} = Z_{2} \cdot Z_{r} ), ( Z_{2} = Z_{2} \cdot Z_{1} ), ( Z_{2} = Z_{1} \cdot Z_{r} ), ( Z_{2} = Z_{2} \cdot Z_{21} \times Z_{11} \cdot Z_{21} \times Z_{11} )</td>
<td>4S, 11M</td>
</tr>
<tr>
<td>3</td>
<td>Compute almost inverse ( inv = r/u \mod u_{1} ), ( inv = inv \cdot \mod u_{1} ), ( inv = y_{1} )</td>
<td>8M</td>
</tr>
<tr>
<td>4</td>
<td>Compute ( s = (y_{1} - y_{2}) \mod u_{1} ), ( s = s_{1} \cdot x + s_{2} ), ( w_{0} = V_{20} \cdot z_{22} - \Delta_{20} ), ( w_{1} = V_{11} \cdot z_{22} - \Delta_{21} ), ( w_{2} = inv \cdot w_{0} ), ( w_{3} = inv \cdot w_{1} ), ( s_{2} = w_{2} - U_{20} \cdot w_{3} ), ( s_{1} = (inv \cdot z_{1} + inv) \cdot (w_{0} + w_{1}) - w_{2} - (z_{1} + U_{11}) )</td>
<td>3S, 6M</td>
</tr>
<tr>
<td>5</td>
<td>Compute ( l = su_{1} ), ( l = x^{3} + 2x^{2} + x + l + l_{1} ), ( l_{1} = s_{1} \cdot U_{20} ), ( l_{2} = s_{1} \cdot U_{21} ), ( l_{1} = s_{1} \cdot (s_{1} + s_{2}) \cdot (U_{21} + U_{20}) - l_{2} - l_{1} ), ( l_{2} = l_{1} + S )</td>
<td>3M</td>
</tr>
<tr>
<td>6</td>
<td>Compute ( u' = k \cdot (s_{1} + s_{2}) \cdot (s_{1} + s_{2}) \cdot (U_{21} + U_{20}) - l_{2} - l_{1} ), ( U_{0} = S_{0} + y_{1} \cdot (U_{1} + \Delta_{21}) - 2y_{2} + y_{2} \cdot z_{2} + 2v' + Z_{2} \cdot (U_{1} + 2U_{20}) )</td>
<td>6M</td>
</tr>
<tr>
<td>7</td>
<td>Precomputations: ( l_{2} = l_{2} - U_{0}' ), ( w_{0} = l_{2} \cdot U_{0}' ), ( w_{1} = l_{2} \cdot U_{0}' )</td>
<td>2M</td>
</tr>
<tr>
<td>8</td>
<td>Compute ( v' = (h + s_{1} + v_{2}) \mod w' ), ( v' = v_{1} \cdot x + v_{2} ), ( V_{1}' = w_{1} - z_{1} \cdot (l_{2} + V_{1}' - U_{0}') )</td>
<td>3M</td>
</tr>
</tbody>
</table>
1. Compute resultant \( r \) of \( v_1, v_2 \):

\[
y = \frac{U_{v_2} U_{v_1}}{Z_{v_2} Z_{v_1}}, \quad y = -\frac{U_{v_1} y_{v_1} + y_{v_2} U_{v_2}}{Z_{v_1} Z_{v_2}}, \quad r = \frac{y_{v_1} y_{v_2} y_{v_1}^2 + y_{v_2} y_{v_1}^2}{Z_{v_1} Z_{v_2}}.
\]

2. Compute almost inverse \( \text{inv} = r/h \), mod \( v_2 \), \( \text{inv} = \text{inv} + i \cdot \text{inv} \):

\[
\text{inv} = \frac{y}{Z}, \quad \text{inv} = \frac{y}{Z}.
\]

3. Compute \( x = r + i \cdot \text{inv} \), mod \( v_1 \):

\[
u_1 = \frac{U_{v_1} U_{v_2}}{Z_{v_1} Z_{v_2}}, \quad \nu_1 = -\frac{U_{v_1} \nu_{v_2} + \nu_{v_1} U_{v_2}}{Z_{v_1} Z_{v_2}}, \quad \nu_2 = \frac{\nu_{v_1} \nu_{v_2} \nu_{v_1}^2 + \nu_{v_2} \nu_{v_1}^2}{Z_{v_1} Z_{v_2}},
\]

