Simple Power Analysis of Unified Code for ECC Double and Add

Colin D. Walter

Comodo Research Laboratory
10 Hey Street, Bradford, BD7 1DQ, UK
Colin.Walter@comodogroup.com

Abstract. Classical formulae for point additions and point doublings on elliptic curves differ. This can make a side channel attack possible on a single ECC point multiplication by using simple power analysis (SPA) to observe the different times for the component point operations. Under the usual binary exponentiation algorithm, the deduced presence or absence of a point addition indicates a 1 or 0 respectively in the secret key, thus revealing the key in its entirety.
Several authors have produced unified code for these operations in order to avoid this weakness. Although timing differences are thereby eliminated from this code level, it is shown that SPA attacks may still be possible on selected single point multiplications if there is sufficient side channel leakage at lower levels. Here a conditional subtraction in Montgomery modular multiplication (MMM) is assumed to give such leakage, but other modular multipliers may be equally susceptible to attack.
The techniques are applicable to a *single* decryption or signature even under prior blinding of both the input text and the secret key. This means that one should use a constant time implementation of MMM even if the secret key is blinded or replaced every time, and *all* side channel leakage should be minimised, whatever multiplier is used.

Keywords: Side channel leakage, simple power analysis, SPA, elliptic curve cryptography, ECC, unified code, Montgomery modular multiplication.

"To ensure that the data carrier consumes the same amount of current whether the requested operation is authorized or unauthorized, a bit is stored in the memory in either event." [Abstract, US Patent 4211919, "Portable data carrier including a microprocessor", filed 28 Aug 1978.]

"No one sews a patch of unshrunk cloth on an old garment, for the patch will pull away from the garment, making the tear worse." [Matt 9:16, The Bible (New International Version)]

1 Introduction

Side channel leakage of secret key information from cryptographic devices has been known publicly for a number of years [1]. In elliptic curve cryptography

(ECC) [9, 13], techniques to reduce this leakage include the use of unified code for point additions and point doublings [3, 7, 8, 12]. When properly implemented, this should eliminate the ability to distinguish the two operations by timing measurements on an appropriate side channel such as power variation [10, 11].

However, for a point doubling some of the arguments are inevitably repeated unless extra precautions are taken. This means that doublings might be separated from additions simply by observing when identical operands, identical arithmetic computations, or identical results are likely to have occurred[1]. This might be done using timing, current or EMR variation [4–6, 10, 11, 15]. The generality of such an attack shows that it may be unwise to attempt to patch up old (i.e. leaky) hardware using new, leak-resistant software counter-measures.

Here, to make the details concrete and quantifiable, this is considered for a specific choice of the algorithms and form of side channel leakage, namely: i) point multiplication using the standard square-and-multiply exponentiation algorithm, together with ii) the version of unified code presented by Brier and Joye [3] and their suggested implementation, iii) field multiplication using a version of Montgomery modular multiplication (MMM) [14] which includes a conditional subtraction and iv) a cryptographic device in which every such subtraction can be observed on some side channel or combination of side channels. It is shown that this association leaks sufficiently for it to be computationally possible to deduce some keys for standard elliptic curves over the smaller fields *even* if they are used just once. The choice of Montgomery, while perhaps not natural for the EC-DSA prime fields, has the benefit of being very well studied, so that the required probabilities are known. Almost *any* other leaky modular multiplier could be substituted.

One of the standard, recommended, EC-DSA elliptic curves is chosen for illustration, namely P-192 [2]. Some theoretical analysis shows that there are some input texts which are weaker than average in terms of the number of additions which can be positively identified as not being doublings. Increasing the sample size yields still weaker cases. Experimental results confirm the theory very strongly, showing indeed that there are even weaker instances where the secret key can be recovered without significant computational resources.

The discussion overturns several potential misconceptions. First, unified code on its own is not a panacea against simple power analysis. It protects *only one* aspect of the implementation. Secondly, the three standard blinding techniques helpfully listed by Coron [4] can provide *no* protection at all gainst some attacks. And thirdly, modular multiplication can leak sufficient data for a successful attack even when a key is used *just once*, such as in the Digital Signature Algorithm (DSA) [2]. Previously, MMM was only known to leak so severely if the same, unblinded key was used repeatedly.

---

[1] For example, for both the redundant-operation-enhanced formulae of [7] and the Hessian form in [8], doubling reveals itself if field squares can be detected. Further, the Jacobi form in [12] suffers from essentially the same potential weakness as is explored here for the Weierstraß form in [3], namely repeated evaluation of identical field products.

In conclusion, even with modern leak-resistant algorithms, field multiplication should clearly be implemented in a time independent manner and, indeed, although the case is not detailed here, in a manner which has a low probability of emitting enough side channel leakage to distinguish successfully between identical and non-identical pairs of multiplications.

2 The Unified Point Addition/Doubling Formula of Brier & Joye

Brier and Joye [3] provide formulae which unify the classical formulae for point addition and point doubling on the Weierstraß form of an elliptic curve. They describe a number of cases, including the use of either affine or projective coordinates.

