Longer Randomly Blinded RSA Keys may be Weaker than Shorter Ones

Colin D. Walter

Comodo Research Laboratory
7 Campus Road, Bradford, BD7 1HR, UK
Colin.Walter@comodo.com

Abstract. Side channel leakage from smart cards has been of concern since their inception and counter-measures are routinely employed. So a number of standard and reasonable assumptions are made here regarding an implementation of RSA in a cryptographic token which may be subjected to non-invasive side-channel cryptanalysis. These include blinding the re-usable secret key, input whitening, and using an exponentiation algorithm whose operation sequence partially obscures the key.
The working hypothesis is that there is limited side channel leakage which only distinguishes very imprecisely between squarings and multiplications. For this typical situation, a method is described for recovering the private exponent, and, realistically, it does not require an excessive number of traces. It just requires the modulus to be public and the public exponent not to be too large.
The attack is computationally feasible unless parameters are appropriately adjusted. It reveals that longer keys are much more vulnerable than shorter ones unless blinding is proportional to key length. A further key conclusion is that designers must assume that the information theoretic level of leakage from smart cards can be transformed into usable key information by adversaries whatever counter-measures are put in place.

Key Words. Side channel leakage, power analysis, SPA, DPA, RSA.

1 Introduction

Side channel leakage of secret key information from cryptographic devices has been known publicly for a number of years [1], and very widely since the work of Kocher [6,7]. In the case of RSA, the main software counter-measures to this have included message whitening, key blinding and more complex exponentiation algorithms. These, therefore, form the main assumptions here.

In the past, there were no obvious ways of extracting weak leaked information from this and using it to recover the secret key. Either the leaked information had to distinguish clearly between squarings and multiplications for individual uses of the key [3] or, with less precise leakage, the same key had to be re-used many times in an unblinded state so that the leakage could be averaged to reduce noise [6,2,13].

S. Kim, M. Yung, and H.-W. Lee (Eds.): WISA 2007, LNCS 4867, pp. 303–316, 2007.

However, from the information-theoretic standpoint, it is clear that there can be enough data for the key to be recovered when blinding is used but side channel leakage is imprecise. Here a means for obtaining the key is given for that situation, developed from the case of perfect, but partial, side channel information described by Fouque *et al.* [3]. One of the main contributions here is a metric for evaluating choices and enabling the best to be investigated first.

The first objective is to determine the blinding factor for several dozen cases. This is done by brute force: testing every possible value until one is found which would provide a trace that matches the measured trace sufficiently well under a suitable metric. The analysis is complicated by an unknown factor k equal to the size of the public exponent E. That factor must also be determined from the side channel leakage in the same way, and therefore affects the computational feasibility of the method if E is large. However, there is no obvious way to avoid the exhaustive search.

Once the blinding factors are determined for as many traces as are needed, the second objective is to determine the unblinded private exponent. Its bits are guessed from most to least significant by looking at both possible values and selecting the one which matches the observed leakage better. Incorrect choices are quickly noticed, and corrected by back-tracking and lookahead. This phase of the attack is less computationally intensive than the first, but it requires more traces when the leakage is weaker – a number inversely proportional to the strength of leakage.

The adversary makes use of certain properties of the exponentiation algorithm which lead to the leakage. The standard 4-ary sliding windows [4] considered here has a pattern of squarings and multiplications which contains information about the bit pattern of the exponent. The method applies equally well to any other algorithm with a variable pattern of operations where the variation is derived from a local property of the secret key, such as bit values.

Finally, the complexity of the attack is considered. There is low space complexity and the attack is highly, and easily, parallelisable to make full use of computing resources. The total time complexity is of order which is the product of the public key, the maximum blinding factor and a measure of the unreliability of the side-channel leakage. Thus, it appears to be computationally feasible to extract the key in many normal circumstances.

A significant conclusion is that, for a fixed amount of blinding, longer keys are less secure because blinding factors are determined more accurately. This means that blinding should be increased in proportion to key length in order to thwart the attack.

The organisation of the paper is as follows. The main assumptions, the leakage model, pre-requisite notation and background algorithms are covered in sections §2 to §5. Phase 1 of the attack, during which the blinding factors are recovered, is treated in §6. Phase 2 of the attack, namely the recovery of the secret key, is described in §7. The computational cost is reviewed in §8 and wide-ranging conclusions are drawn in §9.

