Unified Point Addition Formulæ and Side-Channel Attacks

Douglas Stebila[1,⋆] and Nicolas Thériault[2]

[1] Institute for Quantum Computing,
University of Waterloo, Waterloo, ON, Canada,
dstebila@iqc.ca
[2] Department of Combinatorics and Optimization,
University of Waterloo, Waterloo, ON, Canada,
ntheriau@math.uwaterloo.ca

Abstract. The successful application to elliptic curve cryptography of side-channel attacks, in which information about the secret key can be recovered from the observation of side channels like power consumption or timing, has motivated the recent development of unified formulæ for elliptic curve point operations. In this paper, we give a version of a previously-developed family of unified point addition formulæ that uses projective coordinates for improved efficiency. We discuss the applicability of a recent attack by Walter on this family of projective formulæ and describe how the field arithmetic can be implemented to obtain fully unified formulæ and avoid this type of attack.

Keywords: elliptic-curve cryptography, side-channel attacks, unified point addition formulæ, projective coordinates.

1 Introduction

The study of elliptic curves in cryptography [1, 2] has been ongoing for a number of years. Elliptic curve cryptography offers higher security per key bit compared to other public key cryptosystems and the smaller key size is more suitable for implementation on small devices such as smart cards. More recently, a new class of attacks has been discovered, called *side-channel* attacks [3], which use information observed during the execution of the algorithm to help determine the secret key. There are two classes of side-channel attacks: *simple* side-channel attacks, which analyze the trace of a single execution of a cryptographic protocol, and *differential* side-channel attacks, which compare the traces of multiple executions of a protocol.

The central operation in an elliptic curve cryptosystem is the *point multiplication* operation, in which a point is multiplied by a scalar. The basic method for

⋆ This work was performed while the author was a visiting researcher at Sun Microsystems Laboratories, with additional support from NSERC, ORDCF, CIAR, CFI, and MITACS.

implementing point multiplication is the *double-and-add* technique, which uses a binary representation of the scalar and performs a sequence of point additions and point doublings depending on the bits of the scalar. The double-and-add technique is given in Fig. 1.

Input:	Point P, integer $k = \sum_{i=0}^{n-1} k_i 2^i, k_{n-1} = 1$.
Output:	Point $Q = kP$.
1.	$Q \leftarrow P$
2.	for $i = n-2$ down to 0 do
2.1.	$Q \leftarrow 2Q$
2.2.	if $k_i = 1$ then $Q \leftarrow P + Q$
3.	end for

Fig. 1. Double-and-add point multiplication algorithm

In double-and-add point multiplication, a point doubling is done for every bit of the key k, but a point addition is done only when a bit of the key is 1. If, in a side-channel analysis, a point addition is distinguishable from a point doubling, then the bits of the secret key can be determined; this has been demonstrated experimentally using timing and power analysis [3, 4]. Techniques for counteracting this problem include: performing dummy operations, such as a point addition, each iteration [5]; using alternate point multiplication algorithms, such as Montgomery point multiplication [6]; using alternate curve parameterizations, such as the Jacobi or Hessian forms; and unifying the algorithms for point addition and point doubling so that they use the same sequence of field operations and hence are indistinguishable; it is this last technique that we address in this paper.

A *unified formula* for point addition and point doubling, in which point addition and point doubling use the same sequence of field operations, was first given by Brier and Joye [7] in affine and projective form. Walter [8] demonstrated a theoretical side-channel attack on the formula of Brier and Joye that, instead of exploiting any irregularity in the sequence of field operations performed, exploits an irregularity in the implementation of the field operations themselves in the context of the unified point addition formula. A subsequent paper of Brier, Déchène, and Joye [10] offers an infinite family of unified point addition formulæ in affine form.

In this paper, we give a projective version of the unified point addition formulæ of Brier, Déchène, and Joye. Whereas Walter's attack used the occurrence of the *conditional subtraction* in a Montgomery field multiplication, we note that a conditional addition (in any field representation) is also an integral step of field subtraction. A typical algorithm for computing prime field subtraction is given in Fig. 2; the conditional addition is step 2.[3]

[3] Similarly, a field addition contains a conditional subtraction, however our techniques of Sec. 5 do not make use of this conditional subtraction.

We find that the ability to detect the occurrence of the conditional addition in field subtractions in both the affine and projective form decreases the amount of work necessary to recover the key. In the projective case in Montgomery representation, the effect is substantial when combined with Walter's original attack, and in some cases the key for a 192-bit elliptic curve over a prime field can be recovered in about 2 hours. From this observation we conclude that a secure implementation requires constant-runtime field operations, not just unified point arithmetic.

We also provide some performance results for the various unified formulæ and discuss the applicability of timing attacks. It appears that even though unified point addition and unified point doubling algorithms have slightly different runtime, the timing difference is not large enough to be exploited.

Input:	Integers a, b, q such that $0 \leq a, b \leq q-1$.
Output:	Integer c such that $c = a - b \mod q$ and $0 \leq c \leq q-1$.
1.	$c \leftarrow a - b$
2.	if $c < 0$ then $c \leftarrow c + q$

Fig. 2. Field subtraction algorithm

This paper is organized as follows: Section 2 provides a short introduction to elliptic curve cryptography. In Sec. 3, we describe the unified formula of Brier and Joye and describe an attack by Walter. In Sec. 4, we describe the family of unified formulæ in affine coordinates given by Brier, Déchène, and Joye and give our derivation of the formulæ for projective coordinates. In Sec. 5, we present an extension of Walter's attack, analyze its effect on the formulæ and discuss countermeasures. Section 6 contains performance results and discusses the possibility of timing attacks on double-and-add projective unified point multiplication.

Throughout the body of this paper, we derive formulæ only for elliptic curves over prime fields. Formulæ and results for elliptic curves over binary fields are given in the appendix. We have omitted the discussion of curves over binary fields from the main text because the attacks of Sec. 5 do not apply.