In their point addition formula for affine coordinates, there is no longer a denominator which becomes zero for point doubling. So a separate formula is not needed. Although there *is* still a denominator which may be zero, this special case is no longer associated with point doubling, but with the sum being the exceptional point $\mathcal{O}$ at infinity. So the formula breaks the previously strong connection between bit values of the secret key and sub-sequences of arithmetic operations.

Assume projective coordinates are used for the representation of points $P = (x, y, z)$ on the Weierstraß form of an elliptic curve

$$y^2 z \;=\; x^3 + axz^2 + bz^3$$

where the field characteristic is not 2 or 3. Then, for points $P_i = (x_i, y_i, z_i)$, $1 \le i \le 3$, the point addition $P_3 = P_1{+}P_2$ is achieved by Brier and Joye using:

$$\begin{aligned} x_3 &= 2fw \\ y_3 &= r(g-2w) - l^2 \\ z_3 &= 2f^3 \end{aligned} \tag{1}$$

where

$$\begin{array}{l} u_1 = x_1 z_2, \quad u_2 = x_2 z_1, \quad t = u_1 + u_2; \\ s_1 = y_1 z_2, \quad s_2 = y_2 z_1, \quad m = s_1 + s_2; \\ z \;= z_1 z_2, \quad f = zm, \quad l = mf, \quad g = tl; \\ r \;=\; t^2 - u_1 u_2 + az^2, \quad w \;=\; r^2 - g. \end{array} \tag{2}$$

This involves eighteen field multiplications, of which one is by the constant a, five are in the equations (1) and thirteen in the equations (2). Without loss of generality, it is assumed that the implementation under attack computes the coordinates of P_3 by taking each of these last equations (2) and calculating the right sides in the order presented before calculating those for (1).

3 The Point Multiplication Algorithm

We assume point multiplication is done using the standard *square-and-multiply* (or *double-and-add*) exponentiation method of Fig. 1. There are two main properties of this algorithm which are used here. First, each bit 1 in the key generates

Input: $\mathbf{P}$, $k = (k_{N-1}....k_1k_0)_2$ with $k_{N-1} = 1$
Output: $\mathbf{Q} = k\mathbf{P}$

```
Q ← P ;
For i ← N−2 downto 0 do
Begin
   Q ← 2Q ;
   If k_i ≠ 0 then Q ← Q + P ;
End
```

Fig. 1. Left-to-Right Binary "Double-and-Add" Algorithm.

a point doubling followed by a point addition, but a bit 0 in the key generates a point doubling which is followed by another point doubling. Hence, it is easy to translate from a sequence of adds and doubles obtained through a side channel into a sequence of bits which reveals the secret key. Secondly, the initial input $\mathbf{P}$ is an argument in every point addition. Although this may save some data manipulation, it occasionally produces the bias in each point addition which is exploited here. Choosing instead the right-to-left version of binary exponentiation or m-ary exponentiation ($m > 2$) would probably render the attack here computationally infeasible.

4 Notation and Montgomery Multiplication

For simplicity, we assume the main side channel leakage is from an implementation of field multiplication using Montgomery Modular Multiplication (MMM) [14] in which there is an observable, final, conditional subtraction. However, it must be emphasised that this choice is only for convenience in evaluating the probabilities. A similar attack could be mounted against any modular multiplier exhibiting data-dependent side-channel leakage.

Suppose the elliptic curve is defined over the Galois field $GF(P) = \mathbb{F}_P \cong \mathbb{Z}/P\mathbb{Z}$ and elements of this prime field are represented as long integers modulo P written to base r with digits in lowercase. Thus, for example, $A = \sum_{i=0}^{n-1} a_i r^i \in \mathbb{F}_P$. Suppose R ($\geq P$) is the upper bound we wish to have on the inputs and outputs for MMM. Then we assume the version of MMM given in Fig. 2.

Input: A and B such that $A, B < R \leq r^n$ and P prime to r.
Output: C such that $C \equiv ABr^{-n} \bmod P$ and $C < R$

```
C ← 0 ;
For i ← 0 to n-1 do
Begin
   q_i ← -(c_0+a_i b_0)p_0^-1 mod r ;
   C  ←  (C+a_i B+q_i P)  div r ;
End ;
{ Invariant: Cr^n ≡ A×B mod P and ABr^-n ≤ C < P+ABr^-n }
If C ≥ R then C ← C-P ;
```

Fig. 2. Montgomery's Modular Multiplication Algorithm (MMM).

The invariant after the loop is easy to establish, and, if desired, the digits q_i could be formed into a quotient Q giving the multiple of P which has been subtracted. The post-condition $C < R$ then holds providing n and R satisfy the right properties, namely that the maximum possible value for C is less than $R{+}P$, i.e. $P{+}R^2r^{-n} \leq R{+}P$, which is just the stated pre-condition $R \leq r^n$. This post-condition enables output from one execution of MMM to be fed into another instance of it.

