Software Implementation of Tate Pairing over GF(2^m)

G. Bertoni[1], L. Breveglieri[2], P. Fragneto[1], G. Pelosi[2] and L. Sportiello[1]
ST Microelectronics[1], Politecnico di Milano[2]
Via Olivetti, Agrate B., Milano, Italy - Piazza L. Da Vinci, Milano, Italy
{guido.bertoni, pasqualina.fragneto, luigi.sportiello}@st.com - {breveglieri, pelosi}@elet.polimi.it

Abstract

Recently, the interest about the Tate pairing over binary fields has decreased due to the existence of efficient attacks to the discrete logarithm problem in the subgroups of such fields. We show that the choice of fields of large size to make these attacks infeasible does not lead to a degradation of the computation performance of the pairing. We describe and evaluate by simulation an implementation of the Tate pairing that allows to achieve good timing results, comparable with those reported in the literature but with a higher level of security.

1. Introduction

Nowadays there are many pairing based cryptosystems that are gaining an increasing interest. Most of them are identity-based cryptosystems [18], such as the encryption scheme of Boneh-Franklin [4], the key-agreement protocol of Smart [19], the digital signature scheme of Hess [10] and the sign-cryption scheme of McCullagh-Barreto [13]. Other examples are the tripartite Diffie-Hellmann protocol of Joux [11] and the short signature scheme of Boneh *et al.* [5].

Pairing computation is the most time consuming task in such cryptosystems. Of the Weil [14] and Tate [3] pairing, the latter is theoretically more efficient than the former [7]. In order to obtain an appealing cryptographic application it is important to develop an optimized implementation of the Tate pairing without affecting its security level.

In this paper we present some results about an implementation of the Tate pairing over supersingular curves in characteristic 2, with a set of security conditions stronger than those presented in the literature [7]. We show and discuss a combination of techniques to provide a comparable timing despite the higher security level we achieve.

It is widely known that binary extension fields are particular indicated for embedded devices and for hardware implementations thanks to their properties, but it is even a general belief that discrete logarithm problem is simpler in $GF(2^n)$ than $GF(p)$ with comparable multiplicative group order. This consideration has discouraged the research in the field of Tate pairing over $GF(2^n)$. We would like to demonstrate that using a field of 1800 bits, which should grant a security at least comparable to RSA 1024, the performances obtained on a Pentium III can be considered satisfactory. This result is important for embedded system since most of the time the secure communication is composed by two devices, a network server generally based on PC like architecture and an embedded system.

Our results demonstrate that the server side performances does not suffer of high degradation due to the increase of field size. This should open the research on the embedded system side where $GF(2^n)$ will demonstrate all its potential.

The paper is organized as follows. Section 2 provides background notions about Tate pairing. Section 3 summarizes the security of pairing based cryptosystems built on supersingular elliptic curves over $\mathbb{F}_{2^m}$ and shows some techniques to improve the computation performance of Tate pairing algorithms. Section 4 presents timing results and finally Section 5 gives conclusions. In Appendix A we present some formulas for the computation of the Tate pairing when elliptic curve points are represented in Affine coordinates [9].

2. Preliminaries on Tate Pairing

For cryptographic purposes it is assumed a definition of Tate pairing as that showed in [7, 3]. Let E be a supersingular elliptic curve defined over a finite field $\mathbb{F}_q$ and consider the l-torsion points subgroup, with l prime and $l \mid \#E(\mathbb{F}_q)$; let k be the minimum integer such that all the l-torsion points of the curves have coefficients in the extension field $\mathbb{F}_{q^k}$, with $l \mid q^k - 1$. Given two points $P, Q \in E(\mathbb{F}_q)[l]$, the Tate pairing is defined as a rational function f_P over the points of the curve such that its divisor is $(f_P) = l(P) - l(O_E)$, and such that it maps curve points to the l-th roots of the unity subgroup $\mu_l \subset \mathbb{F}^*_{q^k}$.

$$\langle .,. \rangle : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_q)[l] \rightarrow \mu_l;$$