2 Background

For a field $\mathbb{K}$, the Weierstraß form of an elliptic curve is given by the equation

$$E_{/\mathbb{K}} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ . \tag{1}$$

For fields of characteristic other than 2 or 3, the equation of the curve can be simplified to

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

The set of points on the curve, joined with the *point at infinity* $\mathcal{O}$, forms an abelian group, denoted $E(\mathbb{K})$; the point at infinity serves as the identity element in the group. Two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq Q$, can be added to obtain a third point $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2 \tag{3}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \tag{4}$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \end{cases} . \tag{5}$$

Because λ is defined differently depending on whether or not $P = Q$, the formula for point addition differs from the formula for point doubling.

The formula given above uses *affine coordinates*. The formula for λ requires an inversion, which can be computationally expensive in practice. This has motivated the development of formulæ using *projective coordinates*. In this case, a point is represented by three coordinates, $P = (X, Y, Z)$, with $x = X/Z$ and $y = Y/Z$. The denominator is carried through all of the point additions and point doublings comprising a point multiplication, and only at the end is the inversion Z^{-1} computed to return the final result to affine coordinates. While this technique may require more field operations such as multiplications or squarings in each point addition or doubling, it may be more efficient provided the cost of inversion is higher than the cost of the additional field multiplications and squarings.

3 Unified Formula of Brier and Joye

The formula for λ in (5) when $P \neq \pm Q$ cannot be used for point doubling because $x_1 = x_2$ in that case and the denominator is 0. Starting with the point addition form of λ, Brier and Joye [7] use a series of algebraic manipulations to obtain a form of λ that is defined for both point addition and point doubling:

$$\lambda = \frac{(x_1 + x_2)^2 - x_1 x_2 + a}{y_1 + y_2}, \quad \text{if } y_1 + y_2 \neq 0 \ . \tag{6}$$

This formula for λ is only defined when $y_1 + y_2 \neq 0$. Brier and Joye subsequently derive a projective formula for point addition using this unified value of λ, with $x_i = X_i/Z_i, y_i = Y_i/Z_i$:

$$X_3 = 2FW \ , \quad Y_3 = R(G - 2W) - L^2 \ , \quad Z_3 = 2F^3 \ , \tag{7}$$

where $U_1 = X_1 Z_2, U_2 = X_2 Z_1, S_1 = Y_1 Z_2, S_2 = Y_2 Z_1, Z = Z_1 Z_2, T = U_1 + U_2, M = S_1 + S_2, F = ZM, L = MF, G = TL, R = T^2 - U_1 U_2 + aZ^2$, and $W = R^2 - G$. This formula requires 13 field multiplications and 5 field squarings.

3.1 Walter's Side-Channel Attack

Walter's side-channel attack [8] is a theoretical attack that assumes the occurrence of a conditional subtraction in a Montgomery modular multiplication operation can be detected. This attack should be considered successful if a non-negligible proportion of the keys can be computed significantly faster than an attack on the whole keyspace. We will see that in some cases, the attack becomes practical as a (relatively) high proportion of keys can be found with (relatively) few computations.

Montgomery's modular reduction technique [6] replaces the modular reduction step after each multiplication with a less expensive step, leaving a result that is not completely reduced. This partial results can be used again in further operations, with the expensive modular reduction step being left for the end. Though Montgomery's technique is not efficient for a single modular multiplication, it is effective when amortized over a long sequence of operations, such as a point multiplication.

Montgomery multiplication is implemented as follows (c.f. [11]). Let q be an odd prime, and let $R > q$ with $\gcd(R, q) = 1$; it is often convenient to choose $R = 2^{Wt}$, where W is the word size of the computer architecture. For an input $z < Rq$, Montgomery reduction gives $zR^{-1} \mod q$. Montgomery modular multiplication is given by Fig. 3.

Input:	Integers q, R, z, q' with $q' = -q^{-1} \bmod R$, $R > q$, $0 \leq z < Rq$.
Output:	Integer c such that $c = zR^{-1} \bmod q$, $0 \leq c < q$.
1.	$c \leftarrow (z + (zq' \bmod R)q)R^{-1}$
2.	if $c \geq q$ then $c \leftarrow c - q$

Fig. 3. Montgomery modular multiplication algorithm

Walter considers the effect of being able to detect the conditional subtraction in step 2 of Fig. 3 in a point multiplication using the unified formula of Brier and Joye. In the projective formula of (7), for a point doubling, the computations of U_1 and U_2 are identical, as are the computations of S_1 and S_2. For a point doubling, the occurrence of a conditional subtraction in the Montgomery multiplication for U_1 must be the same as that for U_2. Thus, if a conditional subtraction is observed in the computation of one of U_1 or U_2 but not the other, then a point doubling could not have occurred and the operation must be a point addition. (The same argument allows the computations of S_1 and S_2 to distinguish a point addition.)

We must now determine the probability p_{dist} of distinguishing a point addition from a point doubling. Walter showed [8, (4)] that, in the multiplication multiplication of two independent randomly distributed residues X and Y with uniformly distributed Z, the probability of a conditional subtraction occurring is

$p_{\text{sub}} = \frac{1}{4}q2^{-Wt}$, where q is the prime and 2^{-Wt} is chosen as in the description of Montgomery modular multiplication above. In practice, q is often very close to 2^{Wt}, so $p_{\text{sub}} \approx 1/4$. For this value, the probability that a conditional subtraction occurs in the computation of one of U_1, U_2 but not the other (and similarly for S_1 and S_2) is

$$p_{\text{diff}} = 2p_{\text{sub}}(1 - p_{\text{sub}}) \approx \frac{3}{8} \ . \tag{8}$$

Hence, the probability that the occurrence of conditional subtractions in the computations of U_1, U_2, S_1, and S_2 can be used to distinguish a point addition from a point doubling is

$$p_{\text{dist}} = 1 - (1 - p_{\text{diff}})^2 \approx \frac{39}{64} \approx 0.61 \ . \tag{9}$$