The usual values for R to take are r^n or P. The former gives a cheap test in the conditional statement, whereas the latter yields the least non-negative residue as output. Decreasing R or increasing n reduces the frequency of the final subtraction and so reduces the leakage from the timing side channel. In fact, the subtraction ceases to occur if $P{+}R^2r^{-n} \leq R$, i.e. if $P \leq R(1{-}Rr^{-n})$ [18]. As this requires $P < R < r^n$, we would take R as a power of 2 to make the condition easy to evaluate – say $R = \frac{1}{2}r^n$ – and demand that n be large enough – say $P < \frac{1}{4}r^n$. Indeed, this choice for R permits the maximum possible value of P to give no final subtraction for a given n. Hence, for a given maximum value of P, it is possible to deduce values for n and R which will always prevent the final subtraction occurring.

Where side channel attacks are possible, the final, conditional subtraction should be protected from timing variation by performing the subtraction in all cases if it can occur at all and then selecting the new or old value of C as appropriate. Eliminating the need for the final subtraction may lead to less side channel leakage, but an extra loop iteration may be the result of choosing n sufficiently large for this to happen.

Because, unlike RSA, the ECC-related equations (2) involve field additions or subtractions between some field multiplications, here it is most convenient to take $R = P$. Then, as noted, the final subtraction will occur occasionally. Observe from the post-loop invariant that its occurrence is independent of the order of inputs. So changing the input order is not a counter-measure to any attack on the final subtraction.

5 The Probability of a Conditional Subtraction

To estimate the probability of the extra subtraction of P in MMM, observe that P is large and so one can switch from discrete to continuous methods. For all practical purposes, inputs to MMM are uniformly distributed modulo P. Thus, if the random variable X represents a typical input to MMM, its probability density function f is given by $f(x) = P^{-1}$ on the interval $[0, P]$, and $f(x) = 0$ otherwise. Furthermore, outputs are also effectively uniformly distributed modulo P. So the probability of the extra subtraction for random inputs can be deduced from the invariant formula after the loop in MMM [18]. Derivations of this probability are given in [16, 17, 19] for various contexts[2]. In

[2] Taking R equal to a power of 2 leads to some interesting non-uniform distributions which are illustrated graphically in [17].

the case of squaring X, the probability is $p_S = prob(X^2r^{-n}+Z > P)$ where Z is uniform on $[0, P]$. Hence

$$p_S = \int_0^P P^{-1}f(x)x^2r^{-n}\,dx = \tfrac{1}{3}\,Pr^{-n} \tag{3}$$

In the case of multiplying two independent, random residues X and Y, the probability of the conditional subtraction is $p_M = prob(XYr^{-n}+Z > P)$ for equi-distributed Z, namely

$$p_M = \int_0^P \int_0^P P^{-1}f(x)f(y)xyr^{-n}\,dy\,dx = \tfrac{1}{4}\,Pr^{-n} \tag{4}$$

However, for the standard exponentiation algorithm of Fig. 1, the point doublings require field multiplications by a fixed constant, namely a coordinate of the initial (plaintext) point $\mathbf{P}$. If p_C is the probability of the final subtraction when multiplying a random input X by a constant C, then $p_C = prob(CXr^{-n}+Z>P)$ so that

$$p_C = \int_0^P P^{-1}f(x)Cxr^{-n}\,dx = \tfrac{1}{2}\,Cr^{-n} \tag{5}$$

The different coefficients of r^{-n} in these three equations enable the different operations to be distinguished by their subtraction frequencies when a number of exponentiations occur with the same secret key k [19, 16]. Here cases will be selected where C has the most extreme values in order to force final subtraction behaviour that proves two products have different inputs.

6 Side Channel Leakage

In addition to the assumptions about the algorithms used, in this instance of the attack, it is supposed that *every* occurrence of the conditional subtraction in MMM can be observed. With good monitoring equipment and a lack of other hardware counter-measures, successive data loading cycles can be observed and hence the operations timed, thereby enabling the occurrences to be deduced.

Previous authors have described in detail how timing and power attacks can be performed [10, 11, 5, 15, 6]. In fact, all we require here is a reasonable probability of determining whether or not two given instances of modular multiplication have the same pair of inputs. This might be done equally well by power or EMR analysis rather than by timing – the power consumption curves may be sufficiently characteristic of the inputs for this to be possible [20]. Furthermore, this property of distinguishability may occur for any modular multiplier, implying that the attack described here is not restricted only to Montgomery multipliers.

7 Distinguishing Doublings from Additions

To provide outline theoretical justification for the later experimental results, in this section a particular single signing or decryption is selected from a randomly generated set of samples. This selection has properties which increase the number of point operations that can be definitely determined as additions or doublings from knowledge of the conditional subtractions. Consequently, for this choice there is a reduction in the computational effort to traverse the space of possible keys and recover the key used.