3-9810801-0-6/DATE06 © 2006 EDAA

$$\langle P,Q\rangle = f_P(\phi(Q))^{(q^k-1)/l}$$

In order to have a non-trivial value ($\langle P,Q\rangle \neq 1$) it is necessary to pair two linearly independent points. For this purpose, in the definition of the pairing it was used a distortion map $\phi(\cdot)$ [20] that exists only for supersingular elliptic curves. The distortion map grants that points P, $\phi(Q)$ are linearly independent so that they do not map to 1 when used to evaluate the pairing function. The properties that make the Tate pairing useful to define a number of cryptographic primitives are the following:

Well-defined:
$\langle O_E,Q\rangle = 1\,\forall Q \in E(\mathbb{F}_q)[l]$; $\langle P,O_E\rangle = 1\,\forall P \in E(\mathbb{F}_q)[l]$.

Non-degeneracy:
$\forall P \in E(\mathbb{F}_q)[l]\backslash\{O_E\}$, $\exists Q \in E(\mathbb{F}_q)[l]\, s.t.\, \langle P,Q\rangle \neq 1$.

Bilinearity:
$\forall P,Q,R \in E(\mathbb{F}_q)[l]$, $\langle P+R,Q\rangle = \langle P,Q\rangle \cdot \langle R,Q\rangle$ and $\langle P,Q+R\rangle = \langle P,Q\rangle \cdot \langle P,R\rangle$.

As showed by Algorithm 2.1, a method to compute the pairing function is given by Miller's algorithm, which uses a double-and-add strategy with some extra computation due to the construction and evaluation of the rational function given by the straight line $g_{U,V} : l_1 y + l_2 x + l_3 = 0$ with $l_i \in \mathbb{F}_q$, $i = 1,2,3$ passing through the elliptic curve points U and V. Algorithm 2.1 is one of the possibility for calculating the Tate Pairing, recently it has been publish a variant called *etha* Pairing [2]. The two solutions are comparable in term of number of finite field operations.

Algorithm 2.1: Miller's algorithm to compute the Tate pairing [3].

Input: $q, k, t = \lceil \log_2(l) \rceil$, $l = (l_{t-1},\ldots,l_0)_2$; $P,Q \in E(\mathbb{F}_q)[l]$

Output: $\langle P,Q\rangle = f_P(\phi(Q))^{\frac{q^k-1}{l}} \in \mathbb{F}^*_{q^k}$

```
1  begin
2      f ← 1
3      V ← P
4      for i ← t − 2 down to 0 do
5          f ← f^2 · g_{V,V}(φ(Q))
6          V ← 2V
7          if l_i = 1 and i ≠ 0 then
8              f ← f · g_{V,P}(φ(Q))
9              V ← V + P
10         endif
11     endfor
12     f ← f^{(q^k−1)/l}
13     return f
14 end
```

3. Efficient and Secure Implementation

In this paper we consider the Tate pairing defined over the following supersingular elliptic curves:

$$E_b : y^2 + y = x^3 + x + b \ \text{ over } \mathbb{F}_{2^m}$$

with $k = 4$, $b \in \{0,1\}$, m odd whose orders are:

$$\begin{aligned} \#E_b(\mathbb{F}_{2^m}) &= 2^m + 1 + (-1)^b\sqrt{2^{m+1}} \ \ m \equiv 1,7 \pmod 8 \\ \#E_b(\mathbb{F}_{2^m}) &= 2^m + 1 - (-1)^b\sqrt{2^{m+1}} \ \ m \equiv 3,5 \pmod 8 \end{aligned} \quad (1)$$

and the distortion map [3] is given in Table 1:

Map	Parameters
$\phi : (x,y) \mapsto (x+s^2, y+sx+t)$	$s^4 + s = 0$ $t^2 + t + s^6 + s^2 = 0$ with $s,t \in \mathbb{F}_{2^{4m}}$

Table 1. Distortion map for curves $E_b(\mathbb{F}_{2^m})$.