4. Compute \( x' = x + i \cdot \text{inv}' \):

\[
u_1 = \frac{1}{r}, \quad \nu_1 = \frac{1}{r}, \quad \nu_2 = \frac{\nu_{v_1} \nu_{v_2} \nu_{v_1}^2 + \nu_{v_2} \nu_{v_1}^2}{Z_{v_1} Z_{v_2}}.
\]

5. Compute \( l = l_1 + i \cdot l_3 + l_1 + l_3 \):

\[

6. Compute \( u' = (x + y + k \cdot x', y + r \cdot k \cdot x') \), \( u' = (x + y + k \cdot x', y + r \cdot k \cdot x') \):

\[
\kappa' = \frac{\psi_{v_1} U_{v_1} + \nu_{v_2} U_{v_2}}{Z_{v_1} Z_{v_2}}, \quad \kappa' = \frac{\psi_{v_1} U_{v_1} + \nu_{v_2} U_{v_2}}{Z_{v_1} Z_{v_2}}.
\]

7. Compute \( v' = (x + y + k \cdot x') \) mod \( \nu_{v_1} \):

\[

8. Compute \( Z_l \) and \( Z_1 \):

\[
Z_l = \frac{y_{v_1}^2}{Z_{v_1}}.
\]

**Table 3. Derivation of expressions in different steps of A.3**
Specify the modifications carried out in A.2, which allowed reducing quantity of field operations.

Step A.2.1. This step is to be omitted due to existence of the same denominator of all coordinates, which allows saving 8 multiplications in \( \text{GF}(p) \).

Step A.2.2. While computation of \( y_1 \) and \( y_2 \), reduction to common denominator of coordinates \( U_1 \) and \( U_2 \) is not required, saves 2 multiplications in \( \text{GF}(p) \).

Moreover, the calculation of resulting \( Z_1 \) and \( Z_2 \) coordinates is also significantly simplified, that helps to save 5 multiplications and 3 squaring operations in \( \text{GF}(p) \).

Step A.2.4. While computation of \( w_1 \) and \( w_2 \), reduction to common denominator of coordinates \( V_1 \) and \( V_2 \) is also not required, saves 2 multiplications in \( \text{GF}(p) \).

Steps A.2.5 and A.2.6 The total number of multiplications remains unchanged, however the number of squaring operations in \( \text{GF}(p) \) decreases by 3, due to the interchange of calculations of coefficients \( l_0 \), \( l_1 \) and \( l_2 \) of polynomial \( l \) on the step A.2.6 (Table 4).

<table>
<thead>
<tr>
<th>#</th>
<th>Expression</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compute resultant ( r ) of ( u_1, u_2, v, \ y = U_{1} - U_{2} )</td>
<td>1S, 4M</td>
</tr>
<tr>
<td>2</td>
<td>Compute almost inverse ( \text{inv} = r \mod n )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Compute ( s = (v_1 - v_2) \mod n )</td>
<td>6M</td>
</tr>
<tr>
<td>4</td>
<td>Precomputations: ( R = r \cdot Z_1 \cdot Z_2 \cdot s_2 = s_1 \cdot s_2 \cdot \text{inv} = y_1 \cdot y_2 \cdot U_{1a} )</td>
<td>7M</td>
</tr>
<tr>
<td>5</td>
<td>Compute ( l = s_{11} )</td>
<td>2M</td>
</tr>
<tr>
<td>6</td>
<td>Compute ( u = (q + h + 2v) \cdot k \cdot y_{11} )</td>
<td>25, 6M</td>
</tr>
<tr>
<td>7</td>
<td>Compute weights: ( Z_1 = u_1, Z_2 = R \cdot Z_1 = s_1 \cdot Z_1 = R \cdot s_2 \cdot s_2 \cdot \text{inv} = y_1 \cdot y_2 \cdot U_{1a} )</td>
<td>1S</td>
</tr>
<tr>
<td>8</td>
<td>Adjust ( Z_1 = s_2 \cdot Z_1 = R \cdot Z_1 = s_2 \cdot Z_1 = R \cdot s_1 \cdot s_2 \cdot \text{inv} = y_1 \cdot y_2 \cdot U_{1a} )</td>
<td>6M</td>
</tr>
</tbody>
</table>

Table 4. Algorithm A.3. Co-Z reduced divisors addition
Step A.2.6. unlike algorithm A.2, the A.3 offers considering multiplier $s_3$, present in each coefficient $l_0$, $l_1$ and $l_2$ of polynomial $l$, when using coefficients $l_i$, $i = 0, 2$ on the steps A.2.7 and A.2.9. This allows us factor out $s_3$, thus saving 3 multiplications in $\mathbb{GF}(p)$ (steps A.4.6-A.4.8).