2 Notation

The n-bit RSA modulus N and public exponent E are assumed to be known by the attacker. His aim is to recover the private key D which is re-used a number of times but only in the blinded form $D_i = D+r_i\phi(N)$ where $r_i < R$ is a small random number (typically up to 32 bits) and $\phi(N)$ is unknown.

The modulus is a product of two primes $N = PQ$ which must be of similar magnitude. For convenience, we assume P and Q have the same number of bits. Then, without loss of generality, $P < Q < 2P$ so that $2\sqrt{N} < P+Q < 3\sqrt{N/2}$ and $\phi(N) = N-(P+Q)+1$ is bounded by

$$N - 3\sqrt{N/2} + 1 < \phi(N) < N - 2\sqrt{N} + 1 \tag{1}$$

This interval has length less than $\frac{1}{8}\sqrt{N}$, so that more than half of the most significant bits of $\phi(N)$ are known by the attacker from those of N.

The exponents D and E are related by

$$D\times E \;=\; 1+k\phi(N) \tag{2}$$

for some k. Without loss of generality, let D be the smallest non-negative solution to this congruence, so that $D < \phi(N)$ and $k < E$. When key D is re-used many times, blinding factors are normally added to produce the randomly different exponents which are actually used for decryption or signing [6]:

$$D_i \;=\; D+r_i\phi(N) \tag{3}$$

where r_i is a random number, usually of 16 to 32 bits. Thus,

$$D_i \;=\; \frac{1 + (k+r_iE)\phi(N)}{E} \tag{4}$$

Let R be an upper bound on such r_i. Then the coefficient $k+r_iE$ of $\phi(N)$ is, in effect, a random number in the range 1 to RE. (So it is irrelevant whether k and D were chosen minimally in equation (2)). In equation (4) the adversary is initially only interested in the most significant half of the bits. He ignores the 1 and approximates $\phi(N)/E$ by computing N/E. By the earlier remarks this gives him at least the top $n/2$ bits of $\phi(N)/E$. So,

$$D_i \;\approx\; (k+r_iE)N/E \tag{5}$$

The attacker now has to generate each of the RE possible values of the random coefficient of N/E in order to obtain a set containing an approximation to the value of D_i used in the exponentiation which he has observed.

3 The Exponentiation

For convenience we assume that the exponentiation algorithm is 4-ary sliding windows using the re-coding in Fig. 1 [4,5]. This uses a window of 1 bit width

Input: Binary $D = (b_{n-1}...b_2b_1b_0)_2$
Output: Recoding $D = (d_{m-1}...d_2d_1d_0)$

```
i ← 0 ;
m ← 0 ;
While i < n do
If b_i = 0 then
Begin
    d_m ← 0 ;
    i ← i+1 ;
    m ← m+1 ;
End
else
Begin
    d_m ← 2b_{i+1} + 1 ;
    i ← i+2 ;
    m ← m+1 ;
End
```

Fig. 1. Quaternary Sliding Windows Recoding.

when there is a digit zero in the recoded exponent, and otherwise a window of 2 bits width, for which the digit is 1 or 3. Although this does not provide the same protection against side channel cryptanalysis as the square-and-always multiply algorithm, it is more time efficient even than the usual square-and-multiply algorithm and also creates some difficulty for an attacker who may have to distinguish whether the multiplications pertain to digit 1 or digit 3.

This algorithm, or its fixed-width equivalent, is typical of a smart card because of its speed and low storage overhead: only the first and third powers of the input message need storing for the exponentiation. Both algorithms generate a pattern of squarings and multiplications which is related to occurrences of the zero digit. This is the property that can be exploited here by an attacker.

4 The Leakage Model

With expected counter-measures in place it is unrealistic to assume that every long integer multiplicative operation in an RSA exponentiation can be identified and distinguished as a squaring or not. However, some imperfect deductions may be possible from a side channel trace, particularly in contactless cards where severe resource limitations and an explicit aerial limit the scope and effectiveness of any counter-measures. The following two leakage scenarios are likely in practice. Others are certainly possible.