3.1. Security Assumptions

In this section we summarize the security conditions necessary to set up a pairing over the curve $E_b(\mathbb{F}_{2^m})$. It is important to consider possible attacks to the Elliptic Curve Discrete Logarithm Problem (ECDLP) and make them infeasible. For our explanation consider the following ECDLP instance: $U,V \in E_b(\mathbb{F}_{2^m})[l]$, with $U = r \cdot V$ and $r \in \mathbb{Z}^*_l$. The Pohlig-Hellman algorithm [16] reduces the computation of r to the calculus of r modulo each prime factor of l and then, using the CRT, recovers the value of r. Thus, it is customary to choose l prime. Looking both at the l-torsion points subgroup on the curves and at the subgroup of the l-th roots of unity in $\mathbb{F}^*_{2^{4m}}$, an algorithm to solve the discrete logarithm problem is the Pollard-ρ [17], which makes use of $\sqrt{\pi l}/2$ group operations. To make this attack infeasible, it is advisable to take the integer l as long as at least 160 bits [1]. Moreover, according to Gaudry et al. [8], in the case of curves defined over binary fields, the ECDLP can be solved even faster if the value of m admits a small factor, thus m should be chosen to be a prime number. Obviously, in order to avoid that the subgroup of the l-th roots of unity is embedded in some subfield of $\mathbb{F}_{2^{4m}}$, the following condition must be fulfilled [1]: $\forall d \mid 4m$, $l \nmid 2^d - 1$, with $d < 4m$. The case of interest for us complies with this condition. Finally, the existence of attacks in the multiplicative subgroup of $\mathbb{F}^*_{2^{4m}}$, as for instance the index-calculus [6], makes it necessary [12, 15] to adopt an extension degree $4m$ in the interval [1500, 2000]. This should give a security comparable at least with a 1024-RSA cryptosystem.

3.2. Encoding of the Subgroup Order l

In order to simplify the computation of Algorithm 2.1 it is possible to represent the subgroup order l in NAF [9]. In this case, when a digit l_i of the representation of l is equal to ± 1 and $i \neq 0$ it is necessary to compute the operations in the *if* branch of such algorithm (lines between **7** and **10**), but when $l_i = -1$ it is necessary to consider the point $-P$ instead of P. It is possible to prove the correctness of this idea by Miller's theorem and by the fact that denominators such as $g_{P,-P}(\phi(Q))$ can be discarded thanks to the final exponentiation [3].

3.3. Extension Field Arithmetic

Regarding the way to implement the arithmetic operations among elements of the extension fields $\mathbb{F}_{2^{4m}}$, we chose to follow the results showed in [7]. The extension field is isomorphic to a tower of 2 quadratic extensions of $\mathbb{F}_{2^m}$ such that $\mathbb{F}_{2^{4m}} \cong \mathbb{F}_{\left((2^m)^2\right)^2}$.
Precisely, let $F = \mathbb{F}_{2^m}$, then $F_1 = F[x]/(x^2+x+1) \cong \mathbb{F}_{2^{2m}}$ and $\mathbb{F}_{\left((2^m)^2\right)^2} = F_1[y]/(y^2+(x+1)y+1) \cong \mathbb{F}_{2^{4m}}$. A generic element of $\mathbb{F}_{2^{4m}}$ is represented as $(dx+c)y+(bx+a)$, where $a,b,c,d \in \mathbb{F}_{2^m}$. This solution provides a computational cost of 9 multiplications or of 4 squarings in the base field $\mathbb{F}_{2^m}$, for one multiplication or for one squaring in $\mathbb{F}_{2^{4m}}$, respectively.

3.4. Elliptic Curve Point Representation

It is important to identify the coordinates system to represent the elliptic curve points P and V in Algorithm 2.1, requiring as few operations in $\mathbb{F}_{2^m}$ as possible. Notice that in the *for* loop of such algorithm two operations are performed simultaneously: "double-and-add" of the point P and the iterative evaluation of $f_P(\phi(Q))$. We merge these two operations in order to minimize the number of requested operations in the base field $\mathbb{F}_{2^m}$. We consider [9] Affine coordinates (x,y) and three types of Projective coordinates (X,Y,Z): Standard, where the corresponding affine point is $(X/Z, Y/Z)$; Jacobian, corresponding to $(X/Z^2, Y/Z^3)$; and Lopez-Dahab, corresponding to $(X/Z, Y/Z^2)$. Furthermore, when projective coordinates are used, the point $P(X_P, Y_P, Z_P)$ always has coordinate $Z_P = 1$, because this choice allows to avoid a few operations in $\mathbb{F}_{2^m}$.