It is possible to improve this value in certain cases. Walter considers the following scenario: If, over a number of samples of point multiplications, the input point is randomly distributed, then in about one out of 512 samples there will be a input point of the form $P \approx \left(\frac{1}{16}q, \frac{1}{16}q, \frac{15}{16}q\right)$. For a point of this special form, the probability that a point addition can be distinguished from a point doubling is

$$p_{\text{dist}} \approx \frac{188703}{262144} \approx 0.72 \ . \tag{10}$$

In the sequence of operations in a double-and-add point multiplication algorithm, the position of a point addition determines the point doublings on either side of it. Let n be the size in bits of the prime field. Given p_{dist}, the total number of determined operations is:

$$\frac{3}{2}(n-1)p_{\text{dist}} - (n-2)\left(\frac{1}{2}p_{\text{dist}}\right)^2 \ . \tag{11}$$

Walter provides a worked example for the NIST 192-bit prime curve [12]. For this curve, the prime is of a special form $q = 2^{192} - 2^{64} - 1$ and the natural choice for Montgomery multiplication is $2^{Wt} = 2^{192}$ so $q2^{-Wt}$ is very close to 1. Using p_{dist} in (10), the total number of determined operations given by (11) is approximately 181.6 out of an expected total of $\frac{3}{2}(n-1) = 286.5$ operations, leaving 104.9 unknown operations. Of these, approximately $\frac{1}{2}(n-1)(1-p_{\text{dist}}) = 26.8$ are point additions. Walter concludes that a brute-force search through this space of possible keys requires an expected $\binom{104.9}{26.8} \approx 2^{86.8}$ point multiplications.

Walter's analysis provides additional means of decreasing the key space to be searched, including substring restrictions on the possible sequence of point additions and point doublings; furthermore, for the point addition operations which were not distinguished, some combinations of conditional subtractions are more likely than others, and this can also reduce the key space. These two techniques allow the size of the effective keyspace in the example to be reduced to below 2^{56}.

The probabilistic analysis given above does not give the best estimate of the number of determined operations. In experiments, Walter [8] found that, with

a set of 512 samples, it is most efficient to just pick the sample that has the most number of distinguished point additions. This approach, combined with the substring restrictions given above, can give effective keyspaces for a 192-bit prime curve of size just $2^{17.6}$, which can be easily searched.

4 Unified Formulæ of Brier, Déchène, and Joye

4.1 Affine Coordinates

The unified point addition formula of Brier and Joye from the previous section is defined when $y_1 + y_2 \neq 0$, which always holds in the case of point doubling, but it is not applicable to all possible point additions. Izu and Tagaki [9] showed that in some settings these special cases of the point addition could be used to reveal the key. This concern lead Brier, Déchène, and Joye [10] to develop an infinite family of unified point addition formulæ which are defined for all points.

Let $m = m(x_1, y_1; x_2, y_2)$ be an arbitrary polynomial, and define $\tilde{m} = \tilde{m}(x_1, y_1; x_2, y_2) = m(x_2, y_2; x_1, y_1)$. Recall that we are only concerned with elliptic curves over prime fields. Let

$$\lambda_m = \begin{cases} \frac{(x_1+x_2)^2 - x_1x_2 + a + (y_1-y_2)m}{y_1+y_2+(x_1-x_2)m}, & \text{if } y_1+y_2+(x_1-x_2)m \neq 0 \\ \frac{(x_1+x_2)^2 - x_1x_2 + a + (y_2-y_1)\tilde{m}}{y_1+y_2+(x_2-x_1)\tilde{m}}, & \text{if } y_1+y_2+(x_2-x_1)\tilde{m} \neq 0 \end{cases} . \tag{12}$$

These formulæ are defined for all points except those which satisfy

$$y_1 + y_2 + (x_1 - x_2)m = 0 = y_1 + y_2 + (x_2 - x_1)\tilde{m} \ . \tag{13}$$

If $x_1 = x_2$, then $y_1 = -y_2$ and hence $Q = -P$, so the addition formula is not needed. If $x_1 \neq x_2$, then $m + \tilde{m} = 0$. Thus, if we require that m satisfies the condition

$$[(m + \tilde{m} = 0) \Rightarrow (x_1 = x_2)] \ , \tag{14}$$

then λ_m is well-defined. There is an infinite family of equations satisfying this condition:

$$m_k = (x_1 - x_2)^{2k} \ . \tag{15}$$

For efficiency purposes, we can choose $m = m_0 = 1$. In this case, we get the following unified formula for λ:

$$\lambda = \lambda_1 = \frac{(x_1 + x_2)^2 - x_1x_2 + a + (-1)^\delta(y_1 - y_2)}{y_1 + y_2 + (-1)^\delta(x_1 - x_2)}, \quad y_1+y_2+(-1)^\delta(x_1-x_2) \neq 0 \ , \tag{16}$$

where $\delta = 0$ when $y_1 + y_2 + x_1 - x_2 \neq 0$ and $\delta = 1$ otherwise.

Unified point addition using $\lambda = \lambda_1$ requires 2 field multiplications, 2 field squarings, and 1 field inversion.

4.2 Projective Coordinates

To mitigate the high cost of field inversion compared to the cost of field multiplication, points are expressed in projective coordinates so that field inversion need only be done once per point multiplication rather than at each intermediate point addition or point doubling.