Brier and Joye [3] provide an algorithm for computing $P_3 = P_1+P_2$ by imposing a suitable ordering on equations (2). In the case of the point doubling operation with $P_1 = P_2 = (x, y, z)$, the arithmetic of (2) specialises to:

$$\begin{array}{llll} u \leftarrow x * z; & u \leftarrow x * z; & t \leftarrow u + u; & \\ s \leftarrow y * z; & s \leftarrow y * z; & m \leftarrow s + s; & \\ z \leftarrow z * z; & f \leftarrow z * m; & l \leftarrow m * f; & g \leftarrow t * l; \\ \multicolumn{2}{l}{r \leftarrow t^2 - u^2 + a * z^2;} & w \leftarrow r^2 - g; & \end{array} \tag{6}$$

Here the first two applications of MMM are identical, as are the second two. So both pairs exhibit identical behaviour with respect to the occurrence of the final conditional subtraction. It is the repeated or different behaviour within the pairs which creates the handle for an attack. If the recorded behaviour is different at these points, the curve operation *must* be a point addition.

From a sample of signatures or decryptions, the attacker starts by selecting the case which has the smallest number of undetermined operations. The initial, naïve way to do this is just to count the total number of operations, and subtract the number of those for which the computations of either u_1 and s_1, or u_2 and s_2, or both, involve differences regarding the final subtraction.

Point additions involve the initial input $\mathbf{P} = \mathbf{P}_1 = (x_1, y_1, z_1)$ where the natural variation in random (or randomized) coordinates means that occasionally x_1 and y_1 will both be small (i.e. close to 0) and z_1 will be large (i.e. close to P). By equation (5), this means the computations of u_1 and s_1 in (2) will be *less* likely than average to include the additional subtraction, while the computations of u_2 and s_2 will be *more* likely than average to include the additional subtraction. For such initial $\mathbf{P}$, this enhances the likelihood of different behaviour with respect to subtractions in these pairs of multiplications, thus increasing the number of point operations which can be definitely determined to be additions rather than doublings.

Such a point is used below to develop some theory[3]. But, in fact, with the selection criterion above and others developed later, the attacker is actually likely to choose a case which is *much* more amenable to an attack than that where $\mathbf{P}$ has such extreme coordinates.

[3] A case with small x_1, small y_1 and large z_1 is almost always obtained by choosing the side channel trace which maximises the average number σ of differences per operation:

$$\sigma \ = \ \frac{\sum_{\mathrm{op}}\{\delta\mathrm{op}(u_1,u_2)+\delta\mathrm{op}(s_1,s_2)\}}{\#\mathrm{operations}}$$

where $\delta_{op}(x_1, x_2)$ is 1 or 0 depending on whether or not the products x_1 and x_2 differ with respect to the occurrence of a final subtraction in operation *op* of a complete point multiplication.

Suppose the sample size is 512. This is a reasonably practical number of samples to obtain. In one eighth of cases, x_1 will lie in the interval $[0, \frac{1}{8}P]$ and have an average value of $\frac{1}{16}P$. So, from $512 = 8^3$ cases we can expect one instance where the initial input is

$$\mathbf{P}_0 \;\approx\; (\tfrac{1}{16}P, \tfrac{1}{16}P, \tfrac{15}{16}P) \tag{7}$$

$\mathbf{P}_0$ is always an input, say $\mathbf{P}_1$, to each point addition. It is constant over the set of all point additions of the associated signing or decryption. The other input, $\mathbf{P}_2$, for these point additions is, for all practical purposes, a point with random coordinates. Hence (5) is the formula to apply.

For each of the example EC-DSA curves [2] over a field of odd characteristic P, the field order is a generalized Mersenne prime. For example, P-192 uses $P = 2^{192}-2^{64}-1$. Consequently, P is very close to a large power of 2 and, in MMM, r^n will certainly be taken equal to that power of 2 (except perhaps for P-521). So $r^n = 2^{192}$ for P-192. Therefore, the ratio Pr^{-n} will be essentially 1 for most EC-DSA curves used in practice.

Using $\mathbf{P}_0$ and this value for Pr^{-n} in (5), $u_1 \leftarrow x_1{\times}z_2$ incurs a final subtraction with probability approximately $p_{small} = \frac{1}{32}$, and the same holds for the computation of s_1. Similarly, $u_2 \leftarrow x_2{\times}z_1$ incurs a final subtraction with probability approximately $p_{large} = \frac{15}{32}$, and the same holds for the computation of s_2. As the different inputs are independent, the probability p_{diff} of u_1 incurring a subtraction but not u_2, or *vice versa*, is

$$p_{diff} \;=\; p_{small}{\times}\overline{p_{large}} \;+\; \overline{p_{small}}{\times}p_{large} \;=\; \tfrac{241}{512} \;\approx\; \tfrac{1}{2}$$

There is the same probability of distinguishing between s_1 and s_2 in this way.

