Coordinate Blinding over Large Prime Fields

Michael Tunstall[1] and Marc Joye[2]

[1] Department of Computer Science, University of Bristol
Merchant Venturers Building, Woodland Road
Bristol BS8 1UB, United Kingdom
tunstall@cs.bris.ac.uk

[2] Technicolor, Security & Content Protection Labs
1 avenue de Belle Fontaine, 35576 Cesson-Sévigné Cedex, France
marc.joye@technicolor.com

Abstract. In this paper we propose a multiplicative blinding scheme for protecting implementations of a scalar multiplication over elliptic curves. Specifically, this blinding method applies to elliptic curves in the short Weierstraß form over large prime fields. The described countermeasure is shown to be a generalization of the use of random curve isomorphisms to prevent side-channel analysis, and our best configuration of this countermeasure is shown to be equivalent to the use of random curve isomorphisms. Furthermore, we describe how this countermeasure, and therefore random curve isomorphisms, can be efficiently implemented using Montgomery multiplication.

Keywords: Elliptic curve cryptography, side-channel analysis, countermeasures.

1 Introduction

Side-channel analysis can be used to try and derive unknown information used in cryptographic algorithms, such as cryptographic keys. The first side-channel described in the literature was based on the total time taken to compute a cryptographic algorithm [18]. Preventing this attack is well understood, as one just requires a regular algorithm to prevent any side-channel leakage.

Another side-channel that has been described in the literature is based on the observation that the power consumption of a microprocessor is dependent on the instruction being executed and on any data being manipulated [6, 19]. An attacker can, therefore, observe where functions, and sequences of functions, occur in a power consumption trace. This allows information on cryptographic keys to be determined if the sequence of instructions is affected by the value of the key. An attacker can also determine if a value being manipulated by a microprocessor can be correctly predicted by computing the correlation between a set of predictions and the instantaneous power consumption. This allows information on cryptographic keys to be determined since one can verify a hypothetical set of values that occur after being combined with a key.

It was later observed that the electromagnetic field around a microprocessor also has this property [12, 23]. Preventing an attacker from being able to use this information is more complex, as all the intermediate states of an algorithm need to be masked with some random value [11, 19]. When implementing a block cipher this can be implemented by modifying the algorithm such that it operates in this manner by modifying each function [3].

When a public-key cryptographic algorithm, such as RSA [24], is implemented countermeasures are typically based on the structure of the entire function. For example, when generating a signature σ from a message m using RSA, one computes $\sigma = \mu(m)^d \bmod N$, where d is the private key and μ is an appropriate padding function. That is, a standard exponentiation algorithm in $(\mathbb{Z}/N\mathbb{Z})^*$. This can be changed such that the intermediate states of the calculation cannot be predicted by computing $\sigma = [(\mu(m) + r_1\, N)^{r_2\, \phi(N)+d} \bmod r_3\, N] \bmod N$, where ϕ is Euler's totient function and r_i, for $i \in \{1, 2, 3\}$, are (small) random values. However, one would not want to directly apply this countermeasure to implementations of elliptic curve cryptosystems using prime fields. Increasing the size of the modulus used in RSA has a relatively small impact on the overall execution time. The impact on elliptic curves will be larger since the prime values used in the field arithmetic are much smaller.

Many different countermeasures for preventing side-channel analysis of elliptic curve cryptographic algorithms have been proposed in the literature. In this paper we describe a multiplicative blinding method for elliptic curve cryptographic algorithms over prime fields that is a generalization of previously proposed methods, and describe how it can be efficiently implemented.

The rest of this paper is organized as follows. In the next section, we introduce some background on elliptic curves and review some countermeasures against side-channel analysis. Section 3 is the core of our paper. We define a new addition using blinded coordinates. Detailed formulæ are provided for homogeneous and Jacobian representations. In Section 4 we describe how one could implement the proposed countermeasure. In Section 5 we discuss some further security considerations that one would need to take into account when implementing the proposed countermeasure. Finally, we conclude in Section 6.

2 Preliminaries

2.1 Elliptic Curves

Let $\mathbb{F}_q$ be a finite field. An elliptic curve $\mathcal{E}$ over $\mathbb{F}_q$ consists of points (x, y), with x, y in $\mathbb{F}_q$, that satisfy the full Weierstraß equation

$$\mathcal{E} : y^2 + a_1\, x\, y + a_3\, y = x^3 + a_2\, x^2 + a_4\, x + a_6$$

with $a_i \in \mathbb{F}_q$ $(1 \le i \le 6)$, and the point at infinity denoted $\boldsymbol{O}$. The set $\mathcal{E}(\mathbb{F}_q)$ is defined as

$$\mathcal{E}(\mathbb{F}_q) = \{(x, y) \in \mathcal{E} \mid x, y \in \mathbb{F}_q\} \cup \{\boldsymbol{O}\}\,,$$

where $\mathcal{E}(\mathbb{F}_q)$ forms an Abelian group under the chord-and-tangent rule and $\boldsymbol{O}$ is the identity element.

