Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks

Elisabeth Oswald and Manfred Aigner

Institute for Applied Information Processing and Communications
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
{Elisabeth.Oswald,Manfred.Aigner}@iaik.at

Abstract. Power Analysis attacks on elliptic curve cryptosystems and various countermeasures against them, have been first discussed by Coron ([6]). All proposed countermeasures are based on the randomization or blinding of the inputparameters of the binary algorithm. We propose a countermeasure that randomizes the binary algorithm itself. Our algorithm needs approximately 9% more additions than the ordinary binary algorithm, but makes power analysis attacks really difficult.

Keywords: Power Analysis, Elliptic Curve Cryptosystems

1 Introduction

Elliptic curve cryptosystems (ECC) have attracted much attention since they were first proposed in 1985 by Miller [24] and Koblitz [15]. The underlying discrete logarithm problem seems to be much harder than in other groups. Today, no subexponential-time algorithm is known for this problem in the case of non-supersingular curves. This results in much shorter keylengths for ECC, which makes those cryptosystems especially attractive for hardware implementations for instance on smartcards. One must consider therefore not only mathematical attacks on ECC, but also attacks that exploit weaknesses in the implementation.

In the last years, attacks have been published that use leaked side-channel-information such as the power consumption or timing measurement. These methods are all passive, this means that an attacker just needs to monitor the cryptographic device. The most popular method today, the *differential power analysis* (DPA) was introduced 1998 by Cryptography Research. DPA exploits the information drawn from the leakage of power consumption. First, mostly applied to symmetric cryptosystems, DPA was then applied successfully on public key cryptosystems, see [6], [12] and [20]. Power analysis is a very strong attack. For a successful attack on a straightforward DES implementation only a few hundred measurements are needed. Also the technical effort is comparatively small. One just needs a digital sampling oscilloscope with an appropriate sampling rate for the power measurements, and a standard PC to process the obtained measurement data. The processing of the data itself is also very easy and it is not necessary to understand the concept of the attack to perform it successfully.

To make a long story short, this attack can be mounted by not only experienced cryptanalysts, but by everyone! This fact makes it even more necessary to counteract this attack for both private and public key cryptosystems.

In this paper we deal exclusively with elliptic curve cryptosystems. The countermeasures that were proposed in the before mentioned articles, rely on randomizing or blinding the parameters (the elliptic curve point P and the secret key k) of the binary algorithm. In our article we present a countermeasure based on randomizing the binary algorithm itself. Our method does not only provide security against power attacks, but also does not slow down the encryption algorithm, or require the storage of additional elliptic curve parameters (for instance the number of points on the curve) as other methods do.

The paper is organized as follows. Considering the binary algorithm, we review the concept of power analysis in section 2. In section 3 we explain the method of addition-subtraction chains as a speedup for the standard binary algorithm. Finally we present our randomization method on the grounds of these addition-subtraction chains.

2 Elliptic Curve Cryptosystems, the Binary Algorithm and Power Attacks

Some public key cryptosystems require the computation of a modular exponentiation ($P = M^k \mod p$) or a scalar multiplication ($P = kM$). This is usually done by the binary algorithm $binalg(P, M, k)$ which is sketched (in its bottom-up version) in the following figure:

```
binalg(P,M,k)
$Q = M$
if $k_0 = 1$ then $P = M$ else $P = 0$
for $i = 1$ to $n - 1$
   $Q = Q * Q$
   if ($k_i == 1$) then
      $P = P * Q$
return $P$
```

For validity and explanation see [14]. The $*$ denotes hereby an appropriate operation, which can be the multiplication for instance, but also the addition.