As the outputs from the loop of MMM are uniformly distributed over an interval of length P, the subtractions for the four products are all independent of each other even though some share a common input. So the probability p_{add} of proving that a point addition occurs as a result of observing differences in subtractions for at least one of the pairs is[4]

$$p_{add} \;=\; \overline{(\overline{p_{diff}})^2} \;=\; \tfrac{188703}{262144} \;\approx\; \tfrac{3}{4} \tag{8}$$

In a similar way, one can obtain the probabilities p_{A00}, p_{A02}, p_{A20} and p_{A22} of a point addition displaying, respectively, no subtractions, no subtractions for two multiplications then two subtractions, two subtractions then no subtractions, and four subtractions. These are the cases where the subtractions leave it ambiguous as to whether the operation is a point addition or a point doubling. For the fixed input $\mathbf{P}_0$ given in (7) and a random input $\mathbf{P}_2$,

$$\begin{aligned}
p_{A00} &= \overline{p_{small}}{\times}\overline{p_{large}} \times \overline{p_{small}}{\times}\overline{p_{large}} &= \tfrac{277729}{1048576} \\
p_{A02} &= \overline{p_{small}}{\times}\overline{p_{large}} \times p_{small}{\times}p_{large} &= \tfrac{7905}{1048576} \\
p_{A20} &= p_{small}{\times}p_{large} \times \overline{p_{small}}{\times}\overline{p_{large}} &= \tfrac{7905}{1048576} \\
p_{A22} &= p_{small}{\times}p_{large} \times p_{small}{\times}p_{large} &= \tfrac{225}{1048576}
\end{aligned} \tag{9}$$

[4] The accuracy here and later is absurd, but it should enable readers to understand and check the calculations more easily.

In the case of a (random) doubling, the field multiplications of interest have two independent random inputs and so, by (4), the corresponding probabilities are:

$$\begin{aligned} p_{D00} &= \overline{p_M} \times \overline{p_M} = \tfrac{9}{16} \\ p_{D02} &= \overline{p_M} \times p_M = \tfrac{3}{16} \\ p_{D20} &= p_M \times \overline{p_M} = \tfrac{3}{16} \\ p_{D22} &= p_M \times p_M = \tfrac{1}{16} \end{aligned} \tag{10}$$

These last probabilities sum to 1 because it is not possible for the multiplications within a pair to behave differently.

By (8), about three quarters of the additions are determined immediately. On average, this leaves about $26\frac{3}{4}$ additions unrecognised for P-192 (30 if the sample size is reduced to only 64). In fact, simulations below in Table 2 show that the attacker, with a better selection criterion, has a mere $19\frac{1}{4}$ additions unrecognised for the same sample size of 512.

For undetermined cases, the number of final subtractions can still be used to make a probabilistic decision between an add or a double. Suppose an operation is known to be a point addition with probability π_A and a point doubling with probability $\pi_D = \overline{\pi_A}$, and the subtractions do not distinguish the operation as an addition. Using the same notation as above for counting subtractions, the probabilities of a point addition in the various cases can be deduced from (9) and (10) as:

$$\begin{aligned} p_{add|00} &= \frac{\pi_A \frac{277729}{293764}}{\pi_A \frac{277729}{293764} + \pi_D \frac{9}{16}} \approx \frac{\pi_A}{\pi_A + \pi_D \frac{9}{16}} \\ p_{add|02} &= p_{add|20} = \frac{\pi_A \frac{7905}{293764}}{\pi_A \frac{7905}{293764} + \pi_D \frac{3}{16}} \approx 0 \\ p_{add|22} &= \frac{\pi_A \frac{225}{293764}}{\pi_A \frac{225}{293764} + \pi_D \frac{1}{16}} \approx 0 \end{aligned} \tag{11}$$

Consequently, no subtractions are most likely in such a situation, and the bias to one or other depends on the context, such as neighbouring operations, which might influence the value of π_A. In the unlikely event of two or four subtractions, the attacker would be unfortunate if more than one or two such operations were not doublings: on average he expects only $26\frac{3}{4} \times \frac{7905+7905+225}{277729+7905+7905+225} \approx \frac{3}{2}$ such operations for P-192.

8 Reconstructing the Secret Key

This section covers both the deduction of unclear key bits from the overall structure of the point operations, and a search of the subspace of keys which are consistent with that structure and with previously determined operations.

Again, for this section, numerical examples apply to the standard P-192 curve defined in FIPS 186-2 [2] for EC-DSA. This has the advantage of a short key length. The methods apply in exactly the same way to other curves, but it will become evident that larger fields require more computational effort.

From Fig. 1, the structure of point operations during signing or decryption can be viewed as a string S over the alphabet $\{A, D\}$ in which the first character of S is 'D' (a double), each 'A' (an add) is preceded by a 'D', and there are a known, fixed number of occurrences of 'D', namely $N-1$ where N is the number of bits in the key k. On average, there will be $\frac{1}{2}(N-1)$ 'A's, but for each case this can be established exactly by taking the total number of operations (the length of S) and subtracting the number of doublings.