In order to show results, we conceive the *for* loop divided into two parts: a "double step" and an "add step". The costs of all representations, expressed in terms of the operations of $\mathbb{F}_{2^m}$, are summarized in Table 2 (I, M, S denote respectively 1 inversion, 1 multiplication and 1 squaring in the base field). The coordinate systems Affine V1 and Affine V2 perform the doubling of a point in a different way (explicit formulas are presented in Appendix A). It is also important to observe that using the specific distortion map presented in Table 1, the coordinates $x_{\phi(Q)}$ and $y_{\phi(Q)}$ of $\phi(Q) \in E_b(\mathbb{F}_{\left((2^m)^2\right)^2})[l]$ exhibit the following structure:

$$x_{\phi(Q)} = (0x+0)y + (1x+a) \text{ with } a \in \mathbb{F}_{2^m}$$

$$y_{\phi(Q)} = (1x+0)y + (bx+c) \text{ with } b,c \in \mathbb{F}_{2^m}$$

There are coefficients equal to 0 and 1 that allow to simplify some calculations in Miller's algorithm. For example, the multiplication of elements of $\mathbb{F}_{2^m}$ by $x_{\phi(Q)}$ or $y_{\phi(Q)}$ needs respectively one and two multiplications. The results presented in Table 2 take into consideration these observations.

Coordinates system	Double step	Add step
Affine V1	8M+6S	1I+9M+1S
Affine V2	7M+8S	1I+9M+1S
Projective $(X/Z, Y/Z)$	15M+10S	20M+3S
Jacobian $(X/Z^2, Y/Z^3)$	16M+12S	20M+5S
Lopez-Dahab $(X/Z, Y/Z^2)$	16M+10S	19M+5S

Table 2. Cost of the *for* loop in Miller's algorithm with different point representations.

Notice that it is necessary to choose the best representation according to the specific implementation. Precisely, knowing the computation times of the operations in the base field it is possible to calculate the ratios I/M and S/M and therefore express the costs of the "double" and "add" steps of Table 2 in terms of the number of multiplications in $\mathbb{F}_{2^m}$. Then, knowing the number of non-zero digits in the representation adopted for the integer l (e.g. NAF or else), the number of required "add" steps is automatically determined. With all this information it is possible to express the cost of each representation in terms of multiplications in the base field and thus to identify the most convenient representation technique.

Notice that it is possible to limit the number of "add" steps choosing a subgroup order and its representation such that l presents a low Hamming weight. In these cases the "add" steps are performed rarely, thus the computational cost of the Algorithm 2.1 becomes dominated by the "double" steps. This justifies the choice of Affine coordinates versus Projective ones, since Projectives require almost a number of multiplications equal to 15 versus the 7 required by Affine. In most practical implementations the following inequality S/M< 0.5 holds, and it can be easily seen

from Table 2 that the coordinate system Affine V2 is always preferable to the coordinate system Affine V1.

It is possible that the order of the torsion group can be chosen equal to the order of the elliptic curve, say $l = E(\mathbb{F}_{2^m})$, and as far as the value of m is concerned it ought to be a prime such that the Hamming weight of the order l encoded in Non Adjacent Form (NAF) is minimized.

3.5. Final Exponentiation

In the case of our interest, at the end of the computation of the Tate pairing there is the exponentiation $f^{(2^{4m}-1)/l}$, with $f \in \mathbb{F}_{\left((2^m)^2\right)^2}$. When the number of curve points $\#E_b(\mathbb{F}_{2^m})$ is a prime, the order l of the l-torsion subgroup is equal to $\#E_b(\mathbb{F}_{2^m})$ and in this circumstance it is possible to have also a fast final exponentiation. The following factorization is valid:
$(2^{4m}-1) = (2^{2m}-1)(2^m+1-\sqrt{2^{m+1}})(2^m+1+\sqrt{2^{m+1}})$, hence $\#E_b(\mathbb{F}_{2^m}) \mid (2^{4m}-1)$ (see (1) for $\#E_b(\mathbb{F}_{2^m})$ values).