First, because the conditional subtraction in a Montgomery modular multiplication ([8], *see* Fig. 2) consumes a number of extra clock cycles, there is a possibility that it may be observed in a side channel trace via the longer time for the operation. The slightly different frequencies of the subtraction for squares and multiplies mean that each occurrence or absence of the subtraction makes

a square or multiplication marginally more likely [13]. As previous attacks have been unable to use this information in the presence of exponent blinding and message whitening, implementors may not perceive the leakage as a threat when such counter-measures are in place. One can therefore expect many of them to prefer more widely applicable code which includes the conditional subtraction, despite the existence of straightforward and efficient alternatives [12].

Secondly, the data loading cycles for multiplications and squarings are different and therefore vulnerable. For example, the Hamming weight of the words of the arguments may leak when they pass along the internal bus [7]. A squaring is almost certain where the Hamming weights are equal, and a multiplication must be the case if they are different. However, this information is usually well submerged in noise, and in a well designed implementation it should only yield a minimal bias towards a squaring or a multiplication.

The above are very much more realistic leakage models than that of [3] where it was assumed that each multiplicative operation was known to be a squaring or a multiplication. In practice, only weak probabilistic information is known.

In order to obtain specific measures of implementation strength, the attack here is modelled on the level of data leakage from observing every conditional subtraction in Montgomery modular multiplication. However, the attack is generic, and applies to both of the above scenarios as well as many others.

Input: A and B such that $0 \leq A, B < N < r^n$ and N prime to r.
Output: $C = ABr^{-n} \bmod N$

```
C ← 0 ;
For i ← 0 to n-1 do
Begin
    q_i ← -(c_0+a_i b_0)n_0^{-1} mod r ;
    C ←  (C+a_i B+q_i N) div r ;
End ;
{ Assertion: Cr^n ≡ A×B mod N and ABr^{-n} ≤ C < N+ABr^{-n}}
If C ≥ N then C ← C-N ;
```

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

5 Selecting the Leakiest Traces

The word-based algorithm for Montgomery multiplication (MMM) is given in Fig. 2 where the digits a_i, b_i etc. are for the base r representation of the long integers A, B etc. From the assertion after the loop it is easy to establish the frequency of the conditional subtraction under the reasonable assumption of the output residues being uniformly distributed modulo N. The probability is proportional to the fraction of the interval which is greater than N, namely $ABr^{-n}N^{-1}$. For a typical multiplication with independent arguments, this can

be summed with respect to A and B over the interval $[0, N)$ to obtain the average probability of

$$p_M \approx \frac{1}{4} N r^{-n} \tag{6}$$

Similarly, setting $A = B$ and summing gives the probability of the subtraction for a squaring, namely

$$p_S \approx \frac{1}{3} N r^{-n} \tag{7}$$

The difference between p_M and p_S shows that the occurrences of a conditional subtraction indicate a squaring is slightly more likely to be the case than a multiplication. The difference, however, is small. Early attacks on Montgomery's algorithm relied on being able to perform hundreds or thousands of exponentiations with the same key in order to observe enough subtractions to conclude with high probability whether the operation was a squaring or a multiplication.

The formulae (6) and (7) also indicate that decreasing N or increasing the number of iterations n will reduce the occurrences of the conditional subtraction and so make the algorithm more secure.

However, the multiplications in individual exponentiations are not as random as used for the formula (6). For 4-ary sliding windows, one of the two pre-computed powers of the input message is used as one of the arguments, the other being a random output from an earlier Montgomery multiplication. So only one argument is uniformly distributed in a given exponentiation. Let A be the fixed input to such a multiplication. Then, summing $ABr^{-n}N^{-1}$ with respect to B yields the true probability of a subtraction, *viz.*

$$p_A \approx \frac{1}{2} A r^{-n} \tag{8}$$

Thus, when the pre-computed powers of the input are small (resp. large) there will be very few (resp. many) conditional subtractions resulting from multiplications. This increases the probability of distinguishing between squarings and multiplications. Overall, this will be noticed by an adversary because the total number of conditional subtractions will be less (resp. greater) than the average for such exponentiations. This provides the opportunity for the adversary to select the leakiest traces with very little computational effort.

Similarly, in the Hamming weight leakage scenario, instead of an enhanced or reduced frequency of conditional subtractions from large or small values of A, the adversary homes in on the argument pairs A, B which are the highest Hamming distance apart. They have the highest probability of being multiplications. This occurs most frequently when the re-used, pre-computed multiplier A has the highest number of extreme Hamming weights. So, by screening for extreme Hamming weights, traces which leak significantly more information than average can be identified easily by the adversary.