The addition of two points $\boldsymbol{P} = (x_1, y_1)$ and $\boldsymbol{Q} = (x_2, y_2)$ with $\boldsymbol{P} \neq -\boldsymbol{Q}$ is given by $\boldsymbol{P} + \boldsymbol{Q} = (x_3, y_3)$ where

$$x_3 = \lambda^2 + a_1\,\lambda - a_2 - x_1 - x_2, \quad y_3 = (x_1 - x_3)\lambda - y_1 - a_1\,x_3 - a_3 \tag{1}$$

$$\text{with } \lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{if } \boldsymbol{P} \neq \boldsymbol{Q} \text{ [addition]} \\ \dfrac{3{x_1}^2 + 2a_2\,x_1 + a_4 - a_1\,y_1}{2y_1 + a_1\,x_1 + a_3} & \text{if } \boldsymbol{P} = \boldsymbol{Q} \text{ [doubling operation]} \end{cases} .$$

Provided that the characteristic of field $\mathbb{F}_q$ is different from $2, 3$, we can take $a_1 = a_2 = a_3 = 0$. In the sequel we will also assume that $q = p$ is prime. We define the short Weierstraß form over prime field $\mathbb{F}_p$ by the equation

$$y^2 = x^3 + a\,x + b \ . \tag{2}$$

Note that the slope λ in the doubling then becomes $\lambda = (3{x_1}^2 + a)/(2y_1)$, which can be rewritten as $3(x_1 - 1)(x_1 + 1)/(2y_1)$ when $a = -3$.

The scalar multiplication of a given point is a fundamental operation in cryptographic algorithms that use elliptic curve arithmetic, i.e. $[k]\boldsymbol{P}$ for some integer $k < |\mathcal{E}|$. This operation uses the above addition law in conjunction with algorithms analogous to standard exponentiation algorithms in $(\mathbb{Z}/N\mathbb{Z})^*$.

In this paper we concentrate on the short Weierstraß form since this is typically the form one will find in standards, and is, therefore, the most commonly used. For example, one can find standardized elliptic curves in the short Weierstraß form in FIPS 186-3 [22], WTLS [32] and ANSI X9.62 [33].

2.2 Side-Channel Resistant Scalar Multiplication

When implementing a scalar multiplication using elliptic curve arithmetic on a device that could potentially be attacked using side-channel analysis, there are a variety of considerations that need to be taken into account. The simplest type of side-channel analysis consists of timing or simple power analysis [18, 19], where an attacker attempts to derive information from the time taken for an algorithm to execute or to identify operations from a few traces. Given that an attacker would expect to be able to distinguish an addition from a doubling operation, this requires an implementation to include one of the following countermeasures.

Regular Multiplication Algorithms. A variety of algorithms have been proposed that will always compute a regular sequence of additions and doubling operations (these methods are surveyed in [16]).

Unified Addition Formulae. The addition and doubling operations can be implemented such that the same operations are performed for both an addition and a doubling operation (e.g. [5, 7]).

Dummy Operations. An alternative to unified addition formulae was proposed in [11, 8], where the two operations are rendered indistinguishable using dummy operations. However, this approach can introduce the possibility of a safe-error fault attack [34], although a discussion of fault analysis is beyond the scope of this paper.

Furthermore, implementations need to be able to prevent an attacker from using the observation that the power consumption (and electromagnetic field) is related to the Hamming weight of the data being manipulated by a microprocessor at any given point in time [12, 19, 23], referred to as differential side-channel analysis. This requires an implementation to include further countermeasures to blind the computation. The scalar itself can be protected using:

Multiplier Blinding. The scalar k can be modified by adding a random multiple of the order of the group $\mathcal{E}$ to the scalar k. This modifies the bits of k without changing the output of a scalar multiplication [11].

There are numerous options for blinding the points being operated on. A summary of existing countermeasures is given below.

Point Blinding. If, for a given point $\boldsymbol{R}$, where $\boldsymbol{S} = [k]\,\boldsymbol{R}$ is known, then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = [k]\,(\boldsymbol{P} + \boldsymbol{R}) - \boldsymbol{S}$. Points $\boldsymbol{R}$ and $\boldsymbol{S}$ can be stored in a device along with k and updated after each execution by computing $\boldsymbol{R} \leftarrow [r]\,\boldsymbol{R}$ and $\boldsymbol{S} \leftarrow [r]\,\boldsymbol{S}$ for some small random value r [11, 18].