2.1 ECC Basics

Elliptic curve cryptosystems make use of the binary algorithm for the computation of the scalar point multiplication. An elliptic curve over a field K, short $E(K)$, is defined as a nonsingular homogeneous cubic polynomial $F(x_0, x_1, x_2) \in K[x_0, x_1, x_2]$, provided there is at least one rational point on $E(K)$ [13]. The set of rational points can be made into an abelian group in a natural way. If $P_1, P_2 \in E(K)$, then the line connecting both points intersects the elliptic curve in a third point P_3. Further, one calls the third point of intersection of the line connecting 0 (for example) and P_3 with $E(K)$, the sum of P_1 and P_2. For char $K \neq 2, 3$ every elliptic curve can be written as

$$x_0 x_2^2 = x_1^3 - A x_0^2 x_1 - B x_0^3, \qquad A, B, \in K. \tag{1}$$

This curve has only one point at infinity, which is the identity element of the group. With the transformation $x = x_1/x_0$ and $y = x_2/x_0$, $x_0 \neq 0$, one gets the equation for an elliptic curve in affine coordinates :

$$y^2 = x^3 - Ax - B. \tag{2}$$

The point at infinity is lying infinitely far off in the direction of the y axis. Thus the inverse of a point $P = (x, y) \neq \mathcal{O}$ is $-P = (x, -y)$. Formulas for the point addition and point duplication on an elliptic curve defined over a finite field can be found for example in [5]. The following tables give a brief overview of the different costs of the two operations. I denotes the inversion, M the multiplication and S the squaring in K. Conversion from projective to affine coordinates is not taken into account. Also more efficient projective representations are not included in the tables, see therefore again [5].

Characteristic $K > 3$		
Operation	Coordinates	
	affine	projective
Point addition	1I+3M	16M
Point doubling	1I+4M	10M

Characteristic $K = 2$		
Operation	Coordinates	
	affine	projective
Point addition	1I+2M+1S	15M+5S
Point doubling	1I+2M+1S	5M+5S

Remark 1. In a finite field of characteristic 2, the inverse of an elliptic curve point $P = (x, y)$ is given as $-P = (x, x + y)$. Having that an addition of two elements is calculated by bit by bit Xor, we get the inverse of an elliptic curve point for free again.

2.2 Power Analysis

Power analysis attacks use the fact that the instantaneous power consumption of a hardware device is related to the instantaneous computed instructions and the manipulated data. An unskilled implementation of an elliptic curve point duplication and an elliptic curve point addition, can therefore easily be used to mount a simple power attack (or *simple power analysis*, short SPA). An adversary just needs to monitor the devices power consumption and identify the parts of the power trace that correspond to the additions and duplications. This gives trivially the secret key. It is clear that in order to be SPA resistant, one must try to prevent data depending branches, as sketched in algorithm $binalg'(P, M, k)$. Note, that the computational effort is much higher than in the standard binary algorithm.

```
binalg'(P,M,k)
$P = 1, Q = M$
for $i = 0$ to $n - 1$
    $P[0] = P$
    $P[1] = P * Q$
    $Q = Q * Q$
    $P = P[k_i]$
return $P$
```

Differential power analysis uses more sophisticated, statistical techniques to attack the secret key. One power analysis variant is to partition the measurements in two (or more) different sets by some oracle (for instance the guess of a secret key bit) and then look if these two sets are statistically different. This will only be the case if the oracle was correct and thus reveal some parts of the key. Since statistical difference is usually computed by the $distance-of-mean-test$, which basically compares the means of two distributions, we will refer to this method as the *mean method* in subsequent sections. The second method computes the covariance between the measurements and the oracle. Also, only a correct oracle can correlate to the measurements (we will refer to this as the *correlation method*). We give some examples to clarify this description.

Example 1 (Single-Exponent, Multiple-Data Attack). The SEMD attack [20] compares the power signal of an encryption operation using a known parameter (public key) to a power signal using an unknown parameter (secret key). The attacker can learn where the two signals differ and thus learn the unknown (secret) parameter. Due to noise components, direct comparisons of power signals are unreliable, thus DPA techniques are applied. One computes n random values with the secret and the known parameter. The average signals are calculated and subtracted as in the *mean method* . The portions of the DPA signal that depend on the (random) data will be wiped out by the averaging and subtraction. The portion of the DPA signal that is dependent on the parameter will average out to two different values depending on the performed operation. The portions in the DPA signal that are ≈ 0 are data dependent or the operations in the binary algorithm agree. The other portions indicate that the operations in the binary algorithm differ.