In both leakage models, the attacker can therefore begin by selecting side channel traces which yield the greatest amount of information, and these can be chosen without excessive computational effort for the initial phase of data capture, signal processing and selection.

6 The Attack: Phase 1

In the leakage scenarios of §4, the attacker is expected to obtain little or no useful information about a multiplicative operation in many cases, and only a very weak probability in favour of a squaring rather than a multiplication (or *vice versa*) in other cases. However, as described in §5, he begins his attack by collecting as many traces as possible and selecting those for which the leakage promises to be greatest. Phase one of his attack then progresses as follows.

Suppose he has selected a promising trace corresponding to the use of the blinded exponent D_i. He first determines the top half of the digits of $\phi(N)/E$ as in §2 and then guesses the values of k and r_i. Equation (5) gives him a possible approximation $D_i{}'$ for D_i. He then compares the side channel leakage expected from $D_i{}'$ with that obtained from D_i and discards the guessed pair (k, r_i) if the match is poor. Repeating this for all pairs leaves him with a set S_i of the most likely blinding values for D_i. This process is repeated with more traces – enough for him to complete the second phase successfully.

The decision about whether guesses are good enough is based on a metric $\mu(tr(D_i), ops(D_i{}', m))$. The first parameter $tr(D_i)$ is the processed side channel leakage from use of the unknown blinded key D_i. Specifically, it is a list of probabilities $pr(op)$ that the operations op of the exponentiation using D_i were squares rather than multiplications. Thus $tr(D_i) = [pr(op_s), pr(op_{s-1}), ..., pr(op_3), pr(op_2), pr(op_1)]$ where $s = len(tr(D_i))$ is the total number of operations in the exponentiation with key D_i. In the second parameter, $D_i{}'$ is the bit sequence for the guessed value of D_i, and $ops(D_i{}', m)$ is the sequence of multiplicative operations carried out in an exponentiation with key $\lfloor D_i{}'/2^m \rfloor$. So the m least significant bits of $D_i{}'$ are irrelevant, and need not have been guessed yet. This parameter will be a list containing, say, '0' to denote a squaring, and '1' a multiplication. In this phase we will set $m = n/2 + \log_2 R$.

If the side channel leakage $tr = tr(D_i)$ for D_i indicates with probability tr_j that the jth operation was a squaring and the jth operation in $ops(D_i{}', m)$ is also a squaring then $tr_j - \frac{1}{2}$ is added to the metric. However, if a multiplication occurs as the jth element in $ops(D_i{}', m)$, then $\frac{1}{2} - tr_j$ is added. So 0 is added if the leakage provides no information since then $tr_j = \frac{1}{2}$, but there is a positive contribution to the sum when the operation in $tr(D_i)$ is more likely to be the same as that in $ops(D_i{}', m)$, and there is a negative contribution when the two operations are more likely to be different. Thus, the sum for calculating the metric is

$$\mu(tr, ops(D', m)) \quad = \sum_{1 \le j \le nops(\lfloor D'/2^m \rfloor)} (-1)^{ops(D', m)_j} (tr_j - \frac{1}{2}) \qquad (9)$$

when $ops(D', m)_j \in \{0, 1\}$ as suggested above, and $nops(D'')$ is the number of operations in exponentiating to the power D''. Here the lists tr and $ops(D', m)$ are in temporal execution order assuming an exponentiation algorithm which processes bits from most to least significant. This correctly aligns corresponding elements of the lists when the lowest bits of D' are ignored.

If the guess (k, r_i) is correct, the value $D_i{}'$ provides the same pattern of squarings and multiplications as D_i over the first (approximately) half of the operations of the exponentiation. So, unless the noise is overwhelming, this should maximise the value for the sum when restricted to those operations. Therefore larger values for μ imply a better match between the guessed value $D_i{}'$ and the targetted exponent D_i, whereas smaller and negative values indicate a poor match. Because of the unknown difference between N and $\phi(N)$, the $n/2+\log_2 R$ least significant bits of $D_i{}'$ are unreliable even when (k, r_i) is guessed correctly. By taking $m = n/2+\log_2 R$ the operations corresponding to them are ignored.

The metric could be improved by taking account of the dependence between consecutive operations: for example, the probable occurrence of a multiplication implies that the next operation is more likely to be a squaring. Schindler [9,11,10] treats this in detail and provides theory about the best choice of metric.