Multiplier Splitting. A scalar can be divided into two values whose bitwise representations are random. This allows a scalar multiplication to be conducted with two values whose combined effect is equivalent to that of the desired scalar [9]. There are three methods of multiplier splitting:

- **Additive Splitting.** If we define the scalar $k = r + (k - r)$ for some integer r that has a similar bit-length to k, then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = [r]\,\boldsymbol{P} + [k - r]\,\boldsymbol{P}$.
- **Multiplicative Splitting.** For some elliptic curve $\mathcal{E}$ over $\mathbb{F}_q$ we define $k' = k\,r^{-1} \bmod |\mathcal{E}|$ for some integer r. Then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = [k']\,([r]\,\boldsymbol{P})$.
- **Euclidean Splitting.** If we define the scalar $k' = \lfloor k/r \rfloor$ for some integer r, then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = [k']\,([r]\boldsymbol{P}) + [k \bmod r]\,\boldsymbol{P}$.

Randomized Projective Points. An affine point $\boldsymbol{P} = (x, y)$ can, for example, be represented as a homogeneous projective point $(\theta\,x, \theta\,y, \theta)$ for all $\theta \in \mathbb{F}_p \backslash \{0\}$ (this is covered in more detail in Section 3). When computing a scalar multiplication using projective coordinates a randomly generated $\theta \in \mathbb{F}_p \backslash \{0\}$ can be determined at the beginning of the computation so that an attacker cannot guess what values are being manipulated [11].

Random Curve Isomorphisms. A given $\boldsymbol{P}$ on elliptic curve $\mathcal{E}$ can be randomized by computing $\boldsymbol{P}^* \leftarrow \psi(\boldsymbol{P})$ on $\mathcal{E}^* \leftarrow \psi(\mathcal{E})$ for a random curve isomorphism ψ. Then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = \psi^{-1}([k]\,\boldsymbol{P}^*)$ [17].

Of these countermeasures, the first two are not practical as they highly impact the execution time of a scalar multiplication or require dedicated operations not always readily available. Using randomized projective coordinates is much more efficient but does not allow θ to be set to one. It is for this reason that it is observed in [25] that using random curve isomorphisms is the most efficient of these countermeasures. However, when using random curve isomorphisms the parameters of $\mathcal{E}^*$ cannot be chosen and one cannot take advantage of algorithms that require curve parameters to be set to specific values.

3 Implementing Elliptic Curve Arithmetic

For elliptic curve arithmetic over $\mathbb{F}_p$ the use of projective coordinates is preferred as no inversion is required for an addition or a doubling operation [31]. A point on an elliptic curve can be represented with projective coordinates (X, Y, Z) that are not unique for a given affine point. For example, homogeneous projective coordinates $(\theta\, x, \theta\, y, \theta)$ represent the affine point (x, y) for all $\theta \in \mathbb{F}_p \setminus \{0\}$, and the point at infinity $\boldsymbol{O}$ is represented by $(0, \gamma, 0)$ for some $\gamma \in \mathbb{F}_p \setminus \{0\}$. The simplest countermeasure that can be applied to these coordinate systems is to choose some random $\theta \in \mathbb{F}_p \setminus \{0\}$ and use the point $(\theta\, x, \theta\, y, \theta)$ as a random representation of the affine point (x, y) [11] (referred to as randomized projective points in the previous section). However, when using this representation, the Z-coordinate cannot be chosen to be one. In the following sections we define addition rules for randomized projective coordinates where the Z-coordinate can be chosen to be one.

3.1 Homogeneous Projective Coordinates

As described above, homogeneous projective coordinates $(\theta\, x, \theta\, y, \theta)$ represent the affine point (x, y) for all $\theta \in \mathbb{F}_p \setminus \{0\}$, and the point at infinity $\boldsymbol{O}$ is represented by $(0, \gamma, 0)$ for some $\gamma \in \mathbb{F}_p \setminus \{0\}$. We define the map Φ as mapping a point $\boldsymbol{P} = (X, Y, Z) \in \mathcal{E}$ to the point $\boldsymbol{P'}$ where $\Phi(\boldsymbol{P}) = (X', Y', Z) = (f^{\mu} X, f^{\nu} Y, Z)$ for an arbitrary $f \in \mathbb{F}_p \setminus \{0\}$ and some small integers μ and ν. Note that $\boldsymbol{P'}$ is not necessarily on $\mathcal{E}$. The inverse of Φ can be computed without inverting f since $\boldsymbol{P} = \Phi^{-1}(\boldsymbol{P'}) = (f^{\nu} X', f^{\mu} Y', f^{\mu+\nu} Z)$.