This attack also can be seen as an extension of a SPA attack, and therefore be prevented by the modification sketched in $binalg'(P, M, k)$. Note that this variant does not make much assumptions on the cryptographic device. More sophisticated versions, that make more assumptions can be found in [20].

Example 2 (Correlation Attack). When using algorithm $binalg'$ the *mean method* will be not successful because there is no difference in the sequence of instructions. But if one knows the representation of the computed points one can again mount a successful attack (which has been shown in [6]). At step i, the processed point depends only on the first bits $k_0 \dots k_{i-1}$ of the secret parameter k. When $P[i]$ is processed, power consumption is correlated to the bits of $P[i]$.

No correlation will be observed if the point is not computed. The value of the least significant bit of k can be learned by calculating the correlation between the power consumption and any specific bit of $2M$. As one can see in algorithm binalg' the only key dependent operation in the for-loop is whether the value of $P[1]$ or $P[0]$ is copied to P. If $k_0 = 0$, the value of $P[0]$ (which is 1 in this case) remains in P and therefore a correlation between $2M$ and the power consumption in the subsequent path of the for-loop must be observed, otherwise if $k_0 = 1$, the value of $P[1]$ (which is M in this case) will be copied to P and no correlation between $2M$ and the power consumption of the computation of $P[1]$ in the subsequent path in the for-loop will be observed. The other bits can be recursively recovered in the same way.

Coron also shows in [6] how to extend the *correlation method* to any scalar multiplication algorithm executed in constant time with a constant addition-subtraction chain.

2.3 Countermeasures

Basically all proposed countermeasures suggest blinding or randomizing the secret parameters. When computing $P = kM$ one has the possibility to

- **randomize (blind) k:** One needs to know the number of points $\#E(K)$ on the elliptic curve. Then one chooses a random number r and calculates $k' = k + r * \#E(K)$. Obviously $P = kM = k'M$, because of $\#E(K) * M = \mathcal{O}$. For this approach one has to store an additional parameter of the elliptic curve, on the cryptographic device, which is often not desirable. The second disadvantage is that depending on the bitlength of $r * \#E(K)$, the effective keylength may increase.
- **blind M :** A point is blinded by adding a secret random point R for which one knows $S = kR$. Scalar multiplication is done by calculating $k(R + M)$ and subtracting S to get $P = kM$. The points R and S can be stored inside the cryptographic device and updated for each new execution as follows: $R = (-1)^b 2R$ and $S = (-1)^b 2S$, where b is a random bit. Note that there must be stored two additional points inside the device, which is also often not desirable.
- **randomize M:** Projective coordinates can be used to avoid the inversions as well as for randomization. Because of the fact that

  $$(X, Y, Z) = (\lambda X, \lambda Y, \lambda Z), \quad \forall \lambda \neq 0$$

  one can choose for each new execution another random λ. As one can see, this variant relies on the usage of projective coordinates instead of affine coordinates.

All these countermeasures require to store additional parameters or to make additional operations.

3 Speeding Up the Binary Algorithm

As pointed out in section 2, an elliptic curve cryptosystem needs to efficiently compute the scalar multiplicative of an elliptic curve point. The simplest efficient method, the binary algorithm, is also the oldest. A good survey on (more recent) methods is [10]. Most of these methods try to give an answer to the question, how to find the shortest addition chain. An *addition chain* for an integer k is a list of positive integers $a_1 = 1, a_2, \ldots, a_l = k$, such that for each $i > 1$, there is some j and m, with $1 \leq j \leq m < i$ and $a_i = a_j + a_m$. Thus, if one has an addition chain with length l, one can compute $k * P$ with l additions. Finding the best addition chain is impractical, but there are several methods for finding near-optimal ones. With elliptic curves one has the possibility to use addition-subtraction chains, because the computation of the inverse of a point has no cost. Morain and Olivos discuss in [23] two algorithms that use addition-subtraction chains. We describe their approach in the two subsequent sections.