6.1 Phase 1 Simulation

In our simulation, values were chosen which correspond to a leaky implementation of MMM where every conditional subtraction is observed and $N \approx r^n$. Conditional subtractions were generated randomly with frequencies in accordance with the model described in §5. There was no selection of "better" traces on the grounds of fewer or more subtractions than normal. Since conditional subtractions occur with slightly greater frequency for squarings than for multiplications, the metric was (arbitrarily) incremented by $p_j - \frac{1}{2} = 0.1$ for every conditional subtraction in the trace when there was a squaring in the guessed value and therefore incremented by $\frac{1}{2} - p_j = -0.1$ (i.e. decremented) for every conditional subtraction that coincided with a multiplication.

$\log_2 RE$	8	12	16	32	48
$n = 384$	7.9×10^{-3}	8.0×10^{-3}	8.2×10^{-3}	5.0×10^{-3}	3.6×10^{-3}
512	2.7×10^{-3}	2.4×10^{-3}	3.4×10^{-3}	2.0×10^{-3}	1.0×10^{-3}
768	5.3×10^{-4}	3.2×10^{-4}	1.0×10^{-3}	1.4×10^{-4}	1.2×10^{-4}
1024	2.0×10^{-5}	1.9×10^{-5}	8.7×10^{-4}	1.7×10^{-5}	8.0×10^{-6}
1536	$< 2.5\times10^{-7}$	$< 2.5\times10^{-7}$	$< 2.5\times10^{-7}$	...	...

Table 1. Proportion of Guesses returning a Higher Value of μ than the Correct One.

With only this weak knowledge to distinguish between squarings and multiplications, the "best" guess is rarely the correct one. The correct values are ranked among the best, but do not usually come top. Therefore, to assess the feasibility of the attack, it is necessary to know the size of the set S of best guesses which is big enough to include the correct guess. This depends on the strength of the leakage. With the parameters just described, the results in Table 1 were obtained. It gives a good indication of how well the matching process

works and shows the minimum proportion of all guesses which must be considered if the correct one is not to be excluded. For example, with a modulus of $n = 1024$ bits, and $RE = 2^{32}$, the leakage of interest is from the top 512 or so bits. Then the metric μ places the correct values (k, r_i) above all but about $RE \times 1.7 \times 10^{-5} \approx 2^{16}$ incorrect values, on average.

In information theoretic terms, the metric has extracted about 16 bits from the side channel, i.e. about 1 bit in every $512/16 = 32$. This is the case for all the entries in the table: they all correspond to about 1 bit per 32 in the top half of the key, i.e. $n/64$ bits in total. An improved metric is possible (e.g. taking into account multiplications having to be next to squarings) and this would enable more information bits to be obtained. However, for 2048-bit keys (not tabulated), this means about 32 bits' worth of information is recovered, so that k and r_i should be determined almost uniquely when $RE \leq 2^{32}$. This is indeed what was found in the simulation. Clearly, longer keys are more vulnerable:

– *For a given size of blinding and public exponent, the longer the key, the more likely* (k, r_i) *is to be guessed correctly and uniquely.*

The figures in the table show little effect from increasing RE. k and r_i blind information equivalent to about $\log_2 RE$ bits' worth of operations. However, longer blinding factors also seem to constrain the pattern of the blinded key more tightly. With these conflicting forces, the average success of the method is little changed: the number of bits leaked depends almost entirely on the length n of the key. Consequently, for a given key length, the same proportion of choices (k, r_i) are removed irrespective of the value of RE. Of course, the number of accepted pairs must still increase directly in proportion to RE. Thus,

– *Typical leakage from* 2048*-bit or longer keys will usually reveal* (k, r_i) *correctly with current standards for key blinding and a small public exponent;*

and

– *In these cases, an exhaustive search is computationally feasible to find the correct blinding factors* (k, r_i)*.*

Incidentally, a powerful counter-measure in the case of Montgomery conditional subtractions is just to halve the modulus. This halves the number of conditional subtractions, and so halves the number of bits which are leaked.

6.2 Combining Traces to Determine k in Phase 1

The leakage from t traces can be processed for an outlay of t times the effort for one. If these traces are independent, t times as much bit information is extracted. Thus, a very small number of traces should result in k being determined with some confidence, since the same k is used in all cases. In fact, the correct value of k should have been guessed for all or almost all traces, and, if there is any bias, the correct value for k should be one of the most popular among the best guesses for an individual trace.

