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, timing, or electromagnetic emissions, has motivated the recent development of unified formulæ for elliptic curve point operations. In this paper, we show how an attack introduced by Walter can be improved and used against the unified formulæ of Brier, Déchène and Joye when it relies on a standard field arithmetic implementation, both in affine and projective coordinates. We also describe how the field arithmetic might be implemented to obtain more uniform operations that 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. In recent years, a new class of attacks has been developed, called *side-channel* attacks [3], which use information observed during the execution of the algorithm to help to 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 attack in this paper is only considered in a *simple* side channel context.

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

* Supported by NSERC, Sun Microsystems, CIAR, MITACS, CFI, and ORDCF.

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. 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 [3], power analysis [4], and electromagnetic emissions [5]. Techniques for counteracting this problem include: performing dummy operations, such as forcing a point addition at each iteration [6]; using alternate point multiplication algorithms, such as Montgomery point multiplication [7]; 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 for elliptic curves in Weierstraß form, in which point addition and point doubling use the same sequence of field operations, was first given by Brier and Joye [8] in affine and projective form. Walter [9] demonstrated a theoretical side-channel attack on an implementation of 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 [11] 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 is often an integral step of field subtraction. A typical algorithm for computing prime field subtraction is given in Fig. 1; the conditional addition is step 2.[3]

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

Fig. 1. Field subtraction algorithm

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

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

attack. This observation reinforces the fact that a secure implementation requires constant-runtime field operations, not just unified point arithmetic. In fact, security against side-channel attacks needs to be addressed at three levels: the hardware level, the software level, and the algorithmic level.

We also provide some performance results for the various unified formulæ and discuss the applicability of timing attacks. We find in timing experiments that the runtime of a field subtraction with the conditional addition takes substantially longer than without (520 ns versus 330 ns) and thus seems exploitable.

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 implementation of 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.

The attacks we discuss in this paper only apply to elliptic curves over prime fields and do not apply to curves over binary fields. The countermeasures we present are not intended to be secure against differential side channel attacks; standard countermeasures for that context should still be applied.

2 Background

For fields $\mathbb{K}$ of prime characteristic other than 2 or 3, the Weierstaß form of an elliptic curve is given by the equation $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{K}$. The set of points in $\mathbb{K} \times \mathbb{K}$ on the curve, joined with the *point at infinity* $\mathcal{O}$, forms an abelian group, denoted $E(\mathbb{K})$. 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 $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \text{ (addition)} \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \text{ (doubling)} \end{cases} . \tag{1}$$

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 the ordinary projective case, a point is represented by three coordinates, $P = (X, Y, Z)$, with $x = X/Z$ and $y = Y/Z$. Denominators are used for 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.

3 Unified Formula of Brier and Joye

The formula for λ in (1) when $P \neq Q$ 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 [8] use a series of algebraic manipulations to obtain a form of λ that is defined for both point addition and point doubling:

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

However, this formula for λ is not defined when $y_1 + y_2 = 0$ (see Section 4.1). Brier and Joye subsequently derive a projective formula for point addition using this unified value of λ, with $x_i = X_i/Z_i, y_i = Y_i/Z_i$:

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

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 [9] is an 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 they would with 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.

Walter considers the effect of being able to detect a conditional subtraction in Montgomery modular reductions in a point multiplication using the unified formula of Brier and Joye. For a point doubling using the projective formula of (3), the computations of U_1 and U_2 are identical, as are the computations of S_1 and S_2. The occurrence of a conditional subtraction in the Montgomery multiplication for U_1 must be the same as that for U_2, for a point doubling. 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.) 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

$$p_{\text{diff}} = 2p_{\text{sub}}(1 - p_{\text{sub}}) \approx \frac{3}{8} \ . \tag{4}$$

where p_{sub} is the probability of a conditional subtraction occurring; for Montgomery modular reduction in practice, usually $p_{\text{sub}} \approx 1/4$. 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

$$p_{\text{dist}} = 1 - (1 - p_{\text{diff}})^2 \approx \frac{39}{64} \approx 0.61 \ . \tag{5}$$