Consider the addition of two homogeneous projective points $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q}$. In order to blind the computation, we redefine the addition algorithm such that $\Phi(\boldsymbol{R}) = \boldsymbol{R'} = \boldsymbol{P'} + \boldsymbol{Q'} = \Phi(\boldsymbol{P}) + \Phi(\boldsymbol{Q})$. We define the point $\boldsymbol{R'} = (X'_3, Y'_3, Z_3)$, $\boldsymbol{P'} = (X'_1, Y'_1, Z_1)$ and $\boldsymbol{Q'} = (X'_2, Y'_2, Z_2)$. If $\boldsymbol{P} = \boldsymbol{Q}$, then $\boldsymbol{R'}$ can be computed from $\boldsymbol{P'}$ and $\boldsymbol{Q'}$ by calculating

$$\begin{aligned} X'_3 &= \lambda_{10}\lambda_4 \\ Y'_3 &= f^{2\nu-3\mu}\lambda_3(\lambda_9 - \lambda_{10}) - 2\lambda_8\,, \\ Z_3 &= \lambda_6 \end{aligned}$$

where $\lambda_1 = {X'_1}^2$, $\lambda_2 = {Z_1}^2$, $\lambda_3 = a\, f^{2\mu}\, \lambda_2 + 3\lambda_1$, $\lambda_4 = 2Y'_1 Z_1$, $\lambda_5 = {\lambda_4}^2$, $\lambda_6 = \lambda_4\lambda_5$, $\lambda_7 = Y'_1\lambda_4$, $\lambda_8 = {\lambda_7}^2$, $\lambda_9 = (X'_1 + \lambda_7)^2 - \lambda_1 - \lambda_8$ and $\lambda_{10} =$

$f^{2\nu-3\mu}{\lambda_3}^2 - 2\lambda_9$ [5]. This requires an extra three multiplications with a power of f.

If $a = -3$ a faster doubling algorithm can be used and $\boldsymbol{R'}$ can be computed by calculating

$$\begin{aligned} X'_3 &= \lambda_8\lambda_2 \\ Y'_3 &= f^{2\nu-3\mu}\lambda_1(\lambda_7 - \lambda_8) - 2\lambda_6\,, \\ Z_3 &= \lambda_4 \end{aligned}$$

where $\lambda_0 = f^{\mu}Z_1$, $\lambda_1 = 3(X'_1 - \lambda_0)(X'_1 + \lambda_0)$, $\lambda_2 = 2Y'_1Z_1$, $\lambda_3 = {\lambda_2}^2$, $\lambda_4 = \lambda_2\lambda_3$, $\lambda_5 = Y'_1\lambda_2$, $\lambda_6 = {\lambda_5}^2$, $\lambda_7 = 2X'_1\lambda_5$ and $\lambda_8 = f^{2\nu-3\mu}{\lambda_1}^2 - 2\lambda_7$ [5]. This also requires an extra three multiplications with a power of f.

If $\boldsymbol{P} \neq \boldsymbol{Q}$, then $\boldsymbol{R'}$ is computed by calculating

$$\begin{aligned} X'_3 &= \lambda_6\lambda_{10} \\ Y'_3 &= \lambda_4(\lambda_9 - \lambda_{10}) - \lambda_8\lambda_1\,, \\ Z_3 &= \lambda_8\lambda_3 \end{aligned}$$

where $\lambda_1 = Y'_1Z_2$, $\lambda_2 = X'_1Z_2$, $\lambda_3 = Z_1Z_2$, $\lambda_4 = Y'_2Z_1 - \lambda_1$, $\lambda_5 = {\lambda_4}^2$, $\lambda_6 = X'_2Z_1 - \lambda_2$, $\lambda_7 = {\lambda_6}^2$, $\lambda_8 = \lambda_6\lambda_7$, $\lambda_9 = \lambda_7\lambda_2$ and $\lambda_{10} = f^{3\mu-2\nu}\lambda_5\lambda_3 - \lambda_8 - 2\lambda_9$ [10]. This requires a single extra multiplications with a power of f.

3.2 Jacobian Projective Coordinates

Jacobian projective coordinates $(\theta^2\,x, \theta^3\,y, \theta)$ represent the affine point (x, y) for any $\theta \in \mathbb{F}_p \setminus \{0\}$, and the point at infinity $\boldsymbol{O}$ is represented by $(\gamma^2, \gamma^3, 0)$ for some $\gamma \in \mathbb{F}_p \setminus \{0\}$. We define the map $\varPhi$ as mapping a point $\boldsymbol{P} = (X, Y, Z) \in \mathcal{E}$ to the point $\boldsymbol{P'}$ where $\varPhi(\boldsymbol{P}) = (X', Y', Z) = (f^{\mu}X, f^{\nu}Y, Z)$ for an arbitrary $f \in \mathbb{F}_p \setminus \{0\}$ and some small integers μ and ν. Note that $\boldsymbol{P'}$ is not necessarily in $\mathcal{E}$. The inverse of $\varPhi$ can be computed without inverting f since $\boldsymbol{P} = \varPhi^{-1}(\boldsymbol{P'}) = (f^{\mu+2\nu}X', f^{3\mu+2\nu}Y', f^{\mu+\nu}Z)$.