Guesses at k are ranked as follows. For each sufficiently good guess $k+r_iE$, the value of $k = (k+r_iE) \bmod E$ is extracted and the associated value of the metric μ is added to the weighting of k. The higher the total weight for a guess at k, the more likely that value is to be correct. The possible values of k are then considered in descending order of weight in Phase 2, the heaviest first.

Our simulation did not investigate how much this ranking reduces the search space in Phase 2 as a function of t; from the information theoretic point of view, it seems possible that k is almost completely determined by only a very small number of traces. This is an important detail that still needs to be researched as it affects the effectiveness of the blinding.

7 The Attack: Phase 2

Let S_i be the set of plausible guesses at (k, r_i) for the ith trace, and suppose S_i is partitioned into subsets S_{ik} which share the same k. Armed with these sets, the adversary progresses to phase 2, which is the recovery of the remaining, least significant bits of $\phi(N)$. He repeats this phase for each k separately, choosing the most likely k first. $\phi(N)$ is constructed bit by bit from the most significant end. The first half of $\phi(N)$ was obtained already from the public N and equation (1).

Let $\phi(N)_j = (\phi_{n-1}\phi_{n-2}...\phi_j)_2$ be the part of $\phi(N)$ already determined, so ϕ_{j-1} is the next bit to be guessed. Let $\Phi_j = (\phi_{j-1}\phi_{j-2}...\phi_{j-w})_2$ be a guess at the next w bits of $\phi(N)$. For each possible value of word Φ_j, the right side of equation (4) is evaluated with $\phi(N)_{j-w}$ in place of $\phi(N)$. This yields an approximation $D_{r_i,j-w}$ to D_i in which only the most significant $n-j+w$ bits are of interest. The same metric as in Phase 1 is used again to measure how well this matches the leakage from D_i, namely $\mu(tr(D_i), ops(D_{r_i,j-w}, j-w+\log_2 R))$. (As before, at the point before division by E, we ignore the lowest $\log_2 RE$ bits containing a contribution from ϕ_{j-w} because they are too contaminated by the carries up from less significant bits of $\phi(N)$.) For the given k, the sum

$$\mu_w(k, j, \Phi_j) = \sum_i \sum_{r_i \in S_{ik}} \mu(tr(D_i),\ ops(D_{r_i,j-w},\ j-w+\log_2 R)) \qquad (10)$$

over all guesses is used to assess the worth of the choice for Φ_j. The leading bit of Φ_j from the maximum $\mu_w(k, j, \Phi_j)$ is selected as the value for ϕ_{j-1}.

Correct bit choices amplify any peak (i.e. maximum) values of the metric μ_w whilst incorrect choices decrease it. Moreover, previous mistakes reduce any peaks. When that happens, it is necessary to backtrack and select the most promising previous value. The difference between the two cases is determined using a threshold value for the metric which is obtained by experience. When it becomes too low for every value of Φ_j, it is necessary to backtrack and select the most promising previously untried value. The least significant bits of Φ_j are partly masked by carries up, and contribute less to the peak values than the more significant bits. So only the top one or two bits of the best Φ_j are chosen each time. In this way the bits ϕ_j are chosen from most to least significant. Once most bits have been guessed, the final $\log_2 E$ bits are fully determined by the division being exact in equation (4).

w	1	2	3	4	6	8
t = 25	0.613	0.767	0.833	0.868	0.914	0.930
t = 50	0.642	0.819	0.896	0.939	0.973	0.989
t = 100	0.673	0.846	0.922	0.954	0.981	0.994
t = 250	0.706	0.863	0.930	0.971	0.991	0.995

Table 2. Probability of predicting the correct bit of $\phi(N)$ from t correct guesses r_i with w lookahead bits when $n = 1024$ and $\log_2 RE = 16$.

7.1 Phase 2 Simulation

For the simulation it was assumed that the correct (k, r_i) had been chosen for each i, i.e. that k had been deduced correctly and for the ith trace only the correct r_i had been selected. So $|S_{ik}| = 1$ and $|S_{ik'}| = 0$ if $k' \neq k$. From the conclusions about Phase 1, this should usually be the case for long keys.