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 expected total number of determined operations is:

$$\frac{3}{2}(n-1)p_{\text{dist}} - (n-2)\left(\frac{1}{2}p_{\text{dist}}\right)^2 \ . \tag{6}$$

The probabilistic analysis given above does not give the best estimate of the number of determined operations. In experiments, Walter found that, with a set of 512 samples, it is most efficient to just pick the sample that has the greatest number of distinguished point additions. This approach, combined with additional substring restrictions, can give effective keyspaces for a 192-bit prime curve of size just $2^{17.6}$, which can be easily searched. The analysis in Sec. 5 gives a probabilistic argument that generalizes Walter's experimental sampling.

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 [10] showed that in some settings these special cases of the point addition could be used to reveal the key. Brier, Déchène, and Joye [11] developed an infinite family of unified point addition formulæ which are defined for all points. We are most concerned with the most efficient formula of the family, which has

$$\lambda = \frac{(x_1 + x_2)^2 - x_1 x_2 + a + (-1)^\delta (y_1 - y_2)}{y_1 + y_2 + (-1)^\delta (x_1 - x_2)}, \quad y_1 + y_2 + (-1)^\delta (x_1 - x_2) \neq 0 \ , \tag{7}$$

where $\delta = 0$ when $y_1 + y_2 + x_1 - x_2 \neq 0$ and $\delta = 1$ otherwise (or a randomized choice of δ when both choices give nonzero values). Unified point addition using this λ requires 2 field multiplications, 2 field squarings, and 1 field inversion. Although Brier, Déchène, and Joye give an infinite family of unified point addition formulæ, which would allow a different λ value to be randomly chosen at each point addition, we assume that the most efficient member, given in (7), is used for each operation. If any fixed λ is used, then it may be that the attack in Sec. 5 can still be applied.

4.2 Projective Coordinates

To mitigate the high cost of field inversion compared to the cost of field multiplication, points can be 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 describe an ordinary projective form of the unified point addition formula given by λ as defined in (7). We begin by noting that since $P+Q = Q+P$, the value for y_3 in point addition 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:

$$X_3 = 2FW \quad , \quad Y_3 = R(G - 2W) - LFM \quad , \quad Z_3 = 2F^3 \quad , \tag{8}$$

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 (or a randomized choice of δ when both choices give nonzero values). 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. Under the same assumption that a conditional subtraction (or addition) can be observed, we note that such an operation at the end of a field addition (or subtraction) can be detected. For field subtraction as given in Fig. 1, 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: when $c = d$, we compute $c - d \mod q$ as $c - d = 0$. 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 $c - d \mod q$, then it must be that $d > c$ 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 [14] library provides a function `BN_mod_sub_quick` which performs exactly the operations in Fig. 1, 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. 1 and includes a conditional addition.

5.2 Effect on affine formulæ of Sec. 4.1

The affine formulæ of Brier, Déchène, and Joye in (7) 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 identified 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 assume that the attack can only be successful for 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 "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 $\binom{r}{k_1}$ (this estimate is usually very pessimistic, but it has the advantage of being independent from the value of k).

If we assume that all keys of length r are possible (which is true if $q \geq 2^{r+1} - 1$), the probability 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

$$
\begin{aligned}
p_{k_1} &= \sum_{k=k_1}^{r} \binom{r}{k} \frac{1}{2^r} \binom{k}{k_1} (p_{\text{dist}})^{k-k_1} (1 - p_{\text{dist}})^{k_1} \\
&= \sum_{k=k_1}^{r} \frac{(1 - p_{\text{dist}})^{k_1}}{2^r} \binom{r}{k_1} \binom{r-k_1}{k-k_1} (p_{\text{dist}})^{k-k_1} \\
&= \frac{(1 - p_{\text{dist}})^{k_1}}{2^r} \binom{r}{k_1} \sum_{i=0}^{r-k_1} \binom{r-k_1}{i} (p_{\text{dist}})^{i} \\
&= \binom{r}{k_1} \left(\frac{1 - p_{\text{dist}}}{2}\right)^{k_1} \left(\frac{1 + p_{\text{dist}}}{2}\right)^{r-k_1} .
\end{aligned}
\qquad (9)
$$