3.1 First Algorithm

The idea comes from the observation that long chains of 1's in the binary representation of k are better treated by a subtraction. For instance if one calculates $15 * P$ like

$$15 * P = 16 * P - P = 2(2(2(2P))) - P,$$

one has to perform less operations than in the standard binary algorithm. So the enhancement is to replace a block of at least two 1's in the binary representation of k, by a block of 0's and a -1 : $1^a \mapsto 1\,0^{a-1} - 1$. Automaton 1 in Figure 1 represents this idea. Morain and Olivos state that the expected gain of this version is about 8.33%.

3.2 Second Algorithm

The idea is to treat isolated 0's inside a block of 1's. Using the map of the first algorithm it is

$$1^a01^b \mapsto 10^{a-1} - 110^{b-1} - 1.$$

Since $-2 + 1 = -1$ we can write -11 as $0 - 1$ and therefore

$$1^a01^b \mapsto 10^a - 10^{b-1} - 1.$$

In automaton 2, the state 110 takes this modification into account. In both figures the input path is marked by a distinct arrow, and the output paths are marked by an additional bar. Intermediate states are drawn as circles, and transitions between these intermediate states are represented as arrows. The initial conditions for the automatons are $P = 0$ and $Q = M$. An iterative version of this algorithm can be found in [23]. The expected gain for this variant is about 11.11%.

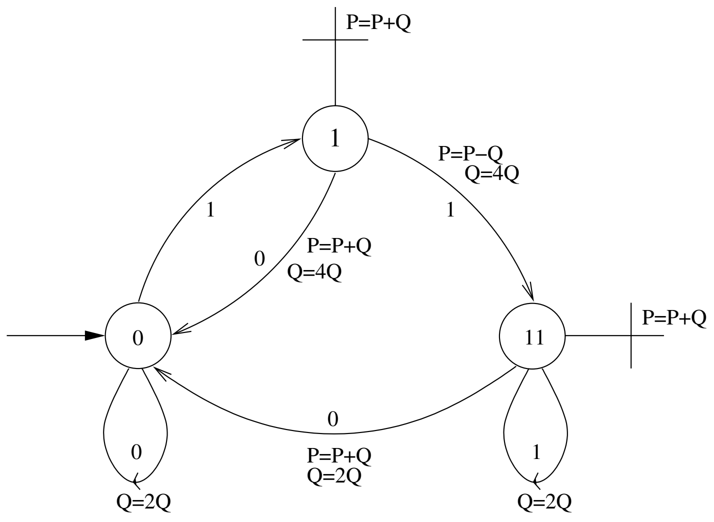

Fig. 1. Automaton 1

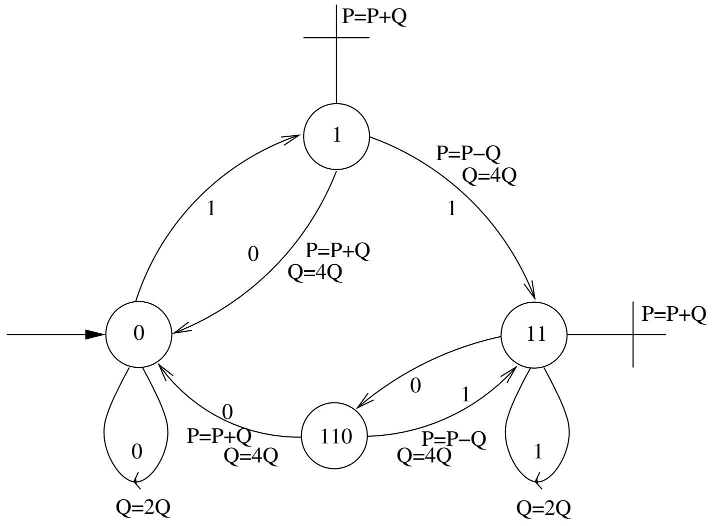

Fig. 2. Automaton 2

4 The New Countermeasure

In contrary to the previously described countermeasures which were introduced by Coron in [6], we intend to randomize the binary algorithm itself. This can be easily done by inserting a random decision in the two algorithms in section