We now obtain a projective form of the unified point addition formula given by λ as defined in (16). We begin by noting that since $P + Q = Q + P$, the value for y in (4) is symmetric and hence $2y_3 = \lambda(x_1 + x_2 - 2x_3) - (y_1 + y_2)$. Letting $x_i = X_i/Z$, $y_i = Y_i/Z$ and completing the square in the numerator of λ, we obtain:

$$X_3 = 2FW \ , \quad Y_3 = R(G - 2W) - LFM \ , \quad Z_3 = 2F^3 \ , \tag{17}$$

where $U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2, M = S_1 + S_2, V = (-1)^\delta(U_1 - U_2), N = (-1)^\delta(S_1 - S_2), E = M + V, F = ZE, L = FE, G = LT, R = T^2 - U_1U_2 + Z(aZ + N)$, and $W = R^2 - G$. Note that $\delta = 0$ when $S_1 + S_2 + U_1 - U_2 \neq 0$ and $\delta = 1$ otherwise. This formula requires 16 field multiplications and 3 field squarings.[4]

5 Extending Walter's Attack

5.1 Conditional Modular Reduction Attack

Walter's original attack in Sec. 3.1 assumed that the conditional subtraction at the end of Montgomery multiplication could be detected. We extend this idea by assuming that the conditional subtraction (or addition) at the end of a field addition (or subtraction) can be detected. For field subtraction as given in Fig. 2, the conditional addition is step 2.[5]

We will observe later in this section that there are some modular subtractions in the unified point addition algorithms where, in the case of a point doubling, the arguments are equal and hence the result of the subtraction is zero, i.e. when $a = b$ we are computing $a - b \mod q$ as $a - b = 0$. In this case, a conditional addition in field subtraction is never performed. However, if we observe the occurrence of a conditional addition for the operation $a - b$, then it must be that $b > a$ and hence the operation in question must be a point addition.

[4] The multiplication by $(-1)^\delta$ in the computation of V and N can be implemented with conditional branching (`if` statement).

[5] In implementation, this is common. For example, the OpenSSL [13] library provides a function `BN_mod_sub_quick` which performs exactly the operations in Fig. 2, and similarly for field addition. When reduction is done using the Extended Euclidean Algorithm, as in OpenSSL's `BN_mod_sub` function, and the value to be reduced is strictly between $-q$ and q, the sequence of steps performed is effectively the same as Fig. 2 and includes a conditional addition.

5.2 Effect on affine formulæ of Sec. 4.1

The affine formulæ of Brier, Déchène, and Joye in (16) requires the computation of $y_1 - y_2$ and $x_1 - x_2$. If all of the coordinates are distributed uniformly at random, then the probability that a conditional addition is necessary in the computation of $y_1 - y_2$ is $1/2$, and similarly for $x_1 - x_2$. In this case, the probability that a point addition can be distinguished is $p_{\text{dist}} = 1 - (1 - 1/2)^2 = 3/4$.

We first note that even when additions and doublings cannot be distinguished, a side channel attack will reveal the number of operations performed in the point multiplication. If the key length is known, then knowing the number of operations gives the number of additions (since the number of doublings is fixed by the key length). To simplify the analysis, we will consider only keys of the most common length. This does not mean that the attack cannot work for other key lengths, but rather that it is more difficult to bound the work required to determine the key.

If q is between $3 \cdot 2^{r-1}$ and $3 \cdot 2^r$, then the most common key length is r and occurs for $p_{l=r} \geq 1/3$ of the keys (integers) between 0 and q. This probability is maximal if q is close to 2^{r+1}, in which case $p_{l=r} \approx 1/2$ of the keys have length r. If there are k additions of which k_1 are not identified, then we can consider the key-space to search as the set of sequences of $r - k$ "zeros" and k_1 "ones". These sequences are combined with the identified additions of the double and add sequence to give a list of possible keys (the substring structure will often remove a number of sequences). The number of possible keys is then bounded by $\binom{r-k+k_1}{k_1}$, which in turn is bounded by $\binom{r}{k_1}$.

If we assume that all keys of length r are possible (which is true if $q \geq 2^{r-1} - 1$), the probaiblity that a key of length r uses k additions is $\binom{r}{k}\frac{1}{2^r}$. Given a key with k additions, the probability that k_1 of them are not identified is $\binom{k}{k_1}(p_{\text{dist}})^{k-k_1}(1 - p_{\text{dist}})^{k_1}$. The probability that exactly k_1 additions are not identified in a key of length r is therefore

$$\begin{aligned}
p_{k_1} &= \sum_{k=k_1}^{r} \binom{r}{k}\frac{1}{2^r}\binom{k}{k_1}(p_{\text{dist}})^{k-k_1}(1 - p_{\text{dist}})^{k_1} \\
&= \sum_{k=k_1}^{r} \frac{(1 - p_{\text{dist}})^{k_1}}{2^r}\binom{r}{k_1}\binom{r-k_1}{k-k_1}(p_{\text{dist}})^{k-k_1} \\
&= \frac{(1 - p_{\text{dist}})^{k_1}}{2^r}\binom{r}{k_1}\sum_{i=0}^{r-k_1}\binom{r-k_1}{i}(p_{\text{dist}})^{i} \\
&= \frac{(1 - p_{\text{dist}})^{k_1}}{2^r}\binom{r}{k_1}(1 + p_{\text{dist}})^{r-k_1} \\
&= \binom{r}{k_1}\left(\frac{1 - p_{\text{dist}}}{2}\right)^{k_1}\left(\frac{1 + p_{\text{dist}}}{2}\right)^{r-k_1} \quad . \qquad (18)
\end{aligned}$$

Although the average number of unidentified addition is $(1 - p_{\text{dist}})r/2 = r/8$, some keys will have fewer additions remaining to be identified.

For our 192-bit prime field example curve, we have $r = 191$ and we get an average of 23.9 additions remaining to be identified, so the search space is still quite large. However, the analysis has been done assuming that the additions in a point multiplication are independent. This may not be strictly true as x_1 and y_1 (the x and y coordinates of the base point) are the same for all the additions.