Consider the addition of two Jacobian projective points $\boldsymbol{R} = \boldsymbol{P} + \boldsymbol{Q}$. In order to blind the computation, we redefine the addition algorithm such that $\varPhi(\boldsymbol{R}) = \boldsymbol{R'} = \boldsymbol{P'} + \boldsymbol{Q'} = \varPhi(\boldsymbol{P}) + \varPhi(\boldsymbol{Q})$. We define the point $\boldsymbol{R'} = (X'_3, Y'_3, Z_3)$, $\boldsymbol{P'} = (X'_1, Y'_1, Z_1)$ and $\boldsymbol{Q'} = (X'_2, Y'_2, Z_2)$. If $\boldsymbol{P} = \boldsymbol{Q}$, then $\boldsymbol{R'}$ can be computed from $\boldsymbol{P'}$ and $\boldsymbol{Q'}$ by calculating

$$\begin{aligned} X'_3 &= \lambda_7 \\ Y'_3 &= f^{2\nu-3\mu}\lambda_6(\lambda_5 - \lambda_7) - 8\lambda_3\,, \\ Z_3 &= (Y'_1 + Z_1)^2 - \lambda_2 - \lambda_4 \end{aligned}$$

where $\lambda_1 = {X'_1}^2$, $\lambda_2 = {Y'_1}^2$, $\lambda_3 = {\lambda_2}^2$, $\lambda_4 = {Z_1}^2$, $\lambda_5 = 2((X'_1 + \lambda_2)^2 - \lambda_1 - \lambda_3)$, $\lambda_6 = 3\lambda_1 + af^{2\mu}\,{\lambda_4}^2$, $\lambda_7 = f^{2\nu-3\mu}{\lambda_6}^2 - 2\lambda_5$ [5]. This requires an extra three multiplications with a power of f.

If $a = -3$ a faster doubling algorithm can be used and $\boldsymbol{R'}$ can be computed by calculating

$$\begin{aligned} X_3' &= f^{2\nu-3\mu}{\lambda_5}^2 - 8\lambda_3 \\ Y_3' &= f^{2\nu-3\mu}\lambda_5(4\lambda_3 - X_3') - 8{\lambda_2}^2 \,, \\ Z_3 &= (Y_1' + Z_1)^2 - \lambda_2 - \lambda_1 \end{aligned}$$

where $\lambda_1 = {Z_1}^2$, $\lambda_2 = {Y_1'}^2$, $\lambda_3 = X_1'\lambda_2$, $\lambda_4 = f^{\mu}\lambda_1$, $\lambda_5 = 3(X_1' - \lambda_4)(X_1' + \lambda_4)$ [4]. Again, this requires an extra three multiplications with a power of f.

If $\boldsymbol{P} \neq \boldsymbol{Q}$, then $\boldsymbol{R'}$ is computed by calculating

$$\begin{aligned} X_3' &= f^{3\mu-2\nu}{\lambda_{10}}^2 - \lambda_9 - 2\lambda_{11} \\ Y_3' &= \lambda_{10}(\lambda_{11} - X_3') - 2\lambda_5\lambda_9 \qquad , \\ Z_3 &= ((Z_1 + Z_2)^2 - \lambda_1 - \lambda_2)\lambda_7 \end{aligned}$$

where $\lambda_1 = {Z_1}^2$, $\lambda_2 = {Z_2}^2$, $\lambda_3 = X_1'\lambda_2$, $\lambda_4 = X_2'\lambda_1$, $\lambda_5 = Y_1'Z_2\lambda_2$, $\lambda_6 = Y_2'Z_1\lambda_1$, $\lambda_7 = \lambda_4 - \lambda_3$, $\lambda_8 = (2\lambda_7)^2$, $\lambda_9 = \lambda_7\lambda_8$, $\lambda_{10} = 2(\lambda_6 - \lambda_5)$, $\lambda_{11} = \lambda_3\lambda_8$ [5]. This requires a single extra multiplications with a power of f.

3.3 Choosing μ and ν

The above algorithms were defined to minimize the number of multiplications with f, or some power of f. However, one would wish to avoid a situation where the inverse of f is required. This means that, for the above algorithms, choices for μ and ν need to satisfy $2\,\nu \geq 3\,\mu$ and $3\,\mu \geq 2\,\nu$; that is, $2\,\nu = 3\,\mu$.

For both homogeneous and Jacobian projective coordinates the choice of $2\,\nu = 3\,\mu$ would allow for any multiplication with a power of f to be removed from the algorithm for computing the addition of two distinct points. That is, the countermeasure would have no impact on the performance of a point addition. Define $a' = a\,f^{2\mu}$. In the case of $a \neq -3$, if the cost of a multiplication by a' is the same to that of a multiplication by a, choosing $2\,\nu = 3\,\mu$ incurs no performance loss for the doubling operation in both homogeneous and Jacobian coordinates. If $a = -3$ the choice of $2\,\nu = 3\,\mu$ leads to only an extra multiplication by f^{μ} (in the evaluation of λ_0 for homogeneous coordinates and in the evaluation of λ_4 for Jacobian coordinates, respectively).