3.1 and 3.2. For example, if we are in state 1 we draw a random variable e. If $e = 0$ we take the path of algorithm 1, else we proceed as in the standard binary algorithm. The finite automaton in figure 3 shows the randomized algorithm according to this idea.

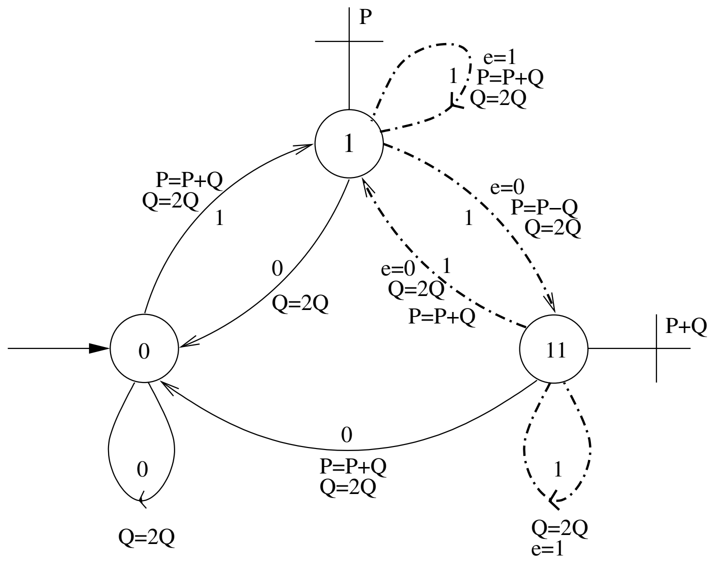

Fig. 3. Randomized Automaton 1

In order to make SPA attacks more difficult, we changed all multipliers so that always only one double or one double and one add (or subtract) is necessary. Note, that the bits of the binary representation don't correspond directly to doubles (resp. adds) anymore! For both, 1 and 0 in the binary representation, there is one path in the algorithm where, for instance, the double operation is performed. In the same manner we modified finite automaton 2 (see figure 4). In both figures, the paths that are randomized are drawn dash-dotted for better visibility. Again, in both figures the input path is marked by a distinct arrow, and the output paths are marked by an additional bar. Intermediate states are represented by circles, and transitions are represented as arrows between them. The initial conditions are again $P = 0$ and $Q = M$. For the sake of completeness we give an iterative algorithm implementing figure 4.

4.1 Analysis of the Randomized Algorithms

As noted before, the binary representation does not correspond directly to the performed operations anymore. A second observation is that due to the fact,