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. This analysis assumes that the additions in a point multiplication are independent. This is not 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 $1/m$ of point multiplications sP, the x-coordinate of the base point P will take on a value between 0 and $\frac{1}{m}q$ and will have an average value of $\frac{1}{2m}q$. We take the notation that in the double-and-add point multiplication algorithm the fixed base point P is the first argument of the unified point addition formula. In the computation of $x_1 - x_2$ in (7), we assume that, over the course of a point multiplication, x_2 will behave as if it is uniformly distributed. Thus it is expected for $1 - \frac{1}{2m}$ 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 $y_1 - y_2$:

$$p_{\text{dist}} = 1 - \left(1 - \left(1 - \frac{1}{2m}\right)\right)\left(1 - \frac{1}{2}\right) = 1 - \frac{1}{4m} \tag{10}$$

Using p_{dist} in (10) with $1/m = 1/8$, the expected number of additions in our example 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 (9), we find that 1 in ≈ 24.6 of all keys satisfy that condition). The number of possible keys is then bounded (loosely) by $\binom{191}{3} \approx 2^{20.1}$, for which an exhaustive search is quite feasible.

5.3 Effect on projective formulæ of Sec. 4.2

Just as for affine formulæ, there are two operations in the projective formulæ of (8) 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, we assume that U_1 and U_2 will behave as if they are independent and uniformly distributed over $0, \ldots, p-1$. 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, we also assume that 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 for all practical purposes the probabilities of identifying point additions are independent from each other.

If the field is implemented using Montgomery representation, Walter's original attack [9] 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 compared to U_2, and of S_1 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 $V = U_1 - U_2$, and
4. conditional addition in computation of $N = S_1 - S_2$.

Under the assumption that these events occur independently, the probability of detecting a point addition given that the operation was a point addition is

$$p_{\text{dist}} = 1 - (1 - p_{\text{add}})^2 (1 - p_{\text{diff}})^2 \ . \tag{11}$$

In practice, the coordinate values observed during a point multiplication do seem to behave as if they are sufficiently uniformly distributed and, with respect to the four conditional events above, sufficiently uncorrelated.

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 $p_{\text{dist}} \approx 231/256 \approx 0.902$. For the distribution obtained in (9), we average $\approx 0.049r$ 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 points. 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 give a probabilistic analysis of the best sample we expect to find in the trace of 512 random point multiplications.[6] In the probabilistic analysis, we determine an upper bound on the number of unidentified additions for which the attack will be considered successful, requiring a probability of success of at least 1 in 512. For each case, we give a (loose) upper bound on the keyspace and a better estimate using the approach described in Appendix A.

We also evaluate the costs of the attack when we give a bound on the maximum number of additions remaining to be identified before the key is attacked

[6] While the theoretical analysis in Section 5.2 only considers traces in which the key has the most common length, the "best of 512" analysis takes into account keys of all lengths (assuming failure of keys of length other than r).

(with 3 as an example). In this 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. It is interesting to note that for the 521-bit case, the expected number of keys required to obtain 3 unidentified additions approximately balances the bound on the keyspace giving, in some sense, an overall minimized complexity of attack. We also compare our results with those of Walter [9]. We assume that general points are considered, so $p_{\text{dist}} \approx 0.902$, that half the keys have size r (that is, $q \approx 2^{r+1}$) and that an attack on a key of size different from r is always considered unsuccessful.

Table 1. Expected number of operations using conditional modular reduction attack, using p_{dist} as in (11).