In one-eighth of point multiplications kP, the x-coordinate of the base point P will take on a value between 0 and $\frac{1}{8}q$ and will have an average value of $\frac{1}{16}q$. We take the notation that in the double-and-add point multiplication algorithm the fixed base point P is the first argument to the unified point addition formula. In the computation of $x_1 - x_2$ in (16), over the course of a point multiplication x_2 will be uniformly distributed and thus it is expected for $\frac{15}{16}$ of the point addition operations that $x_2 > x_1$ and a conditional addition occurs. We do not put any condition on the y-coordinate of P and assume that the size of y_1 can be considered independent from the size of x_1. In this case, the probability that a point addition can be distinguished is the probability that a conditional addition occurs in either the computation of $x_1 - x_2$ or $y_1 - y_2$:

$$p_{\text{dist}} = 1 - \left(1 - \frac{15}{16}\right)\left(1 - \frac{1}{2}\right) = \frac{31}{32} \tag{19}$$

Using p_{dist} in (19), the expected number of additions remaining to be identified decreases to 2.99. We can then conclude that a significant proportions of keys of length r will be left with at 3 or fewer unidentified additions (using the distribution in (18), we find that 1 in ≈ 24.6 of all keys satisfy that condition). The number of possible keys is then bounded (loosely) by $\binom{191}{3} \approx 2^{20.1}$.

With point multiplication using the modified Jacobian wNAF algorithm with $w = 5$, the fastest point multiplication technique listed in Table 2, it would take 116.5 minutes on an 900 MHz UltraSPARC III to search a keyspace of size $2^{20.1}$. Hankerson, Menezes, and Vanstone [11, §3.7] provide a survey of point multiplication algorithms with even faster performance.

5.3 Effect on projective formulæ of Sec. 4.2

Just as for affine formulæ, there are two operations in the projective formulæ of (17) where we can take advantage of the ability to detect a conditional addition in a field subtraction. Without loss of generality, suppose $\delta = 0$. Consider the calculations $V = U_1 - U_2$ and $N = S_1 - S_2$. In the case of a point doubling, $U_1 = U_2$ and $S_1 = S_2$, so no conditional addition will occur in the calculation of either V or N. However, in the case of a point addition, U_1 and U_2 will be uniformly distributed over $0, \ldots, p-1$ and so with probability $p_{\text{add}} = \frac{1}{2}$, $U_2 < U_1$ and a conditional addition is needed in the computation of $V = U_1 - U_2$ (similarly for $N = S_1 - S_2$). Moreover, the occurrence of a conditional addition in the computation of V is independent of the occurrence for N. If a conditional addition is observed in at least one of these computations, then the operation is known to be a point addition, revealing the key bit. The probability of distinguishing a point addition is again $1 - (1 - p_{\text{add}})^2 = 3/4$. It should be noted that

taking advantage of base points of a special form is not possible here as U_1, U_2, S_1 and S_2 all depend on both of the points of the addition, so the probabilities of identifying point additions are independent from each other.

If the field is implemented using Montgomery representation, Walter's original attack [8] on detecting conditional subtractions in Montgomery reductions still applies to this projective formula. The detection of a conditional subtraction is used to distinguish a point addition from a point doubling in the computation of U_1 as compared to U_2, and of S_1 as compared to S_2. We can combine the two sources of information (conditional additions in the field subtractions and differences in conditional subtractions in the Montgomery reductions) to increase the probability of success.

We now have four different conditional events which distinguish a point addition from a point doubling:

1. conditional subtraction in computation of one of U_1, U_2 but not the other,
2. conditional subtraction in computation of one of S_1, S_2 but not the other,
3. conditional addition in computation of $N = U_1 - U_2$, and
4. conditional addition in computation of $V = S_1 - S_2$.

Each of these events occurs independently, so the total probability of distinguishing a point addition from a point doubling is

$$p_{\text{dist}} = 1 - (1 - p_{\text{add}})^2 (1 - p_{\text{diff}})^2 \ . \tag{20}$$

If we assume no special knowledge on the base point of the point multiplication, i.e. $p_{\text{add}} = 1/2$ and $p_{\text{diff}} \approx 3/8$, we get $p_{\text{dist}} \approx 231/256 \approx 0.902$. From the distribution obtained in (18), we have an average of $\approx 0.049r$ unidentified additions.

If we look for base points of a special form as in Walter's attack for the formulæ of Brier and Joye, the increase in the probability of success is relatively small. With a point of the form $\sim \left(\frac{1}{16}q, \frac{1}{16}q, \frac{15}{16}q\right)$, we get $p_{\text{diff}} \approx 0.93$ and the expected number of unidentified addition decreases to $\approx 0.035r$. This decrease is small considering that we have to restrict ourselves to 1 in 512 point. In this case, it is much more practical to consider all base points and take advantage of the variability. For example, for a field of 192 bits, 15.4% of all point multiplications have 6 or less unidentified additions, while the special base points (1 in 512) have 6.7 unidentified additions on average.

Table 1 gives estimates for the attack at various field sizes. At each field size, we give the average number of additions remaining to be identified. We also evaluate the costs of the attack when we give a bound on the maximum number of additions remaining to be identified (we chose 3 and 6 as examples) before the key is attacked. In each case, we give the expected number of point multiplications that must be observed before finding such a key and a (loose) upper bound on the size of the remaining keyspace. We assume that general points are considered, so $p_{\text{dist}} \approx 0.902$, and that all keys have size r. To take other key sizes into account, the expected number of keys required should be multiplied by a factor between 2 and 3 (depending on the ratio $q/2^r$) if keys of length other than r are assume to give a failure.