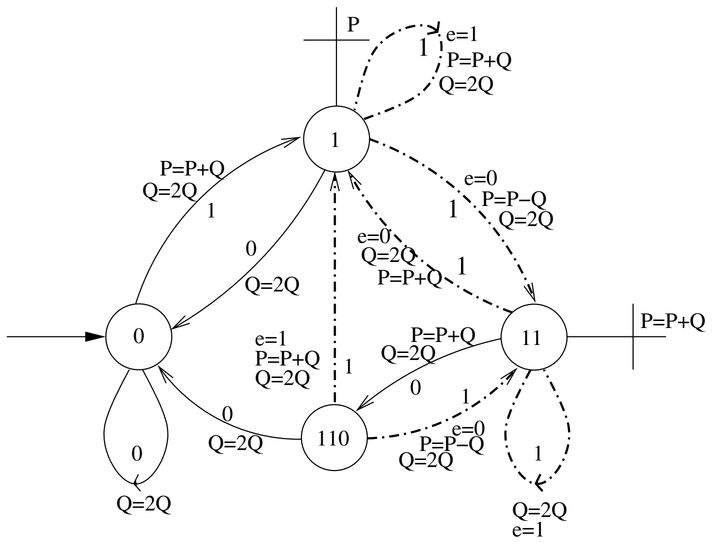

Fig. 4. Randomized Automaton 2

```
randomized_version_automaton2(k,M) {
        state=0;P=0;Q=M;
        while (k>0) {
          if ((k&1) == 0) {
            if (state == 11) P = P+Q;
            state=0;Q = 2*Q;
          }
          if ((k&1) == 1) {
            switch (state) {
               case 1: e=rand();
                       if (e==1) P= P+Q;
                           else P = P-Q and state=3;
                       Q = 2*Q;
               case 11: e = rand();Q = 2*Q;
                        if (e==0) P = P+Q and state=1;
               case 0: P = P+Q;Q=2*Q;state=1;
            }
          }k >>=1;
        }
        if (state==11) {P = P+Q;}
        return P;
}
```

Fig. 5. Iterative Version of the Randomized Automaton 2

that $-P(x, y) = P(x, -y)$ e.g. $-P(x, y) = P(x, x + y)$(the add and subtract operations are basically the same), they aren't distinguishable in the power trace. The difference of add (subtract) and double depends on the underlying number field, and the used coordinates. For instance, as listed in the table in section 2.1, in $GF(2^m)$ with affine coordinates, for both add and double operation, 1 inversion, 2 multiplications and 1 squaring is needed. We consider now some possible scenarios for a power attack :

– **SPA case :** Suppose one has an implementation were the distinction between double and add (subtract) is possible with a single measurement (this could be the case when working with projective coordinates). It would be possible to identify a block of 0's at the beginning of the algorithm. Also, blocks of 0's result more likely in consecutive doubles than blocks of 1's. Basically there are more likely binary representation than others, and this could be used to identify the correct key. But that this is not as easy as mounting an SPA attack on the standard binary algorithm.
– **DPA case :** Let us assume now, that we don't have such a dumb implementation, and therefore the difference between a double and an add (or a subtract) operation is not visible with only one power measurement. Every time the algorithm is performed, it takes due to the randomization a different path. Therefore the sequence of doubles, adds, and subtracts is slightly different. If the random numbers are close to uniform, this makes an attack like the mean method infeasible. But also the correlation method does not work anymore. Because of the randomization, the intermediate values that are attacked, are computed at different times, or are sometimes not even calculated. This washes out the DPA bias signal.

The performance of the randomized algorithms is close to the standard binary algorithm. The following table shows the percentage of additional operations (additions and subtractions) in comparison with the ordinary binary algorithm for various key lengths (which were chosen according to the commonly suggested field sizes). For this table we counted the number of additions and subtractions (the number of doublings is for both variants approximately the same) for several thousands of executions (for several values of k) of the algorithm given in figure 5. The table shows that the additional number of operations is almost independent of the keylength and is approximately 9%.

What can we say about the storage of additional parameters? The speedup in the article of Morain and Olivos uses the bottom-up variant of the binary algorithm, which requires the additional storage of one elliptic curve point. This is obviously a disadvantage. On the other hand, the proposed standard IEEE P1363a, includes this version of the binary algorithm. The other additional parameter which we have to store, is the random bit e.

5 Conclusion

We described an alternative approach for the development of countermeasures against power attacks. Our countermeasure does not depend on any input pa-

Table 1. Performance comparison

Bitlength	Additional operations (mean value)	Variance
112	9%	3%
128	10%	5%
160	9%	3%
192	8%	3%
224	8%	3%
236	7%	3%
384	9%	2%
521	9%	2%

rameters of the binary algorithm, but on the algorithm itself. We've analyzed its efficiency in preventing the mean and the correlation method, and its efficiency in performance. Fact is, that our method prevents recent power analysis attacks largely without the necessity to store large additional parameters or do any precomputations. One can also combine this countermeasure with the other suggested countermeasures to achieve a higher security level.