Field size in bits $(r+1)$	**160**	**192**	**224**	**256**	**384**	**521**
Average missing additions per point mult.:	7.76	9.33	10.89	12.45	18.70	25.43
Sampling best trace from 512 samples:						
k required for prob. of success $> 1/512$:	2	2	3	4	8	13
Upper bound on keyspace for this k:	$2^{13.6}$	$2^{14.1}$	$2^{20.8}$	$2^{27.4}$	$2^{53.2}$	$2^{84.5}$
Estimated keyspace (Appendix A):	$2^{9.03}$	$2^{9.67}$	$2^{14.3}$	$2^{18.9}$	$2^{37.2}$	$2^{59.4}$
To obtain at most 3 unidentified additions:						
Expected number of keys required:	22	67	217	746	$2^{17.1}$	$2^{25.2}$
Bound on keyspace:	$2^{19.3}$	$2^{20.1}$	$2^{20.8}$	$2^{21.4}$	$2^{23.1}$	$2^{24.4}$
Walter's attack [9]:						
Average missing additions:		19.2	23.0	26.6	41.5	57.9
Bound on keyspace (no restrictions):		$2^{33.2}$	$2^{42.8}$	$2^{52.4}$	$2^{91.4}$	$2^{134.3}$
Bound using substring restrictions:		$2^{17.6}$	$2^{24.0}$	$2^{30.4}$	$2^{56.0}$	$2^{84.2}$

Just as in Walter's analysis it may be 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. This approach has only a limited impact in our case, since the operations remaining to be identified consist in a large number of doublings and a few additions.

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 create additional risks in the setting of differential side-channel attacks.

A nicer countermeasure would be to program the subtraction of $c-d \mod q$ as $(2q+c-d)-mq$, where $m \in \{1,2\}$ depends on the value of $2q+c-d$, so the time required for a field subtraction is constant. For the field addition, one would replace $c+d \mod q$ by $(q+c+d)-mq$ where $m \in \{1, 2\}$, and similarly for Montgomery reduction (replacing $c+d$ by the value of the Montgomery reduction at the moment the conditional subtraction is used). These countermeasures still require a conditional operation to be performed, based on the appropriate value of m, but may have less of a detectable difference.

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 and q-reduction where multiples of q are added/subtracted. In an attempt to avoid possible extensions of the attack, we 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 fully 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 precomputed.

For the projective formula, we let the X, Y and Z coordinates be in the range $[0, 2q - 1]$. Note that by construction $(x_1 + x_2)^2 \geq x_1 x_2$.

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 [15] 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 [16].

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 from Section 4 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 were obtained by recording the time of each addition or doubling in a single point multiplication using the double-and-add algorithm.

Table 2. Average point operation timings for secp160r2 curve.

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

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

Formulæ	Operation	Average	Standard Deviation
BDJ affine	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 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 to the standard deviation to be of practical use on its own. However, we do not dismiss the fact that this information could be helpful when combined with other side-channel information.

7 Future Work

The major drawback with our approach for the analysis is that we concentrate on keys with a minimal number of unidentified additions, not necessarily on keys for which the remaining keyspace is minimized. In general, binary keys where the zeros only appear in small groups in the key are much easier to break than keys with large groups of consecutive zeros since the first and last doubles coming from a string of zeros in the binary representation are likely to be identified and this is almost the same as identifying one of the zeros. For example, the 521-bit key $10101\cdots0101$ with 16 unidentified additions has a much smaller remaining keyspace than the 521-bit key consisting of 261 ones followed by 260 zeros for which only 8 of the additions have not been identified, even though the number of additions remaining to be located is divided by two in the second key.

A cost analysis that includes this idea would require a study of the distribution of strings of zeros in the binary representation of the key, taking into account the effect of unidentified additions. This is clearly beyond the scope of the work presented here.

Acknowledgments The authors wish to acknowledge the assistance of I. Déchène of the University of Waterloo, N. Gura and S. Chang of Sun Microsystems Labs, and the anonymous referees.