Table 1. Expected number of operations using conditional modular reduction attack, using p_{dist} as in (20).

field size in bits (r+1)	160	192	224	256	384	512
average missing additions:	7.76	9.33	10.89	12.45	18.70	24.95
at most 3 unidentified additions:						
expected number of keys required:	21.8	66.6	217	746	$2^{17.1}$	$2^{25.2}$
bound on the keyspace:	$2^{19.3}$	$2^{20.1}$	$2^{20.8}$	$2^{21.4}$	$2^{23.1}$	$2^{24.4}$
at most 6 unidentified additions:						
expected number of keys required:	2.96	5.82	12.8	30.9	$2^{10.9}$	$2^{17.8}$
bound on the keyspace:	$2^{34.2}$	$2^{35.9}$	$2^{37.2}$	$2^{38.4}$	$2^{41.9}$	$2^{44.4}$

Just as in Walter's analysis [8] it is possible to decrease the key space remaining to be searched, for example by using substring restrictions on the possible sequence of point additions and point doublings.

5.4 Countermeasures to the Conditional Modular Reduction Attack

The success of the Conditional Modular Reduction attack depends on information leaked on the size of intermediate values which is observed based on the field subtraction having a conditional addition. If the field subtraction were to have constant runtime, for example by inserting a dummy addition to offset the conditional addition, then the attack would not apply. However, inserting dummy operations may increase the susceptibility of the algorithm to differential side-channel attacks.

Another countermeasure would be to replace an operation $a-b \mod q$ with a sequence of instructions that determines which of a and b is larger, subtracts the smaller from the larger, and then flips the sign of the result. In most software implementations, the sign of a number is stored in a different register than the value of the number, so flipping the sign of a number is a single machine instruction; the difficulty is now in detecting a single instruction compared to a multi-precision integer addition.

A third countermeasure consist in taking the field reductions (both Montgomery reductions and addition/subtraction of a multiple of q) as independent operations from the multiplications, squarings, additions and subtractions and rewrite the unified formulæ in consequence. This means we will accept that some of the values used during the computations may be greater than q. Although this approach removes any danger of an attack based on variations in the field arithmetic, it may have a negative impact on the efficiency, in particular when the field size is close to a multiple of the word size.

At this point we should also note that choosing $\delta = 0$ or $\delta = 1$ requires the comparison of two field elements, so at least these two must be fully reduced. Since repeated operations or even a change from addition to subtraction could

potentially lead to an attack, we choose δ to ensure that $\frac{1}{2}(x_1 + x_2 + y_1 + y_2) \neq x_{2-\delta}$ instead of $y_1 + y_2 + (-1)^\delta(x_1 - x_2) \neq 0$ (the two conditions are equivalent, but the computations required for the first test are more uniform).

For simplicity, we will assume the field is in Montgomery representation. We distinguish Montgomery reductions (denoted $MontRed(\cdot)$) and q-reduction where multiples of q are added/subtracted (denoted $reduce(\cdot)$). To avoid all possible extensions of the attack, we prefer to err on the side of caution and implement the field operations as follows:

- Products (and squares) are not reduced unless stated.
- Sums are not reduced unless stated.
- Subtractions never contain a conditional addition (a fixed multiple of q is always added to the first operand before doing the subtraction).
- If an integer is to be q-reduced, then it is at least as large as q.
- For the affine formula, inversion accepts any integer between 1 and $6q - 1$ and coprime to q and returns an integer between 1 and $q - 1$.
- Montgomery reductions are allowed to return an output between 0 and $2q-1$. They accept inputs between 0 and $6q^2$ for affine coordinates ($R > 6q$) and between 0 and $16q^2$ for projective coordinates ($R > 16q$).
- The multiples of q used in the formulæ are pre-computed.

For the affine formula, we get the algorithm in Fig. 4. For the projective formula, we let the X, Y and Z coordinates be in the range $[0, 2q - 1]$ and obtain the algorithm in Fig. 5. Note that by construction $(x_1 + x_2)^2 \geq x_1 x_2$, so the pre-reduction result in step 8 of the affine formula is indeed positive (similarly for step 14 in the projective formula)

Input:	Points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq Q$.
Output:	Point $(x_3, y_3) = P + Q$.
1.	$E \leftarrow x_1 + y_1 + y_2 + x_2$
2.	if E is odd, then $F \leftarrow E + 3q$, else $F \leftarrow E + 2q$
3.	$G \leftarrow reduce(F/2)$
4.	if $G \neq x_2$, then $\delta \leftarrow 0$, else $\delta \leftarrow 1$
5.	$H \leftarrow F - 2x_{2-\delta}$
6.	$I \leftarrow H^{-1} \mod q$
7.	$J \leftarrow x_1 + x_2$
8.	$K \leftarrow MontRed(J^2 - x_1 \cdot x_2)$
9.	$L \leftarrow reduce(2q + K + a + y_{1+\delta} - y_{2-\delta})$
10.	$\lambda = MontRed(L \cdot I)$
11.	$M \leftarrow MontRed(\lambda^2)$
12.	$x_3 \leftarrow reduce(3q + M - x_1 - x_2)$
13.	$N \leftarrow MontRed(\lambda \cdot (q + x_1 - x_3))$
14.	$y_3 \leftarrow 2q + N - y_1$

Fig. 4. Affine coordinates uniform point addition formula

Input: Points $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$, $P \neq Q$.
Output: Point $(X_3, Y_3, Z_3) = P + Q$.