References

1. E. Biham, A. Shamir, *Power Analysis of the Key Scheduling of the AES Candidates* Second AES Candidate Conference, Rome, March 1999, pp 115-121.
2. S. Chari, Ch. Jutla, J. Rao, P. Rohatgi.*A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*. Second AES Candidate Conference, Rome, March 22-23,1999, pp 133-147.
3. S. Chari, Ch. Jutla, J. Rao, P. Rohatgi.*Towards Sound Approaches to Counteract Power-Analysis Attacks*, Proceedings of Advances in Cryptology-CRYPTO'99, Lecture Notes in Computer Science, vol. 1666, Springer, 1999, pp.398-412
4. C. Clavier, J.-S. Coron, N. Dabbous, *Differential Power Analysis in the presence of Hardware Countermeasures*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer, 2000, pp. 252-263
5. I. Blake, G. Seroussi, N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, Lecture Notes Series 265, Cambridge Universtiy Press
6. J.-S. Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 1999), Lecture Notes in Computer Science, vol. 1717, Springer,1999, pp.292-302
7. J.-S. Coron, L. Goubin, *On Boolean and Arithmetic Masking against Differential Power Analysis*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer, 2000, pp. 231-237
8. J.-S. Coron, P. Kocher, D. Naccache, *Statistics and Secret Leackage*, to appear in Proceedings of Financial Cryptography, Springer-Verlag, February 2000
9. P. Fahn, P. Pearson. *IPA: A New Class of Power Attacks*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 199), Lecture Notes in Computer Science, vol. 1717, Springer 1999

10. D. M. Gordon, *A survey of fast exponentiation methods.*, J. Algorithms, 27, pp. 129-146, 1998
11. L. Goubin, J. Patarin.*DES and Differential Power Analysis.* Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 199), Lecture Notes in Computer Science, vol. 1717, Springer 1999, pp 158-172.
12. M. A. Hasan, *Power Analysis Attacks and Algorithmic Approaches to Their Countermeasures for Koblitz Cryptosystems*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer 2000, pp. 93-108
13. K.F. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory*, Graduate Texts in Mathematics, vol. 84, Springer-Verlag, Fifth printing, 1998
14. D. E. Knuth. *Seminumerical algorithms.* The Art of Computer Programming. T. II, Addison-Wesley.
15. N. Koblitz. *Elliptic Curve Cryptosystems*, Mathematics of Computation, vol. 48, 1987, pp.203-209
16. P. Kocher, J. Jaffe and B. Jun, *Differential Power Analysis*, Proceedings of Advances in Cryptology-CRYPTO'99, Springer 1999, pp. 388-397
17. R. Mayer-Sommer, *Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer 2000, pp. 78-92
18. A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993
19. T.S. Messerges, E. A. Dabbish and R. H. Sloan, *Investigations of Power Analysis Attacks on Smartcards*, Proceedings of USENIX Workshop on Smartcard Technology, May 1999, pp. 151-61.
20. T.S. Messerges, E. A. Dabbish and R. H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*, Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, vol. 1717, Springer 1999.
21. T. S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant Software*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer 2000, pp. 238-251
22. A. Shamir,*Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000), Lecture Notes in Computer Science, vol. 1965, Springer 2000, pp. 71-77
23. F. Morain, J. Olivos. *Speeding up the computation on an elliptic curve using addition-subtraction chains*, Inform. Theory Appl. 24 (1990), 531-543.
24. V. S. Miller. *Use of Elliptic Curves in Cryptography*, Proceedings of Crypto 85, Lecture Notes in Computer Science 218, Springer, 1986, pp. 417-426
25. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Company, 1993.