Consider next the application of proposed algorithm for the mixed divisor addition $D_1 = \begin{bmatrix} U_{11}, U_{10}, V_{11}, V_{10}, Z_1, Z_1^2, Z_2, Z_2^2 \end{bmatrix}$ and $D_2 = \begin{bmatrix} U_{21}, U_{20}, V_{21}, V_{20}, 1, 1, 1, 1 \end{bmatrix}$ (mixed representation). Therefore it is necessary to reduce divisor $D_2$ to common $Z$-coordinate, i.e. $\begin{bmatrix} U_{21} \cdot Z_1^3, U_{20} \cdot Z_1^3, V_{21} \cdot Z_2^3, Z_2, V_{20} \cdot Z_1^3 \cdot Z_2, Z_1, Z_1^2, Z_2, Z_2^2 \end{bmatrix}$, that requires 5 multiplications in $\mathbb{GF}(p)$. Hereinafter the provided algorithm A.4 should be used for addition of (prior formed) divisors with the same $Z$-coordinate.

6. Results

It is to be considered that after computing $D_3 = D_1 + D_2$ one of the items, for example $D_2$, should be transformed so that it has the same $Z$-coordinate as divisor $D_3$. For this purpose, at the step A.3.9, values $Z_{1c}$ and $Z_{2c}$, where $Z_1^2 = s_1^2 = Z_1^3 \cdot Z_{1c}$ and $Z_1^3 Z_2 = s_3^3 \cdot Z_1^2 \cdot Z_{2c}$, i.e. $Z_{1c} = s_1^2 \cdot Z_1^2$ and $Z_{2c} = s_3^3 \cdot r$, this requires 2 additional multiplications and 1 squaring in $\mathbb{GF}(p)$. In other words, reduction of a divisor to unified $Z$-coordinates takes 6 additional multiplications and 1 squaring. Ultimately, for the divisor addition step in A.1 46M+5S (M - multiplication, S - squaring) field operations are required. For the Fibonacci step 41M+5S field operations are required. If we reduce the obtained complexity estimations to the parameters of the curve (Lange, 2002c), we obtain that for the divisor addition step 40M+5S are required and for the Fibonacci step 45M+5S operations are required. In accordance to the computational complexity estimation, the approach described in this paper, is not effective, due to the complexity of mixed addition is 36M+5S (Lange, 2002c). However, the alternative approach to the divisor addition for the scalar multiplication implementation is proposed. Let us draw a computational complexity comparison between scalar multiplication algorithms described in (Kovtun and Zbitnev, 2004) and those suggested in this paper, based on idea (Meloni, 2007).

6.1 Comparison with other method

The results of known and proposed algorithms comparison are set out in Table 5. The algorithm complexity represented in field operations (Table 6).

Assume that $s=0.8M$ and scalar multiplier is an 80-bit integer and refer to the estimations (Kovtun and Zbitnev, 2004; Meloni, 2007) for the estimation of complexity of scalar multiplication algorithms. The results of comparison are set out in Table 7. Computational complexity of Fibonacci-and-add scalar multiplication algorithm is by 23% greater than Binary left-to-right algorithm and by 12.5% greater than Window Fibonacci-and-add.

In other case computational complexity of Fibonacci-and-add scalar multiplication algorithm in weighted coordinates is by 23% greater than Binary left-to-right algorithm in mixed weighted coordinates and by 14.2% greater than Window Fibonacci-and-add.

Weighted coordinates with Co-Z approach are more effective than ordinary projective coordinates with Co-Z approach.
### Table 5. Algorithms and curve parameters