Case of $\mu = 2$ *and* $\nu = 3$. As a reminder, the elliptic curves $\mathcal{E} : y^2 = x^3 + a\,x + b$ and $\mathcal{E}^* : y^2 = x^3 + a^*x + b^*$ over $\mathbb{F}_p$ are isomorphic if and only if there exists some $f \in \mathbb{F}_p \setminus \{0\}$ such that $a^* = f^4\,a$ and $b^* = f^6\,b$. The isomorphism is given by $\psi : \mathcal{E} \xrightarrow{\sim} \mathcal{E}^* : \boldsymbol{P} = (x, y) \mapsto \boldsymbol{P^*} = (f^2\,x, f^3\,y)$ and $\boldsymbol{O} \mapsto \boldsymbol{O}$. It appears that the specific choice of $\mu = 2$ and $\nu = 3$ corresponds to the technique of using randomized curve isomorphisms [17].

4 Implementation Considerations

4.1 Using Montgomery multiplication

When implementing an elliptic curve cryptographic algorithm over $\mathbb{F}_p$, it would be natural to use Montgomery multiplication [21], since the modular reduction is interleaved with the multiplication. As shown in Algorithm 1, the result of a Montgomery multiplication is not the product of x and y modulo p. The algorithm actually returns $x\,y\,R^{-1} \bmod p$, where $R^{-1} \bmod p$ is introduced by the algorithm ($R = b^n$, where the modulus consists of n words of size b). In order to use Montgomery multiplication x and y need to be converted to their Montgomery representation, i.e. $\widetilde{x} \leftarrow x\,R \bmod p$ and $\widetilde{y} \leftarrow y\,R \bmod p$. Then, when $\widetilde{x}$ and $\widetilde{y}$ are multiplied together using Montgomery multiplication, the result is $x\,y\,R \bmod p$.

Algorithm 1: Montgomery multiplication

```
Input: $p = (p_{n-1}, \ldots, p_1, p_0)_b$, $x = (x_{n-1}, \ldots, x_1, x_0)_b$, $y = (y_{n-1}, \ldots, y_1, y_0)_b$
       with $0 \le x, y < p$, $R = b^n$, $\gcd(p, b) = 1$ and $p' = -p^{-1} \bmod b$.
Output: $A = x\,y\,R^{-1} \bmod p$.
$A \leftarrow 0$ ;
for $i = 0$ to $n - 1$ do
    $u_i = (a_0 + x_i\,y_0)p' \bmod b$ ;
    $A = (A + x_i\,y + u_i\,p)/b$ ;
end
if $A \ge p$ then $A \leftarrow A - p$ ;
return $A$
```

When implementing Montgomery multiplication for use in a group exponentiation one has to be aware that an attacker can use the final conditional subtraction to try and derive information on the exponent used. An attacker can potentially use the difference in time caused by the total number of subtractions [30] or by identifying individual subtractions in a power consumption trace (or other suitable side-channel) [28, 29]. The final subtraction can be removed by increasing the number of iterations of the main loop [14, 27]. However, these attacks and countermeasures are beyond the scope of this paper, since the arguments concerning the efficiency of the countermeasure described in Section 3 will remain unchanged.

Where a multiplication with a small value is required, such as the multiplication with the constant a in the short Weierstraß form, this value needs to be converted into its Montgomery representation. This means that the cost of such a multiplication will require the same number of single-precision multiplications as any other multiplication or squaring operations over $\mathbb{F}_p$, i.e. $n(2\,n + 2)$ single-precision multiplications.

4.2 Generating f

A random value is typically generated for blinding purposes in a given instance of a side-channel resistant implementation of an algorithm. These values do are typically chosen to be relatively small, since the bit-length of the random value only needs to be large enough that an attacker cannot guess its value for multiple executions of the algorithm. That is, an attacker is required to guess this value for each acquisition in order to conduct a differential side-channel analysis [20]. For example, we can define f to be in $\{1, \dots, b-1\}$ but multiplying with f has the same problem as multiplying by a since f is not in its Montgomery representation. However, we can define the value that is used when multiplying with the mask to be $f' \equiv b\, f \pmod{p}$ with $f' \in \{1, \dots, b-1\}$ and $f \in \mathbb{F}_p \setminus \{0\}$. This means that Algorithm 2, which is one iteration of the main loop of the Montgomery multiplication algorithm, can be used without having to correct the factor of $b^{-1} \bmod p$ that is introduced.

This allows a multiplication with the mask f using $2\,n+2$ single-precision multiplications, and can be repeated for powers of f as required, i.e. multiplying with f^{μ} requires $\mu(2\,n{+}2)$ single-precision multiplications. In practice this means a random value in $\{1, \dots, b-1\}$ can be generated and used in Algorithm 2.

We can note that a scalar multiplication using the algorithms described in Section 3 can be implemented using an arbitrarily chosen f' without knowing f. That is, all required multiplications with f can be conducted using f' and Algorithm 2. This includes converting a blinded point to a valid projective point by multiplying the coordinates by the required power of f. The further advantage of only using f' is that it is not necessary to store the montgomery representation of any powers of f in memory.