By (8), a substantial number of the 'A's are known. Each known 'A' determines its neighbours as 'D' on either side (*see* Table 1). Some occurrences of 'D' are determined by two 'A's. The probability of this for an interior 'D' is $(\frac{1}{2}p_{add})^2$. Neglecting differences caused by end conditions (such as the last 'A' is, or is not, followed by a 'D' and the initial 'D' is known), the total number of determined operations is, on average, about

$$\tfrac{3}{2}(N-1)p_{add} - (N-2)(\tfrac{1}{2}p_{add})^2 \qquad (12)$$

For the on-going P-192 example, about $68\frac{3}{4}$ additions are determined from the subtractions and so, by (12), around 181.6 operations in total. This leaves about $26\frac{3}{4}$ 'A's to allocate among approximately $286.5-181.6 = 104.9$ unknown positions – approximately $\binom{104.9}{26.75} \approx 2^{82}$ choices. However, these choices cannot be made freely.

Two thirds of the string S is accurately determined as above. This leaves a number of short substrings to be guessed. These are restricted to patterns which do not have consecutive 'A's. For our parameter choices, the number of these substrings decreases exponentially with their length. So, most frequently, the unknown substrings are of length 1. They can be either 'A' or 'D'. Each possibility can occur. However, substrings of length 2 are constrained to 'AD', 'DA' or 'DD', and those of length 3 to only 'ADA', 'ADD', 'DAD', 'DDA' or 'DDD'. So only $\frac{3}{4}$ of choices are possible for length 2 substrings, only $\frac{5}{8}$ for length 3, and one half or less for longer substrings. (The numerators go up in a Fibonacci sequence: 2, 3, 5, 8, 13,...; and the denominators in powers of two.)

In the P-192 example, about 7 substrings of length 2 occur, 4 of length 3, and 7 of longer lengths. So fewer than $(\frac{3}{4})^7(\frac{5}{8})^4(\frac{1}{2})^7$ of the 2^{82} choices, and, more precisely, only roughly one in $2^{15.6}$, satisfies the constraint of preceding each 'A' by a 'D' (*see* Table 2). The search space is therefore reduced to under 2^{67}.

In allocating the 'A's, we can also note that 'D' is much more likely in some cases, and 'A' perhaps in others. For example, by (10), $\frac{7}{16}$ of doubles will exhibit 2 or 4 subtractions, but, by (11), very few operations with this behaviour could be additions. In the example, about $\frac{7}{16}(191-(2\times 68\frac{3}{4}-24.6)) \approx 34$ doubles can be so identified with high probability and on average it is only necessary to try up to 2 of them as doubles. Thus, in estimating the computational effort, the $\binom{104}{26.75}$ can be replaced by $\binom{34}{2}\binom{70}{24.75} \approx 2^{71}$. With the substring constraints, this now reduces the search space to below 2^{56}.

Suppose each key in the search space is checked by applying it once in a point multiplication – some 286.5 point operations, each containing 18 field multiplications. With 32-bit arithmetic, MMM requires $6\times(1+6\times 2) = 78$ native machine multiplications. Thus, about 2^{19} MULs are required per key, giving a time complexity of $O(2^{75})$ machine multiplications to break one member of the sample of

512 P-192 signings/decryptions. This is probably not yet computationally feasibility at $O(2^{18})$ Pentium® IV years, although it could be distributed easily over a number of machines. However, ...

9 Worked Examples

This section provides more detail for a typical P-192 attack and assesses the impact of changing various parameters. In particular, the attacker invariably chooses very much better examples than those of the previous section, showing that the attack is, in fact, already feasible. Sequences of conditional subtractions were simulated using equations (3)–(5) and continuous mathematics on randomly generated keys k and inputs P. Different sized samples were generated and one member selected to:

> *minimize the number of point operations which were not distinguishable as additions by virtue of the conditional subtractions, nor distinguishable as doublings by adjacency to a known addition.*

Table 1 shows the initial few bits, point operations, conditional subtractions and deductions for a typical example selected using this criterion. There is one column for each point add (`A`) or double (`D`). The penultimate row records differences (marked `Y`) within the first or the second pair of subtractions. These are all adds and the final row extends this to include the doubles on either side of a known add. The attacker computes this for each signature, and chooses the one with the fewest unknowns (marked `*`) in the last line.