<table>
<thead>
<tr>
<th>Alg. #</th>
<th>Curve description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( h(x) = 0 ) (Harley, 2000)</td>
</tr>
<tr>
<td>2</td>
<td>( h_2 = 1 ) (Lange, 2002a)</td>
</tr>
<tr>
<td>3</td>
<td>( h(x) = 0 ) (Matsuo et al., 2001)</td>
</tr>
<tr>
<td>4</td>
<td>( h(x) = 0, f_4 = 0 ) (Miyamoto et al., 2002)</td>
</tr>
<tr>
<td>5</td>
<td>( h(x) = 0 ) (Takahashi, 2002)</td>
</tr>
<tr>
<td>6</td>
<td>( f_4 = 0, h_2 \neq 0 ) (Lange, 2002a)</td>
</tr>
<tr>
<td>7</td>
<td>( \text{deg}(h) = 2, h_i \in F_2 ) (Lange, 2002b)</td>
</tr>
<tr>
<td>8</td>
<td>( \text{deg}(h) = 2, h_i \in F_2 ) (Kovtun and Zbitnev, 2004)</td>
</tr>
<tr>
<td>9</td>
<td>( h(x) = 0, f_4 = 0 ) (Kovtun, 2006)</td>
</tr>
<tr>
<td>10</td>
<td>( h(x) = 0, f_4 = 0 ) (Lange, 2002c)</td>
</tr>
<tr>
<td>11</td>
<td>( \text{deg}(h) = 2, h_i \in F_2 ) (Kovtun, 2010)</td>
</tr>
<tr>
<td>12</td>
<td>( h(x) = 0, f_4 \neq 0 ) [proposed]</td>
</tr>
<tr>
<td>13</td>
<td>( h(x) = 0, f_4 = 0 ) [proposed]</td>
</tr>
</tbody>
</table>

### Table 6. Computational complexity of group law in Jacobian of genus 2 HEC over \( GF(p) \)

<table>
<thead>
<tr>
<th>#</th>
<th>Addition</th>
<th>Doubling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>General</td>
<td>Mixed</td>
</tr>
<tr>
<td></td>
<td>( ^{-1} )</td>
<td>( ^{-1} )</td>
</tr>
<tr>
<td>Affine coordinates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>Projective coordinates</td>
<td>( [U_1, U_0, V_1, V_0, Z] )</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>Weighted coordinates</td>
<td>( [U_1, U_0, V_1, V_0, Z_1, Z_2, Z_1^2, Z_2^2] )</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>47</td>
</tr>
<tr>
<td>Co-Z projective coordinates</td>
<td>( [U_1, U_0, V_1, V_0, Z] )</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>Co-Z weighted coordinates</td>
<td>( [U_1, U_0, V_1, V_0, Z_1, Z_2, Z_1^2, Z_2^2] )</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>46</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
<td>45</td>
</tr>
</tbody>
</table>
### Table 7. Computational Complexity Of Scalar Multiplication Algorithms

<table>
<thead>
<tr>
<th>#</th>
<th>Scalar multiplication algorithm</th>
<th>Cost, $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Addition in mixed projective coordinates (Kovtun and Zbitnev, 2004)</td>
<td>5192</td>
</tr>
<tr>
<td>2</td>
<td>Binary (left-to-right)</td>
<td>4629</td>
</tr>
<tr>
<td>3</td>
<td>$w$-NAF, $w = 4$</td>
<td>4349</td>
</tr>
<tr>
<td>7</td>
<td>Fibonacci-and-add</td>
<td>6773</td>
</tr>
<tr>
<td>8</td>
<td>Window Fibonacci-and-add</td>
<td>5970</td>
</tr>
<tr>
<td>4</td>
<td>Addition with Co-Z method in projective coordinates (Kovtun, 2010)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>NAF</td>
<td>4570</td>
</tr>
<tr>
<td>6</td>
<td>$w$-NAF, $w = 4$</td>
<td>4307</td>
</tr>
<tr>
<td>7</td>
<td>Fibonacci-and-add</td>
<td>6629</td>
</tr>
<tr>
<td>8</td>
<td>Window Fibonacci-and-add</td>
<td>5829</td>
</tr>
</tbody>
</table>

6.2 Another aspect of using the proposed approach

Yet another aspect of using a biometric authentication on based Co-Z approach to divisor scalar multiplication in Jacobian of genus 2 hyperelliptic curves over fields with odd characteristic in weighted coordinates is using it in the fight against Cyber terrorism. Using this approach and biometric authentication will significantly reduce financial losses of enterprises.

Primary solutions of the given problem are confirmed also with the data presented on figure 1 on which (Kavun, 2007) finance indexations of the put damage for some countries are shown. Apparently from the presented statistics, the state infrastructure is more developed; the larger it receives damage from cyber criminality.

![Fig. 1. Damage from cyber criminalities](http://www.intechopen.com)

During an epoch of the world economic crisis and modern transformations of economy of different countries the aspect of economic security becomes even more urgent. For example, the tendency of increase in crime in sphere of cyber terrorism, having an economic (money) basis is shown in figure 2.
Fig. 2. Statistics of cyber criminality rate

From the presented data which has been average for some years (Kavun, 2007), it can be seen, that on the average the amount of incidents in the sphere of cyber criminality over a year increased by 50%. It testifies to negative tendencies of cyber criminality development in the world.