1. $\hat{U}_1 \leftarrow X_1 \cdot Z_2$, $\hat{U}_2 \leftarrow X_2 \cdot Z_1$, $\hat{S}_1 \leftarrow Y_1 \cdot Z_2$, $\hat{S}_2 \leftarrow Y_2 \cdot Z_1$
2. $\hat{T} = \hat{U}_1 + \hat{U}_2$
3. $M \leftarrow MontRed(\hat{S}_1 + \hat{S}_2 + \hat{T})$
4. if M is odd, then $V \leftarrow M + 3q$, else $V \leftarrow M + 2q$
5. $C \leftarrow reduce(V/2)$
6. $B \leftarrow reduce(p + MontRed(\hat{U}_2))$
7. if $C \neq B$, then $\delta \leftarrow 0$, else $\delta \leftarrow 1$
8. $E \leftarrow V - MontRed(2\hat{U}_{2-\delta})$
9. $Z \leftarrow MontRed(Z_1 \cdot Z_2)$, $T \leftarrow MontRed(\hat{T})$
10. $F \leftarrow MontRed(Z \cdot E)$, $U_1 \leftarrow MontRed(\hat{U}_1)$, $U_2 \leftarrow MontRed(\hat{U}_2)$
11. $L \leftarrow MontRed(F \cdot E)$
12. $G \leftarrow MontRed(L \cdot T)$
13. $K \leftarrow 2q + MontRed(a \cdot Z + \hat{S}_{1+\delta}) - MontRed(\hat{S}_{2-\delta})$
14. $R \leftarrow MontRed(Z \cdot K + T^2 - U_1 \cdot U_2)$
15. $H \leftarrow MontRed(R^2)$
16. $W \leftarrow reduce(3q + H - G)$
17. $X_3 \leftarrow MontRed((2F) \cdot W)$
18. $J \leftarrow MontRed(F^2)$
19. $Z_3 \leftarrow MontRed((2F) \cdot J)$
20. $N \leftarrow MontRed(F \cdot M)$
21. $Y_3 \leftarrow 2q - MontRed(R \cdot (2q + 2W - G) + L \cdot N)$

Fig. 5. Projective coordinates uniform point addition formula

6 Timing

The timings in this section were performed on a 900 MHz UltraSPARC III using the multi-precision integer and elliptic curve libraries from NSS 3.9 [14] with no optimized assembly code. To obtain high-resolution timings, we used the Solaris `hrtime` C library, which has a resolution of 100 ns. We use the 160-bit prime field curve `secp160r2` [15].

On our test system, the average time of a 160-bit prime field modular subtraction $a - b \mod q$ when $a > b$ is about 320 ns. When $a < b$, and hence when a conditional addition is required, the average time is about 550 ns.

Table 2 gives performance timings for point operations using the unified point addition and doubling formulæ given in this paper as well as other schemes. Point multiplications for all fomulæ except Jacobian projective and modified Jacobian wNAF use the double-and-add technique. The timings in the table are the average of 10^5 operations.

Table 3 gives average timings and standard deviations for point additions and point doublings in the course of a single point multiplication. The results

Table 2. Average point operation timings for `secp160r2` curve.

Formula	Addition	Doubling	Multiplication
BDJ affine, $m = 1$	126.5 μs	126.2 μs	29.03 ms
BDJ projective	58.9 μs	58.5 μs	13.99 ms
BJ projective	49.8 μs	49.5 μs	11.76 ms
Affine	115.7 μs	118.4 μs	27.89 ms
Jacobian projective			7.95 ms
Modified Jacobian wNAF, $w = 5$			6.22 ms

were obtained by recording the time of each addition or doubling in a single point multiplication using the double-and-add algorithm.

Table 3. Individual point operation timings from a single point multiplication for `secp160r2` curve.

Formulæ	Operation	Average	Standard Deviation
BDJ affine, $m = 1$	unified addition	126.528 μs	4.094 μs $\approx 3.2\%$
	unified doubling	126.155 μs	3.700 μs $\approx 2.9\%$
	difference	0.373 μs $\approx 0.3\%$	
BDJ projective	unified addition	58.992 μs	0.474 μs $\approx 0.8\%$
	unified doubling	59.307 μs	0.448 μs $\approx 0.75\%$
	difference	0.315 μs $\approx 0.53\%$	

In the top half of Table 3, timings are given for point addition and doubling using the affine formulæ of Brier, Déchène, and Joye. A unified doubling takes slightly less time than a unified addition on average, but difference between the two operations (0.3%) is one-tenth the size of the standard deviation of either operation, so the timings of the two operations cannot be reliably distinguished.

In the bottom half of Table 3, timings are given for point addition and doubling using the projective formulæ developed in Sec. 4.2. A unified doubling takes slightly more time (0.53%) than a unified addition. The standard deviation of either operation, at 0.8% for addition and 0.75% for doubling, is less than twice difference.

For the affine formulæ, unified addition is slower than unified doubling, but the situation is reversed for the projective formulæ. This phenomenon may be particular to the compiler and processor used.

For both the affine and projective formulæ of Brier, Déchène, and Joye in Table 3, the average difference in timing between a point addition and point doubling is too small compared the standard deviation to be of practical use

on its own. However, we do not dismiss the fact that this information may be helpful when combined with other side-channel information.

Acknowledgments

We are grateful to I. Déchène for her insight on unified formulæ and other helpful comments. We also thank N. Gura of Sun Microsystems Laboratories for his advise on how to perform high-resolution timings on Solaris. D.S. is grateful of the hospitality of S. Chang of Sun Microsystems Laboratories where he was resident during this research.