Key	`1 1 1 1 1 1 1 100 100 100 100 1 1 100 1 1 1 1 10 1 10 1 1 1 1 1 1`
Pt Op$^{\mathrm{n}}$	`DADADADADADADADDDADDDADDDADDDADADADDDADADADADADDADADDADADADADADA`
u_1 sub$^{\mathrm{n}}$	`010110000001010100001100110001010101000101000000100010100000110 0`
u_2 sub$^{\mathrm{n}}$	`0100100101000001010011001000000100010000000000000000111000011101`
s_1 sub$^{\mathrm{n}}$	`0000101111000010001110010100001111000000100011000100001000100000`
s_2 sub$^{\mathrm{n}}$	`0100101011010110001110010100011010000101100111001110011000110001`
Diff$^{\mathrm{nce}}$	` Y Y   Y Y Y Y   Y       Y   Y Y Y   Y Y Y Y   Y Y  Y     Y   Y`
Known	`YYYYY*YYYYYYYYY*YYY*****YYY*YYYYYYY*YYYYYYYYY**YYYYYYYY***YYY*YY`

Table 1. Example Deductions of Operation Types

Table 2 shows how the average number of undetermined operations in the selected sequence varies according to the sample size from which it is taken. The most significant benefit from larger samples is the decrease in the total number of indeterminate operations. The undetermined point additions, α in number, must be chosen from the undetermined operations, τ in number. The number of such combinations is given lower in the table. The longer substrings of unknown operations have more pattern restrictions. The average factor by which this cuts the search space is given in the penultimate line, and the last line presents the resultant overall order of the search space. The computational effort is this multiplied by the work involved in verifying a single key.

P-192 Sample Size	32	64	128	256	512	1024
Total Ops	292.5	293.7	294.8	295.9	296.9	297.9
Unknown Ops (τ)	54.4	49.2	44.4	40.0	36.2	32.7
Unknown Adds (α)	22.4	21.4	20.4	19.9	19.2	18.7
Unknown 1-strings	11.2	11.5	11.7	11.7	11.9	12.2
Unknown 2-strings	7.1	7.1	7.1	7.2	7.2	7.0
Unknown 3-strings	4.6	4.5	4.5	4.5	4.4	4.1
Unknown 4-strings	3.0	2.9	2.8	2.7	2.6	2.6
Unknown 5-strings	1.9	1.8	1.8	1.7	1.6	1.4
Longer unknown strings	3.5	3.1	2.8	2.6	2.4	2.4
Combinations $\binom{\tau}{\alpha}$	2^{50}	$2^{45.5}$	$2^{41.1}$	$2^{37.0}$	$2^{33.2}$	$2^{29.4}$
Substring restrictions	2^{19}	2^{18}	2^{17}	$2^{16.2}$	$2^{15.6}$	$2^{15.0}$
Search Space Order	2^{31}	$2^{27.5}$	$2^{24.1}$	$2^{20.8}$	$2^{17.6}$	$2^{14.4}$

Table 2. Average Numbers of Operations in the selected Point Multiplication.

The figures make it clear that the criterion just given for selecting the side channel trace is hugely more powerful than if an extremal initial point **P** were used. For the sample size of 512, there is a reduction in workload by a factor of 2^{38}. This means that the computational effort is reduced to $O(2^{36.6})$ native multiplications, i.e. about a minute on a Pentium IV processor running at 2^{32} cycles per second. This is certainly feasible for any back street attacker with access to suitable monitoring equipment, not just for government organisations. Indeed, less than a week of computing reveals 1 in 32 keys.

The table also shows that twice as many keys can be extracted from a given sample for about 10 times the effort – the easiest keys are found first, after which they become steadily more difficult to find.

Key Length	192	224	256	384	521
Total Ops	296.9	345.7	394.2	588.4	795.7
Unknown Ops (τ)	36.2	45.9	55.7	96.3	141.6
Unknown Adds (α)	19.2	23.0	26.6	41.5	57.9
Combinations $\binom{\tau}{\alpha}$	$2^{33.2}$	$2^{42.8}$	$2^{52.4}$	$2^{91.4}$	$2^{134.3}$
Substring restrictions	$2^{15.6}$	$2^{18.8}$	$2^{22.0}$	$2^{35.4}$	$2^{50.1}$
Search Space Order	$2^{17.6}$	$2^{24.0}$	$2^{30.4}$	$2^{56.0}$	$2^{84.2}$

Table 3. Search Space Sizes for Different Key Lengths (Sample of 512).

Finally, in Table 3 a comparison is made of the search spaces for the other recommended EC-DSA curve sizes[5]. It would appear that it is still computationally feasible to attack some keys over fields as large as that of P-256.

[5] Here Pr^{-n} is assumed to be essentially 1. However, 521-bit numbers cannot be partitioned into words of equal length without loss. So, for P-521, Pr^{-n} may not be close to 1. This would result in many fewer final subtractions and so more unknown operations than tabulated.

10 Conclusion

The clear conclusion is that, on its own, unified code for point operations provides insufficient security in some standard implementations of ECC which employ arithmetic hardware that is subject to side channel leakage. In particular, we demonstrated the feasibility of attacking hardware that uses a time-varying implementation of Montgomery modular multiplication with the Brier-Joye formulae for point addition [3].

Several easy counter-measures would defeat the attack. For example, it should be possible to re-code the point evaluation to avoid repeated field operations when a doubling occurs. The formulae of [7] and [8] avoid the problem, but have field squares precisely when a doubling occurs – leaky hardware might still reveal this. Alternatively, picking any other exponentiation algorithm than the standard binary one may reduce or entirely eliminate the bias given in some decryptions as a result of re-using an extremal input in every point addition. Thus, using m-ary exponentiation with $m > 2$ would reduce the frequency of weak cases as well as introducing ambiguities about which point addition is being performed. Certainly, using a more leak-resistant multiplier would improve matters.