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. LNCS, Vol. 218. 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. LNCS, Vol. 1109. 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. LNCS, Vol. 1666. Springer-Verlag (1999) 388–397

5. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-Channel(s). In B.S. Kaliski Jr. and Ç.K. Koç and C. Paar, eds.: Cryptographic Hardware and Embedded Systems – CHES 2002. LNCS, Vol. 2523. Springer–Verlag (2003), 29–45
6. 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. LNCS, Vol. 1717. Springer-Verlag (1999) 292–302
7. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation **44** (1985) 519–521
8. Brier, É., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In Naccache, D., Paillier, P., eds.: Public Key Cryptography – PKC 2002. LNCS, Vol. 2274. Springer-Verlag (2002) 335–345
9. Walter, C.D.: Simple power analysis of unified code for ECC double and add. In Joye, M., Quisquater, J.J., eds.: Cryptographic Hardware and Embedded Systems (CHES) 2004. LNCS, Vol. 3156. Springer-Verlag (2004) 191–204
10. 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/
11. 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
12. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag (2004)
13. 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.
14. OpenSSL Project: OpenSSL v0.9.8 (2005) Available online: http://www.openssl.org/.
15. Mozilla Foundation: Netscape Security Services (NSS) v3.9 (2005) Available online: http://www.mozilla.org/projects/security/pki/nss/.
16. Certicom Research: SEC 2: Recommended elliptic curve domain parameters (2000) Available online: http://www.secg.org/.
17. 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. LNCS, Vol. 1965. Springer-Verlag (2000) 1–24

A Cost Estimates

A.1 Zeros in the Binary Expansion

As our attack looks for keys with few unidentified additions, it introduces a bias in the expected number of zeros of the binary representation of successful keys (increasing its value). This is because keys with fewer ones need fewer identified additions to be successful, which makes them more likely to work then keys that have more ones.

The solutions is to go back to the probabilities of Section 5.3 and to compute the expected number of zeros under the condition that the attack is successful.

Computations for the key sizes considered in Table 1 show the increase to be less than 5% of $r/2$ (the average expected value).

Also, the attack is considered successful for keys with fewer than k unidentified additions, so the expected number of unidentified additions is slightly lower than k, and once again the expected value can be found by going back to the probabilities. Both of these numbers (which also give the expected number of identified additions) are taken into account to produce the estimated keyspace.

A.2 Unidentified Substrings

To estimate the number of unidentified doublings, we introduce an approach that could also be used to detail the lengths of the unidentified substrings.

We will consider the probabilities that would occur for a key of infinite length that has the same proportion of zero bits and identified additions as our finite key. These proportions will give us an estimate on the number of identified (and unidentified) doublings in the finite key.

We use a state diagram consisting of six states, illustrated in Figure 2.

- The first three are doublings coming from moving from one bit of the key to the next: D_* (unidentified), D_1 (identified, preceding an identified addition), and D_2 (identified, following but not preceding an identified addition);
- The remaining three correspond to bit operations (that is, depending on the value of the bit): A (identified addition), A_* (unidentified addition), and V (absence of addition, for the bit "zero").

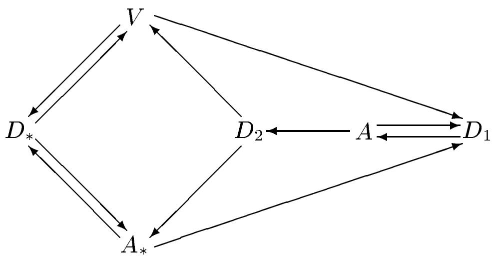

Fig. 2. State diagram

To estimate the number of identified and unidentified doublings, we must find which proportion of the doublings (r in total for the finite key) are in states D_1 and D_2. Every time we are in state D_1, the next state must be A, so the time spent in D_1 is the same as the time spent in A, that is, it corresponds to the number of identified additions. The only way to enter state D_2 is from state A, and the probability of moving to D_2 when leaving A is the probability that the next bit does not correspond to an identified addition.

If the r-bit key has m identified additions, then we can estimate the number of identified doubling as $\left(1+\frac{r-m}{r}\right)m$ (the 1 is for moving from D_1 to A and $\frac{r-m}{r}$ is the probability of moving to D_2 when leaving A). The estimated costs in Table 1 are then straightforward to obtain.