As long as there is a reasonable probability of detecting the correct bit each time, all of $\phi(N)$ can be determined. Typical probabilities can be seen in Table 2. There seems little to be gained from having more than 100 traces; more is achieved by having more lookahead bits. In fact, the probability of picking the wrong bit seems to fall exponentially as the number w of lookahead bits increases[1]. From Table 3, $w \geq 8$ allows a significant proportion of keys to be recovered if the k and the randoms r_i have been guessed correctly. The figures are for an implementation of the algorithm without backtracking. When incorrect bits are predicted, the process does not recover and random bits are generated thereafter. With most bits being correct, backtracking is a cheaper alternative to solve this than increasing the number of lookahead bits. Assuming that Table 2 probabilities are constant over the length of the key and are independent of the key length, it is possible to compute the probability of successfully recovering the key: approximately $p^{n/2}$ where p is the table entry and n the key length.

Table 3 gives these probabilities as obtained from a simulation with 100 traces and $w = 8$. This corresponds to $p = 0.9973$. With 10 lookahead digits the simulation shows there is a 60% chance of recovering 2048-bit keys, and this corresponds to $p = 0.999512$. Lastly, with 50 traces but w varying dynamically between 8 and 16 as necessary, 2048-bit keys were recovered in 11% of cases. Since the values of k and r_i from Phase 1 will be mostly correct for 2048-bit keys with $\log_2 RE \leq 32$,

[1] The maximum values of $\mu_w(k, j, \Phi_j)$ were computed where Φ_j ranged over i) values with $\phi_j = 0$ and ii) values with $\phi_j = 1$. The difference between these was a good indicator of the reliability of the choice of ϕ_j. Increasing w just for the cases for which this difference was smallest led to a remarkable improvement in accuracy. Moreover, decreasing w for other cases led to a considerable computational saving. Many 2048-bit keys were recovered successfully using just 50 traces and varying w between 8 and 16.

– *It is computationally feasible to recover a substantial number of 2048-bit keys using 50 traces, current standards for random blinding, typical small public exponents, and expected levels of weak side channel leakage.*

n	512	768	1024	1536	2048
prob	0.50	0.40	0.29	0.13	0.04

Table 3. Probability of success in determining $\phi(N)$ from $t = 100$, correct r_is and key length n with $w = 8$ lookahead bits, no back-tracking and $\log_2 RE = 16$.

7.2 The Case of some Incorrect Phase 1 Deductions

Now consider the case where not all pairs (k, r_i) are correct. If (k, r_i) is incorrect then the above process applied only to this pair (i.e. $t = 1$ and $|S_{ik}| = 1$) would result in choosing the lower bits of $\phi(N)$ to satisfy (4) with the incorrect values (k, r_i) and the correct D_i. This makes the lower bits incorrect by a multiplication factor of $(k'+r'_iE)/(k+r_iE)$ where (k', r'_i) is the correct pair. Moreover, for these bit choices the metric retains the peak values associated with a correct choice. So, without the context of other traces, the error will remain undetected and the pair (k, r_i) cannot be removed from consideration.

Thus, if the above process is performed with a set of pairs (k, r_i), some of which are correct and others incorrect, then the incorrect values predict random bits, while the correct ones predict the correct bits. This averages to a weaker prediction of the correct bits. However, the incorrect choices become more apparent as more correct bits are appended to $\phi(N)$. Eventually this is noticed and those choices can be dropped to speed up the process. It is easy to choose threshold values for the metric – several standard deviations below the average, say – to guide this decision.

So the Phase 2 process is applied to all the outputs of Phase 1 for a given k, i.e. every $(k, r_i) \in S_{ik}$ for every trace, and the sum of all the metric values is used to choose the lower bits of $\phi(N)$. Clearly, however, the limiting factor in this phase is the ratio of correct to incorrect predictions (k, r_i). If this is too small it will not be possible to identify correct bits through peaks in the value of the metric. Table 1 shows that key length is a very strong contributor to this: longer keys improve the ratio, making recovery of $\phi(N)$ much easier.

7.3 Comparison with Fouque

In this algorithm the bits of $\phi(N)$ are determined in the reverse order from that used by Fouque *et al.* [3]. This has several advantages. It makes the transition between the known upper half of $\phi(N)$ and unknown lower half seamless, it allows the metric easily to include the value of all previous decisions, and it allows the division by E to be done straightforwardly. The problems of carry influence in the multiplications of equation (4) is similar for both directions.