Outcomes of the conducted analysis are presented in figure 3 from which it appears, that main sources of illegal operations are the USA and China which cover 50% of all threats (Kavun, 2009).

7. Conclusion

In the paper, a new algorithm of weight 2 divisor addition with identical (shared) Z-coordinates (by the Co-Z approach) has been proposed, which requires more $\text{GF}(p)$ operations then algorithm (Kovtun and Zbitnev, 2004), however it allows a decrease in computational complexity of Fibonacci-and-add scalar multiplication algorithm while approaching to the Binary left-to-right algorithm.

Biometrics are not secrets and are not revocable (Schneier, 1999) while revocability and secrecy have been critical requirements of conventional cryptosystem design, one then wonders whether it is possible to design a secure authentication system from the system components which in themselves are neither secrets nor revocable—for example, whether the methods of ensuring liveness of biometric identifiers and challenge-response schemes (Maltoni et al., 2003) obviate fraudulent insertion of “stolen” biometric identifiers. Is it possible to nontrivially combine knowledge and biometric identifiers to arrive at key generation/release mechanisms where biometric identifiers are necessary but not sufficient for cryptographic key generation/release? Is it possible to require multiple biometrics to make it increasingly difficult for the attacker to fraudulently insert multiple biometrics into the system? Is it possible to make it unnecessary to revoke/update the cryptographic key in the event of a “stolen biometric”? Exploring challenges in designing such systems is a promising (yet neglected) avenue of research. When cryptobiometric systems eventually come into practical existence, there is a danger that biometric components may be used as an irrefutable proof of existence of a particular subject at a particular time and place. Mere incorporation of biometrics into a system does not in itself constitute a proof of identity. We need to understand how these foolproof guarantees can be theoretically proved in a deployed cryptosystem and how to institute due processes that will provide both
technological and sociological freedom to challenge the premises on which nonrepudiability is ascertained.

Genus 3 curves might save about a third in key lengths and so 180-bit ECC (which is beyond the usual range, as given in standards) is equivalent to 60-bit HCC, which can be implemented on a fast 64-bit computer. In practice, 160-bit ECC is typically used and so there is even room for added security in case of computational speed-ups in attacks. Stein, in particular, has pointed out how the use of “real” forms of hyperelliptic curves (i.e., with infrastructure) allows considerable speed-ups in implementation in some cases.

On the other hand, Gaudry (2000), showed that hyperelliptic curves of genus bigger than or equal to 5 and possibly 4 are less secure than hyperelliptic curves of genus g<4 (or 5) (ICCIT Biometrics Research Group, 2005). That means that the key-per-bit-strength of hyperelliptic curves of genus 2 and 3 is the same as for elliptic curves, and thus far better than conventional systems based on discrete logs or integer factoring. In fact, genus 2 is particularly interesting because the arithmetic appears to be only minimally slower than elliptic curve arithmetic and the bit size of the underlying finite field is half as big as for elliptic curves having the same security level. To our knowledge nobody has performed a down-to-earth implementation of genus 2 hyperelliptic curves.

We are considering hardware implementations of hyperelliptic curves of genus 2 and 3. Among the currently available hyperelliptic curves there are, for instance, Koblitz curves and curves constructed by a complex multiplication method (CM-method). Both are natural extension of ideas from elliptic curves.

8. References


In the recent years, a number of recognition and authentication systems based on biometric measurements have been proposed. Algorithms and sensors have been developed to acquire and process many different biometric traits. Moreover, the biometric technology is being used in novel ways, with potential commercial and practical implications to our daily activities. The key objective of the book is to provide a collection of comprehensive references on some recent theoretical development as well as novel applications in biometrics. The topics covered in this book reflect well both aspects of development. They include biometric sample quality, privacy preserving and cancellable biometrics, contactless biometrics, novel and unconventional biometrics, and the technical challenges in implementing the technology in portable devices. The book consists of 15 chapters. It is divided into four sections, namely, biometric applications on mobile platforms, cancellable biometrics, biometric encryption, and other applications. The book was reviewed by editors Dr. Jucheng Yang and Dr. Norman Poh. We deeply appreciate the efforts of our guest editors: Dr. Girija Chetty, Dr. Loris Nanni, Dr. Jianjiang Feng, Dr. Dongsun Park and Dr. Sook Yoon, as well as a number of anonymous reviewers.
© 2011 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.