Longer Keys may facilitate Side Channel Attacks

Colin D. Walter

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

Abstract. Increasing key length is a standard counter-measure to cryptanalysis. However, longer key length generally means greater side channel leakage. For embedded RSA crypto-systems the increase in leaked data outstrips the increase in secret data so that, in contrast to the improved mathematical strength, longer keys may, in fact, lead to lower security. This is investigated for two types of implementation attack. The first is a timing attack in which squares and multiplications are differentiated from the relative frequencies of conditional subtractions over several exponentiations. Once keys are large enough, longer length seems to decrease security. The second case is a power analysis attack on a single m-ary exponentiation using a single k-bit hardware multiplier. For this, despite certain counter-measures such as exponent blinding, uncertainty in determining the secret bits decreases so quickly that longer keys appear to be noticeably less secure.

Key Words: RSA Cryptosystem, Key Length, Side Channel Attacks, Timing Attack, Power Analysis, DPA.

1 Introduction

So-called *side channel attacks* on smartcards to discover secret keys contained therein follow a well-established tradition pursued by the military and secret services, and exemplified by the long-running Tempest project of the US [27]. That project concentrated on detecting and obscuring electro-magnetic radiation (EMR) and led to both heavily shielded monitors (for those based on electron beam technology) and TV detector vans. EMR can be, and is, used to break smartcards – but with a somewhat smaller aerial, one some 3mm long or less [5]. If correctly placed and set up with sufficiently sensitive equipment, these can detect useful variations in EMR non-invasively.

Side-channel leakage occurs through data dependent variation in the use of resources such as time and hardware. The former results from branching in the code or compiler optimisation [2, 9], and the latter manifests itself through current variation as well as EMR [10, 12, 13]. For the RSA crypto-system [17], conditional modular subtractions should be removed to make the time constant

M. Matsui and R. Zuccherato (Eds): SAC 2003, LNCS 3006, pp. 42–57, 2004.

[18, 19, 21]. Bus activity is the major cause of power variation, with a strong relationship between it and the Hamming weight of the data on the bus. Instructions and memory locations pass along the bus and, in the context of the limited computational resources of a smartcard, potentially also large quantities of data. This is partly solved by encryption of the bus [1, 11].

For all the popular crypto-systems used in practice where the key length is variable, the greater the key length, the greater the *mathematical* strength of the system against attack is believed to be. Indeed, a brute force attack will take time exponential in the key length. However, longer key lengths require more computation for encryption and decryption or signature and verification. Hence there is more data which leaks through timing, current and EMR variation. In an embedded crypto-system to which an attacker has access, such as a smartcard, a valid question to ask is whether or not the increased data from side channel leakage actually makes longer keys *more* vulnerable to attack.

In the symmetric crypto-systems of DES, 3-DES and AES [25, 26], the block length is fixed and the number of rounds is proportional to the key length (comparing DES with 3-DES, and AES with different choices for its parameter Nk). Hence the data leakage is also proportional to the key length and the *implementation* strength of the cipher is unlikely to decrease as key length increases.

However, public key cryptography such as in RSA, DSA, ECC, Diffie-Hellman or El-Gamal [3, 4, 14, 17, 24], usually involves exponentiation in some form, where the block length and exponent are proportional to the key length. Assuming multiplication of double-length arguments takes four times the time for single-length arguments on the same hardware, total decryption/signing time is proportional to the cube of the key length. Consequently, *more* leaked data is available per key bit as key length grows. Indeed, if the multiplicative operations of the exponentiation are performed sequentially using one of the standard algorithms [7, 8] and no form of blinding, then there is more data per exponent bit for longer key lengths and one should expect the implementation strength to decrease.

Two attacks illustrate that such an outcome may be possible from increasing the key length. The first is a timing attack [21] whose success is apparently easier for very short and very long key lengths. By its very nature, the attack assumes essentially none of the currently expected standard counter-measures which would ensure no data-dependent time variations and would introduce random variation in the exponent. Each key bit is determined by data proportional to the square of the key length.

The second attack [20] requires more expensive monitoring equipment to perform and uses power analysis and/or EMR. It is applied to a single exponentiation and so avoids any difficulty arising from employing exponent blinding as a counter-measure. Key bits are determined independently with each using all available data, that is, with data proportional to the cube of the key length. Individual exponent bits are now identified correctly with an accuracy which increases sufficiently quickly to compensate for the increased number of key bits. Consequently the attack becomes easier with longer key lengths.

Having assessed the vulnerabilities, our conclusion is that indeed increased key length will normally lead to a weaker implementations unless appropriate counter-measures are taken. As this is counter-intuitive, it achieves the main aim of the paper, namely to provide the justification for cautioning readers strongly against the temptation to assume that counter-measures to cryptanalysis can be used successfully as counter-measures to side channel leakage.

2 Security Model

The contexts for the two attacks [20, 21] are slightly different, but, for convenience, in both cases we assume a similar, but realistic, scenario. For each, a smartcard is performing RSA with limited resources and must be re-usable after the attack. The attacker is therefore limited in what he is allowed to do: he can only monitor side channel leakage. He cannot choose any inputs, nor can he read inputs or outputs. In most well-designed crypto-systems, the I/O will be blinded and the attacker will be able to see at most the unblinded data which is not used directly in the observed computations. However, the attacker is allowed to know the algorithms involved, perhaps as a result of previous destructive studies of identical cards, insider information and public specifications. His goal is to determine the secret exponent D whether or not the Chinese Remainder Theorem has been used, and he may use knowledge of the public modulus M and public exponent E to confirm proposed values. It is assumed that the m-ary exponentiation algorithm is used, but similar arguments apply to other classical (i.e. non-randomised) algorithms, such as sliding windows.