However, the three standard counter-measures listed by Coron [4] are insufficient here; they may make no difference or even make the attack more feasible. Key blinding only helps if more than one decryption is required for key recovery. Only one decryption was used here, but if side channel leakage is weak then several decryptions with the same key could help to distinguish additions from doublings with enough certainty for it to be computationally feasible to search the key space. So this counter-measure might ameliorate the situation, although not solve it. Message blinding only helps against chosen ciphertext attacks, but that was not required here. Indeed, the third counter-measure of randomizing the input point coordinates may help the attack to succeed by guaranteeing a uniform distribution of coordinate values which will contain the necessary examples of attackable extremal points.

Previously it was uncertain that time variation was a serious threat except when unblinded keys were used at least several hundred times in an embedded cryptographic device [19]. Now it is clear that constant time modular multiplication is essential for security even when secret keys are always blinded.

References

1. *Portable data carrier including a microprocessor*, Patent no. 4211919 (Abstract), US Patent and Trademark Office, July 8, 1980.
2. *Digital Signature Standard (DSS)*, FIPS PUB 186-2 (Appendix 6), U.S. National Institute of Standards and Technology, 27 Jan 2000.
3. E. Brier & M. Joye, *Weierstraß Elliptic Curves and Side-Channel Attacks*, Public Key Cryptography (Proc. PKC 2002), D. Naccache & P. Paillier (editors), LNCS **2274**, Springer-Verlag, 2002, pp. 335–345.
4. J.-S. Coron, *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*, Cryptographic Hardware and Embedded Systems (Proc. CHES 99), C. Paar & Ç. Koç (editors), LNCS **1717**, Springer-Verlag, 1999, pp. 292–302.

5. J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater & J.-L. Willems, *A practical implementation of the Timing Attack*, Proc. CARDIS 1998, J.-J. Quisquater & B. Schneier (editors), LNCS **1820**, Springer-Verlag, 2000, pp. 175–190.
6. K. Gandolfi, C. Mourtel & F. Olivier, *Electromagnetic Analysis: Concrete Results*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 251–261.
7. C. Gebotys & R. Gebotys, *Secure Elliptic Curve Implementations: An Analysis of Resistance to Power-Attacks in a DSP Processor*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2003, pp. 114–128.
8. M. Joye & J.-J. Quisquater, *Hessian Elliptic Curves and Side Channel Attacks*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 402–410.
9. N. Koblitz, *Elliptic Curve Cryptosystems*, Mathematics of Computation **48**, 1987, pp. 203–209.
10. P. Kocher, *Timing attack on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology – Crypto '96, N. Koblitz (editor), LNCS **1109**, Springer-Verlag, 1996, pp. 104–113.
11. P. Kocher, J. Jaffe & B. Jun, *Differential Power Analysis*, Advances in Cryptology – Crypto '99, M. Wiener (editor), LNCS **1666**, Springer-Verlag, 1999, pp. 388–397.
12. P.-Y. Liardet & N. P. Smart, *Preventing SPA/DPA in ECC Systems using the Jacobi Form*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 391–401.
13. V. Miller, *Use of Elliptic Curves in Cryptography*, Advances in Cryptology – Crypto '85, H. C. Williams (editor), LNCS **218**, Springer-Verlag, 1986, pp. 417–426.
14. P. L. Montgomery, *Modular Multiplication without Trial Division*, Mathematics of Computation **44**, no. 170, 1985, pp. 519–521.
15. J.-J. Quisquater & D. Samyde, *ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards*, Smart Card Programming and Security (Proc. e-Smart 2001), I. Attali & T. Jensen (editors), LNCS **2140**, Springer-Verlag, 2001, pp. 200–210.
16. W. Schindler, *A Combined Timing and Power Attack*, Public Key Cryptography (Proc. PKC 2002), P. Paillier & D. Naccache (editors), LNCS **2274**, Springer-Verlag, 2002, pp. 263–279.
17. W. Schindler & C. D. Walter, *More detail for a Combined Timing and Power Attack against Implementations of RSA*, Cryptography and Coding, K.G. Paterson (editor), LNCS **2898**, Springer-Verlag, 2003, pp. 245–263.
18. C. D. Walter, *Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli*, Topics in Cryptology – CT-RSA 2002, B. Preneel (editor), LNCS **2271**, Springer-Verlag, 2002, pp. 30–39.
19. C. D. Walter & S. Thompson, *Distinguishing Exponent Digits by Observing Modular Subtractions*, Topics in Cryptology – CT-RSA 2001, D. Naccache (editor), LNCS **2020**, Springer-Verlag, 2001, pp. 192–207.
20. C. D. Walter, *Sliding Windows succumbs to Big Mac Attack*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 286–299.