Algorithm 2: Montgomery multiplication with f

```
Input: $p = (p_{n-1}, \dots, p_1, p_0)_b$, $f \in \{0, \dots, b-1\}$, $y = (y_{n-1}, \dots, y_1, y_0)_b$ with
        $0 \leq y < p$, $R = b^n$, $\gcd(p, b) = 1$ and $p' = -p^{-1} \bmod b$.
Output: $A = f\,y\,b^{-1} \bmod p$.
$u = f\,y_0\,p' \bmod b$ ;
$A = (f\,y + u\,p)/b$ ;
if $A \geq p$ then $A \leftarrow A - p$ ;
return $A$
```

4.3 Performance

If the above optimization is applied to the algorithms described in Sections 3.1 and 3.2 the number of single-precision multiplications required can be reduced.

It is observed in Section 3.3 that the smallest penalty for using the proposed blinding method is incurred when $\mu = 2$ and $\nu = 3$, and no extra cost is observed for many of the operations. Where $a = -3$, the cost of using the proposed blinding method will incur an extra multiplication with f^{μ} can be computed with $(4n + 4)$ single-precision multiplications, rather than $n(2n + 2)$ for a full Montgomery multiplication.

If we consider randomize curve isomorphisms, the case where $a* = -3f^4$ (i.e. an isomorphic curve that would allow one to use the algorithm defined for $a = -3$) the necessary extra multiplication can be computed in the same way, and the same gain in performance would be observed. Using the above observation would also, therefore, allow the time required to compute operations on an isomorphic curve to be reduced.

5 Further Security Considerations

The algorithms defined in Section 3 can readily be used to implement a side-channel resistant scalar multiplication. These building blocks are, themselves, resistant to side-channel analysis (within certain bounds we will discuss in this section) and merely require a suitable multiplication algorithm to be chosen. We refer the reader to [16] for a discussion of this topic.

However, in [13] it is observed that elliptic curve arithmetic that uses multiplicative blinding will not necessarily prevent a scalar multiplication from being derived. If a point corresponding to the affine points $(0, y)$ or $(x, 0)$ exists, for some $x, y \in \mathbb{F}_p$, then an attacker could attempt to have this point produced as an intermediate state of a scalar multiplication, which could then be used to verify hypotheses concerning the scalar. This would be possible as multiplicative blinding will have no effect on a coordinate set to zero.

An extension to this attack was proposed in [1] that relied on the same observation, that the value zero cannot be blinded multiplicatively. They noted that the same attack could be conducted if any of the intermediate states could be equal to zero. That is, if there exists some combination of points where the point arithmetic will generate a zero as an intermediate state.

The simplest countermeasure to this attack would be simply use curves that do not have these points. However, it is noted in [13] that in many standardized curves a point exists where the x-coordinate is zero, but not where the y-coordinate is zero.

Countermeasures, therefore, need to be included when implementing a cryptosystem that uses an elliptic curve that can be attacked in this manner. Two such countermeasures are:

Linear Blinding. One countermeasure to this type of attack is described in [15], where the authors propose that the coordinates of a projective coordinate are modified by adding an extra coordinate. For example, to protect the x-coordinate one could define a projective point, for example, where an affine point $\boldsymbol{P} = (x, y)$ can be represented as a projective point $(\theta\,(x - \beta), \theta\, y, \theta, \beta)$ for all $\theta, \beta \in \mathbb{F}_p \setminus \{0\}$. This involves redefining the algorithms for addition and doubling operations and considerably increases the number of operations required.

Isogenies. It is pointed out in [26] that an isogenous curve can be selected. That is, an isogeny between elliptic curves $\mathcal{E}_1$ and $\mathcal{E}_2$ over the same field exists if a surjective morphism κ can be defined that preserves the identity element (i.e. the point at infinity $\boldsymbol{O}$). When implementing a scalar multiplication using an elliptic curve where zero-coordinates are possible, one can select an isogenous curve that does not have any points with zero-coordinates. Then $\boldsymbol{Q} = [k]\,\boldsymbol{P}$ can be computed by calculating $\boldsymbol{Q} = \kappa^{-1}([k]\,\kappa(\boldsymbol{P}))$. Further constraints on what isogenies can be used were defined in [2] to avoid intermediate states being attacked.

Of these two countermeasures, the use of isogenies is more efficient as the same algorithms can be used for point arithmetic with the addition of two transformations. Moreover, these transformations can be defined when a cryptosystem is implemented to minimize the impact on the time taken to compute a scalar multiplication. The principle problem with using linear blinding is that it has a large impact on the point addition algorithms.

6 Conclusion

In this paper we propose an multiplicative blinding method for protecting a scalar multiplication that is a generalization of the use of randomized curve isomorphisms. We also discuss how one could efficiently implement this countermeasure using Montgomery multiplication, and show that this would allow for a faster implementation than a naïve use of randomized curve isomorphisms.