In these cases the exponent $(2^{4m}-1)/l$ provide a NAF representation with only 6 digits equal to ± 1. Notice that using NAF it is requested the knowledge of f^{-1} in exponentiation, but such inversion can be performed only once at the beginning of the exponentiation phase. In addition, observing the representation of such exponents, it is trivial to verify that window techniques for exponentiation do not give any speed up and therefore this efficient solution does not need memory to store pre-computed values. We point out that it is possible to execute the *for* loop of Algorithm 2.1 considering the number of curve points $\#E_b(\mathbb{F}_{2^m})$ instead of the subgroup order l also when $l < \#E_b(\mathbb{F}_{2^m})$ (see [7]). Also in these cases the exponent is $(2^{4m}-1)/\#E_b(\mathbb{F}_{2^m})$ and the solution explained above is still applicable.

4. Timing Results

We have implemented the Tate pairing using the improvements presented above and choosing its parameters according to the security conditions presented in Section 3.1. In fact we have selected the curve $E_1(\mathbb{F}_{2^m})$ with $m = 457$ and such that $l = \#E_1(\mathbb{F}_{2^{457}})$ is also a prime of 457 bits. We have used $f(x) = x^{457} + x^{16} + 1$ as irreducible polynomial to generate the field $\mathbb{F}_{2^{457}}$. We have developed a software library implementing all the operations required for the computation of the Tate pairing; all the code was written in C, compiled with Microsoft Visual C++ V6.0 and simulations were performed on a Pentium III working at 1 GHz. With our implementation we obtain the ratios I/M$=52.36$ and S/M$=0.12$, while adopting the representation in NAF of the subgroup order l, it is easy to see that only one "add" step is required; from these considerations and according to Table 2, Affine V2 was found to be the best coordinate system for the representation of elliptic curve points. In Table 3 we compare our result to those by Galbraith *et al.* [7] and by Barreto *et al.* [3] over binary fields (also their simulations are on Pentium III 1 GHz).

Finite Field $\mathbb{F}_{2^m}$	**Timing**
$\mathbb{F}_{2^{457}}$	30.61
$\mathbb{F}_{2^{241}}$ in [7]	32.5
$\mathbb{F}_{2^{283}}$ in [7]	57.19
$\mathbb{F}_{2^{271}}$ in [3]	23

Table 3. Comparison of the times of the computation of the Tate pairing (in ms) over binary fields, on a P III at 1 GHz.

Notice that all the timings are comparable though our Tate pairing algorithm is defined over a finite field of about double size with respect to [3, 7]. This could be imputable to the several optimizations that we apply altogether; furthermore papers [3, 7] do not specify anything about the representation adopted both for the subgroup order l and for the curve points. In addition, thanks to the greater size of m (i.e. the field size) we provide a higher level of security. In fact, an efficient way to attack pairing based cryptosystems over binary fields is given by the index-calculus attack for the discrete logarithm problem in $\mathbb{F}^*_{2^{4m}}[l]$. But we have shown that it is possible to double the field size, thus reducing the effectiveness of such an attack, while maintaining the same time performance with respect to literature solutions using smaller fields.

5. Conclusions

This paper summarizes a methodology to select parameters for implementing the Tate pairing using supersingular curves over binary fields, $E_b(\mathbf{F}_{2^m})$. We have reviewed some criteria to select the most convenient coordinate system to represent the points of the curve; how to manage the l-torsion subgroup order l efficiently, to minimize the number of operations in the inner loop of Miller's Algorithm and, at the same time, to reduce the computational cost of the final exponentiation; we observed that with a fixed class of curves and an appropriate field representation, the corresponding distortion map exhibits a form yielding some savings in terms of arithmetic operations among the elements of the base field.

It is commonly accepted that a cryptosystem that employs a Tate pairing over a binary field $\mathbb{F}_{2^m}$ with an embedding degree $k = 4$ must use a field size such that $4m \approx 1000$ to grant a security level equivalent to 1024-RSA. The implementation described in this paper shows that the use of

a field size greater than the examples exhibited in the literature does not penalize the timing performance. The results contained in this paper prove that our implementation has a computational time comparable to the most efficient implementations reported in the literature [3, 7], that however are less secure than ours because they use substantially smaller base fields. In light of this, we point out that also the Tate pairing over binary fields may be an effective solution for implementing paring based cryptosystems, particularly for embedded system where large prime fields are difficultly managed.