8 Complexity

The first phase has time complexity $O(REt\log(RE))$ where t is the number of traces needed to complete the second phase, and depends on the level of leakage. This complexity results from an exhaustive search over all possible (k, r_i). It was remarked that there was an information leakage which is proportional to the length of the traces. Therefore, recovering the $\log_2(RE)$ bits of each (k, r_i) only requires processing a part of the traces with length $O(\log(RE))$, not the whole length. Space is not an issue in this phase as only one pair (k, r_i) need be considered at any one time. The pairs are treated independently and so the work can be completely parallelised. For standard choices of $E = 2^{16}+1$, $R = 2^{32}$ and a similar level of leakage to the example, this is clearly computationally feasible.

In the second phase the worst situation is that all RE guesses are considered for every trace at each bit selection, making a total time complexity $O(REnt)$, which is at worst similar to the first phase. However, if only $R'E'$ guesses survive then the complexity is reduced to $O(R'E'nt)$. This assumes that metrics do not have to be recomputed over the whole length of the trace every time another bit is guessed; instead the incremental effect of the new bit is used to update the preceding value. This approach requires $O(R't)$ space as different values of k are processed sequentially. The second phase requires strong leakage or a high ratio of correct pairs (k, r_i) to have a chance of working. Therefore practical limitations on the number of traces that can be obtained guarantees that space will not be the overriding problem. Furthermore, the work can be parallelised without difficulty at least as far as distributing the effort for each k to different processors. This would reduce the time complexity by a factor of $O(E)$.

9 Conclusion

The scope of the attack of Fouque *et al.* [3] has been extended to include imprecise leakage by introducing a practical metric which prioritises the selection of guesses at the random blinding factors and bits of $\phi(N)$ for an RSA modulus N. Both attacks target the typical set-up for RSA decryption/signing in a smartcard with standard counter-measures which include exponent blinding.

It was found that very weak, imprecise leaked data could be successfully manipulated to reduce the ambiguity in the blinding factors by a factor essentially proportional to the length of the keys, so that the blinding factors are fully determined when the key is long enough. For typical choices of public exponent and blinding parameters, and a leakage rate equivalent to only 1 bit per 32 bits of key per trace, the blinding factors can be recovered correctly for keys above about 2048 bits in length.

Reconstruction of the unknown lower bits of $\phi(N)$ requires most of the blinding factors to be recovered correctly and sufficiently many traces to be available. With a leakage rate of 1 bit per r key bits, $1.5r$ traces suffice to recover $\phi(N)$ and hence factor N without any need for an expensive search. In a simulation, a sizeable proportion of 2048-bit keys were successfully recovered using leakage

from only 50 traces (r=32). Thus the attack is certainly computationally feasible with only weak, imprecise leakage.

So longer keys were found to be *more vulnerable*. The best counter-measure is to ensure that blinding increases with key length at least until it becomes computationally infeasible to test every blinding value individually. The attack illustrates that the information theoretic level of leakage can be into converted successfully into the secret key even in the presence of a typical collection of standard counter-measures.

References

1. *Portable Data Carrier including a Microprocessor*, Patent 4211919, US Patent and Trademark Office, July 8, 1980.
2. 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.
3. P.-A. Fouque, S. Kunz-Jacques, G. Martinet, F. Muller & F. Valette, *Power Attack on Small RSA Public Exponent*, Cryptographic Hardware and Embedded Systems – CHES 2006, L. Goubin & M. Matsui (editors), LNCS **4249**, Springer-Verlag, 2006, pp. 339–353.
4. D. E. Knuth, *The Art of Computer Programming*, vol. **2**, "Seminumerical Algorithms", 3rd Edition, Addison-Wesley, 1997.
5. Ç. K. Koç, *High Radix and Bit Recoding Techniques for Modular Exponentiation*, International J. of Computer Mathematics, **40**, no. 3-4, 1991, pp. 139–156.
6. 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.
7. 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.
8. P. L. Montgomery, *Modular Multiplication without Trial Division*, Mathematics of Computation **44**, no. 170, 1985, pp. 519–521.
9. 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.
10. W. Schindler, *On the Optimization of Side-Channel Attacks by Advanced Stochastic Methods*, Public Key Cryptography (Proc. PKC 2005), S. Vaudenay (editor), LNCS **3386**, Springer-Verlag, 2005, pp. 85–103.
11. 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.
12. 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.
13. 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.