The specific choice of $\mu = 2$ and $\nu = 3$ incurs only a small increase in the execution time of a scalar multiplication. However, as noted above, this corresponds to using an isomorphic curve, and that the optimizations presented in Section 4 also apply. That is, if we apply the same criteria in choosing f, i.e. such that $f' \equiv f\,b \bmod p$ is $\in \{1, \ldots, b-1\}$, the performance will be identical.

We note that Algorithm 2 could also be used to efficiently implement randomized projective coordinates [11]. The aim of this randomization is to make the intermediate values unpredictable by an attacker and it is not necessary to choose a random value with the same bit-length as the x and y-coordinates.

Acknowledgments

The work described in this paper has been supported in part by the European Commission IST Programme under Contract IST-2002-507932 ECRYPT and EPSRC grant EP/F039638/1 “Investigation of Power Analysis Attacks”.

References

1. T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystems. In C. Boyd and W. Mao, editors, *ISC 2003*, volume 2851 of *LNCS*, pages 218–233. Springer, 2003.

2. T. Akishita and T. Takagi. On the optimal parameter choice for elliptic curve cryptosystems using isogeny. In F. Bao, R. H. Deng, and J. Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 346–359. Springer, 2004.
3. M.-L. Akkar and C. Giraud. An implementation of DES and AES secure against some attacks. In C. K. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 309–318. Springer, 2001.
4. D. J. Bernstein. A software implementation of NIST P-224. `http://cr.yp.to/nistp224.html`, 2001.
5. D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 29–50. Springer, 2007.
6. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.
7. E. Brier and M. Joye. Weierstraß elliptic curve and side-channel attacks. In D. Naccache and P. Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 335–345. Springer, 2002.
8. B. Chevallier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, 2004.
9. C. Clavier and M. Joye. Universal exponentiation algorithm. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 300–308. Springer, 2001.
10. H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In K. Ohta and D. Pei, editors, *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 51–65. Springer, 1998.
11. J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In C. K. Koç and C. Paar, editors, *CHES 99*, volume 1717 of *LNCS*, pages 292–302. Springer, 1999.
12. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In C. K. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.
13. L. Goubin. A refined power analysis attack on elliptic curve cryptosystems. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 199–210. Springer, 2003.
14. G. Hachez and J.-J. Quisquater. Montgomery exponentiation with no final subtractions: Improved results. In C. K. Koç and C. Paar, editors, *CHES 2000*, volume 1965 of *LNCS*, pages 293–301. Springer, 2000.
15. K. Itoh, T. Izu, and M. Takenaka. Efficient countermeasures against power analysis for elliptic curve cryptosystems. In J.-J. Quisquater et al., editors, *Smart Card Research and Advanced Applications VI*, pages 99–113. Kluwer Academic Publishers, 2004.
16. M. Joye and M. Tunstall. Exponent recoding and regular exponentiation algorithms. In B. Preneel, editor, *AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 334–349. Springer, 2009.
17. M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography: An algebraic approach. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 377–390. Springer, 2001.
18. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

19. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
20. S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks — Revealing the Secrets of Smart Cards*. Springer, 2007.
21. P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.
22. National Institute of Standards and Technology (NIST). Recommended elliptic curves for federal government use. In the appendix of FIPS 186-3, available from `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf`, June 2009.
23. J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards — E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.
24. R. Rivest, A. Shamir, and L. M. Adleman. Method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
25. N. Smart, E. Oswald, and D. Page. Randomised representations. *IET Proceedings on Information Security*, 2(2):19–27, 2008.
26. N. P. Smart. An analysis of Goubin's refined power analysis attack. In C. D. Walter, editor, *CHES 2003*, volume 2779 of *LNCS*, pages 281–290. Springer, 2003.
27. C. D. Walter. Montgomery exponentiation needs no final subtractions. *Electronic Letters*, 35(21):1831–1832, October 1999.
28. C. D. Walter. Longer keys may facilitate side channel attacks. In M. Matsui and R. J. Zuccherato, editors, *SAC 2004*, volume 3006 of *LNCS*, pages 42–57. Springer, 2004.
29. C. D. Walter. Simple power analysis of unified code for ECC double and add. In M. Joye and J.-J. Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 191–204. Springer, 2004.
30. C. D. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. In D. Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 192–207. Springer, 2001.
31. E. De Win, S. Mister, B. Preneel, and M. Wiener. On the performance of signature schemes based on elliptic curves. In J.-P. Buhler, editor, *Algorithmic Number Theory Symposium*, volume 1423 of *LNCS*, pages 252–266. Springer, 1998.
32. Wireless Application Protocol (WAP) Forum. Wireless transport layer security (WTLS) specification. Available from `http://www.wapforum.org`.
33. ANSI X9.62. Public key cryptography for the financial services industry, the elliptic curve digital signature algorithm (ECDSA), 1999.
34. S.-M. Yen and M. Joye. Checking before output may not be enough against fault based cryptanalysis. *IEEE Transactions on Computers*, 49(9):967–970, 2000.