References

[1] IEEE Standard Specification for Public-Key Cryptography. Technical report, 2000.

[2] P. Barreto, S. Galbraith, C. hEigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Cryptology ePrint Archives, http://eprint.iacr.org*, 2004.

[3] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. *Advances in Cryptology - CRYPTO 2002, LNCS*, 2442:354–368, 2002.

[4] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *Advances in Cryptology - CRYPTO 2001, LNCS*, 2139:213–229, 2001.

[5] D. Boneh, B. Lynnn, and H. Shacham. Short signature from the Weil pairing. *Proceedings of Asiacrypt 2001, LNCS*, 2248:514–532, 2001.

[6] D. Coppersmith. Evaluating logarithms in GF(2^n). *Annual* ACM *Symposium on Theory of Computing '84, ACM Press*, pages 201–207, 1984.

[7] S. Galbraith, K. Harrison, and D.Soldera. Implementing the Tate pairing. *Algorithm Number Theory Symposium - ANTS V, LNCS*, 2369:324–337, 2002.

[8] P. Gaudry, F. Hess, and N. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *Journal of Cryptology*, 15:19–46, 2002.

[9] D. Hankerson, J. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. *Cryptographic Hardware and Embedded Systems - CHES, LNCS*, 1965:1–24, 2000.

[10] F. Hess. Exponent groups signature schemes and efficient identity based signature schemes based on pairings. *Cryptology ePrint Archives http://eprint.iacr.org*, 2002.

[11] A. Joux. A one round protocol for tripartite diffie-hellman. *Algorithm Number Theory Symposium - ANTS IV, LNCS*, 1838:385–394, 2000.

[12] A. K. Lenstra. *Contributions Paper on Cryptographic Key Lengths to the Information Security Management Handbook*. Harold F. Tipton, 2004.

[13] N. McCullagh and P. Barreto. Efficient and forward-secure identity-based signcryption. *Cryptology ePrint Archives http://eprint.iacr.org*, 2004.

[14] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transaction on Information Theory*, 39:1639–1646, 1993.

[15] A. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. *Advances in Cryptology* - EUROCRYPT*'84, LNCS*, 209:224–314, 1985.

[16] S. Pohlig and M. Hellman. An improved algorithm for computing logarithm over GF(p) and its cryptographic significance. *IEEE Transaction on Information Theory*, 24:106–110, 1978.

[17] J. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978.

[18] A. Shamir. Identity-based cryptosystems and signature schemes. *Advances in Cryptology - CRYPTO '84, LNCS*, 196:47–53, 1985.

[19] N. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Cryptology ePrint Archives http://eprint.iacr.org*, 2001.

[20] E. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *Advances in Cryptology - Eurocrypt 2001, LNCS*, 2045:195–210, 2001.

A. Operations in Miller's Algorithm using Affine Coordinates

In Algorithm 2.1 it is performed the "double and add" of point $P \in E_b(F_{2^m})$ and the iterative evaluation of $f_P(\phi(Q))$.

Merging these two operations and considering Affine coordinates to represent elliptic curve points we obtain the following formulas, where $V, Q \in E_b(F_{2^m})$ and $\phi(Q) \in E_b(F_{2^{4m}})$. For the "double" step:

$$\begin{cases} \lambda = x_V^2 + 1 \\ x_{2V} = \lambda^2 \\ y_{2V} = \lambda(x_{2V} + x_V) + y_V + 1 & \text{(V1)} \\ y_{2V} = y_V^4 + x_V^4 & \text{(V2)} \\ g_{V,V}(\phi(Q)) = \lambda(x_{\phi(Q)} + x_V) + (y_{\phi(Q)} + y_V) \end{cases}$$

Notice that there are no inversions in $\mathbb{F}_{2^m}$ for the "double" step in the Affine system. For the "add" step it holds:

$$\begin{cases} \lambda = \frac{y_V + y_P}{x_V + x_P} \\ x_{V+P} = \lambda^2 + x_V + x_P \\ y_{V+P} = \lambda(x_{V+P} + x_P) + y_P + 1 \\ g_{V,P}(\phi(Q)) = \lambda(x_{\phi(Q)} + x_P) + (y_{\phi(Q)} + y_P) \end{cases}$$