References

1. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48** (1987) 203–209
2. Miller, V.: Use of elliptic curves in cryptography. In Williams, H.C., ed.: Advances in Cryptology – Proc. CRYTPO '85. Volume 218 of LNCS., Springer-Verlag (1986) 417–428
3. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz, N., ed.: Advances in Cryptology – Proc. CRYPTO '96. Volume 1109 of LNCS., Springer-Verlag (1996) 104–113
4. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In Wiener, M., ed.: Advances in Cryptology – Proc. CRYPTO '99. Volume 1666 of LNCS., Springer-Verlag (1999) 388–397
5. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin K. Koç, Paar, C., eds.: Cryptographic Hardware and Embedded Systems (CHES) '99. Volume 1717 of LNCS., Springer-Verlag (1999) 292–302
6. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation **44** (1985) 519–521
7. Brier, É., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In Naccache, D., Paillier, P., eds.: Public Key Cryptography – PKC 2002. Volume 2274 of LNCS., Springer-Verlag (2002) 335–345
8. Walter, C.D.: Simple power analysis of unified code for ECC double and add. In Joye, M., Quisquater, J.J., eds.: Crytpographic Hardware and Embedded Systems (CHES) 2004. Volume 3156 of LNCS., Springer-Verlag (2004) 191–204
9. Izu, T., Takagi, T.: On the Security of Brier-Joye's Addition Formula for Weierstrass-form Elliptic Curves Technical Report, Technische Universität Darmstadt, Available online: http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/
10. Brier, É., Déchène, I., Joye, M.: Unified point addition formulæ for elliptic curve cryptosystems. In Nedjah, N., de Macedo Mourelle, L., eds.: Embedded Cryptographic Hardware: Methodologies and Architectures. Nova Science Publishers (2004) 247–256
11. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag (2004)
12. National Institute of Standards and Technology: Recommended elliptic curves for federal government use (1999) Available online: http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf.
13. OpenSSL Project: OpenSSL v0.9.8 (2005) Available online: http://www.openssl.org/.
14. Mozilla Foundation: Netscape Security Services (NSS) v3.9 (2005) Available online: http://www.mozilla.org/projects/security/pki/nss/.
15. Certicom Research: SEC 2: Recommended elliptic curve domain parameters (2000) Available online: http://www.secg.org/.
16. Hankerson, D., Hernandez, J.L., Menezes, A.: Software implementation of elliptic curve cryptography over binary fields. In Çetin K. Koç, Paar, C., eds.: Crytpographic Hardware and Embedded Systems (CHES) 2000. Volume 1965 of LNCS., Springer-Verlag (2000) 1–24

A Elliptic Curves over Binary Fields

A.1 Background

For fields of characteristic 2, the equation of an elliptic curve can be simplified to:

$$E_{/\mathbb{K}} : y^2 + xy = x^3 + ax^2 + b \ . \tag{21}$$

Two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ can be added to obtain a third point $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \tag{22}$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \tag{23}$$

$$\lambda = \begin{cases} \frac{y_1+y_2}{x_1+x_2}, & \text{if } P \neq Q \\ x_1 + \frac{y_1}{x_1}, & \text{if } P = Q \end{cases} \ . \tag{24}$$

A.2 Unified Formula of Sec. 3

The unified form of λ for point addition and doubling for curves over binary fields is:

$$\lambda = \frac{(x_1 + x_2)^2 + x_1x_2 + a(x_1 + x_2) + y_1}{y_1 + y_2 + x_2}, \quad \text{if } y_1 + y_2 + x_2 \neq 0 \ . \tag{25}$$

Point addition or doubling using the affine form requires 1 inversion, 4 multiplications, and 2 squarings; using a projective form requires 20 multiplications and 3 squarings. In practice, a binary field inversion has the same computational cost as about 10 field multiplications [16], so the affine form is faster than the projective form.

A.3 Unified Formulæ of Sec. 4

Let

$$\lambda_m = \begin{cases} \frac{(x_1+x_2)^2+x_1x_2+y_2+a(x_1+x_2)+(y_1+y_2)m}{y_1+y_2+x_2+(x_1+x_2)m}, & \text{if } y_1 + y_2 + x_2 + (x_1 + x_2)m \neq 0 \\ \frac{(x_1+x_2)^2+x_1x_2+y_1+a(x_1+x_2)+(y_1+y_2)\tilde{m}}{y_1+y_2+x_2+(x_1+x_2)\tilde{m}}, & \text{if } y_1 + y_2 + x_1 + (x_1 + x_2)\tilde{m} \neq 0 \end{cases} . \tag{26}$$

These formulæ are defined for all points except those which satisfy

$$y_1 + y_2 + x_2 + (x_1 + x_2)m = 0 = y_1 + y_2 + x_1 + (x_1 + x_2)\tilde{m} \ . \tag{27}$$

For efficiency purposes, we can choose $m = m_0 = 0$. In this case, we get the following unified formula for λ:

$$\lambda = \lambda_0 = \begin{cases} \frac{(x_1+x_2)^2+x_1x_2+a(x_1+x_2)+y_2}{x_1+y_1+y_2}, & \text{if } x_1 + y_1 + y_2 \neq 0 \\ \frac{(x_1+x_2)^2+x_1x_2+a(x_1+x_2)+y_1}{x_2+y_1+y_2}, & \text{if } x_2 + y_1 + y_2 \neq 0 \end{cases} . \tag{28}$$

Unified point addition using $\lambda = \lambda_0$ requires 4 field multiplications, 2 field squarings, and 1 field inversion.

A.4 Unified Formulæ of Sec. 4.2

We now obtain a projective form of the unified point addition formula given by λ as defined in (28). Letting $x_i = X_i/Z_i$, $y_i = Y_i/Z_i$ and completing the square in the numerator of λ, we obtain:

$$X_3 = FW \ , \quad Y_3 = R(LU + W) + X_3 + HES \ , \quad Z_3 = FH \ , \qquad (29)$$

where $U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2, M = S_1 + S_2, E = M + U_{1+\delta}, F = ZE, L = FE, G = LT, H = F^2, R = T^2 + U_1U_2 + Z(aT + S_{2-\delta}), K = FR + G + aH$, and $W = R^2 + K$. Note that $\delta = 0$ when $M \neq U_1$ and $\delta = 1$ otherwise. These terms were derived using $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ but by symmetry of point addition could have equally been derived using $y_3 = \lambda(x_2 + x_3) + x_3 + y_2$.

This projective formula requires 19 field multiplications and 3 field squarings. In practice, a binary field inversion has the same computational cost as about 10 field multiplications [16] so the affine form of the unified point addition formulæ is faster than the projective form.

A.5 Application of Attack of Sec. 5

For the projective formula for unified point addition for curves over binary fields as given in (29), Walter's attack does not apply nor does our extension in Sec. 5.1. Walter's original attack does not apply because field multiplication in binary fields is not implemented using Montgomery multiplication. Our extension does not apply because no modular reduction operations are necessary in field addition.