The timing attack [21] assumes the use of a modular multiplication algorithm which includes a final, conditional subtraction of the modulus. We assume the consequent timing variations enable the attacker to record accurately almost all occurrences of these subtractions. He then observes a number of exponentiations for which the same, unblinded exponent is used. We demonstrate the attack using an implementation of Montgomery's method which is described in the next section.

Power use, and hence also EMR, varies with the amount of switching activity in a circuit. The average number of gates switched in a multiplier is close to linear in the sum of the Hamming weights of the inputs, and the same is true for the buses taking I/O to and from the multiplier. So, by employing a combination of power and EMR measurements from carefully positioned probes [5], it should be assumed that an attacker can obtain some data, however minimal, which is related to the sum of the Hamming weights of these inputs. His problem is to combine these in a manner which reveals the Hamming weights with sufficient accuracy to deduce the digits of the exponent. The differential power analysis (DPA) attack [20] shows how this might be done from observations of a *single* exponentiation. Hence it does not matter if the exponent has been masked by the addition of, say, a 32-bit random multiple of $\phi(M)$ [9].

3 Notation

As above, we assume an n-bit modulus M and private exponent D for the RSA crypto-system. Ciphertext C has to be converted to plaintext C^D mod M using a single, small k-bit multiplier. Hence, except for the exponent, the n-bit numbers X involved in the exponentiation are represented using base $r = 2^k$ and (non-redundant) digits x_i $(0 \le i < s$, say) in the range $[0, r)$. Thus $X = \sum_{i=0}^{s-1} x_i r^i$ and, without loss of generality, $n = ks$.

The exponent D is represented with a different base m, typically 2 or 4, depending on the exponentiation algorithm. Exponentiation is usually performed using the binary "square-and-multiply" algorithm, processing the exponent bits in either order, or the generalisation of the most-to-least significant case, called m-ary exponentiation [7, 8], in which D is represented in radix m using, say, t digits, and some powers of $C^{(i)} = C^i$ mod M $(1 \le i < m)$ which are pre-computed:

The m-ary (Modular) Exponentiation Algorithm

```
C^(1) ← C ;
For i ← 2 to m-1 do
    C^(i) ← C^(i-1) × C mod M ;
P ← C^(d_{t-1}) ;
For i ← t-2 downto 0 do
Begin
    P ← P^m mod M ;
    If d_i ≠ 0 then P ← P×C^(d_i) mod M ;
End ;
```

Output: $P = C^D$ mod M for $D = \sum_{i=0}^{t-1} d_i m^i$

The modular products here are too large for the smartcard multiplier to perform in one operation. Typically a form of Montgomery's modular multiplication algorithm (MMM) is used [16]. This gives an output related to $(A \times B)$ mod M via a scaling factor $R = r^s$ which is determined by the number of digits in input A. The form of interest here includes a final conditional subtraction which reduces the output to less than M, but causes variation in the time taken.

Montgomery's Modular Multiplication Algorithm (MMM)

```
P := 0 ;
For i := 0 to s-1 do
Begin
    P   := P + a_i×B ;
    q_i := (-p_0 m_0^-1) mod r ;
    P   := (P + q_i×M) div r ;
End ;
If P >= M then P := P-M
```

Output: $P = ABr^{-s}$ mod M for $A = \sum_{i=0}^{s-1} a_i r^i < M$ and $B < M$.

Here ${m_0}^{-1}$ under the $\bmod r$ is the unique residue modulo r with the property ${m_0}^{-1}\times m_0 \equiv 1 \bmod r$, i.e. the multiplicative inverse of $m_0 \bmod r$. Similarly, r^{-s} appearing under the mod M is the inverse of r^s modulo M. The digit products such as $a_i\times B$ are generated over s cycles by using the k-bit multiplier to compute each digit by digit product $a_i\times b_j$ for $0 \le j < s$ from least to most significant digit of B, propagating carries on the way so that a non-redundant representation can be used.

Using induction it is readily verified for $R = r^s$ that:

Theorem 1. [22] *The MMM loop has post-condition* $ABR^{-1} \le P < ABR^{-1}+M$.

With the advent of timing attacks [9], the conditional subtractions should be avoided. This is easy to achieve by having extra loop iterations [6, 19, 22]. Alternatively, a non-destructive subtraction can always be performed if space is available, and the correct answer selected using the sign of the result. However, EMR measurements might still reveal this choice.

4 The Timing Attack

Walter and Thompson [21] observed that the final, conditional subtraction takes place in Montgomery's algorithm with different frequencies for multiplications and squarings. Indeed, different exponent digits are also distinguished. It is assumed that the attacker can partition the traces of many exponentiations correctly into sub-traces corresponding to each execution of MMM and use their timing differences to determine each instance of this final subtraction. This gives him a matrix $Q = (q_{ij})$ in which q_{ij} is 1 or 0 according to whether or not there is an extra subtraction at the end of the ith modular multiplication of the jth exponentiation. We now estimate the distance between two rows of this matrix.

With the possible exception of the first one or two instances, it is reasonable to assume that the I/O for each MMM within an exponentiation is uniformly distributed over the interval $0..M{-}1$ since crypto-systems depend on multiplications performing what seem to be random mappings of multiplicands onto $0..M{-}1$. Suppose π_{mu} is the probability that the final subtraction takes place in MMM for two independent, uniformly distributed inputs. Let A, B and Z be independent, uniformly distributed, discrete random variables over the interval of integers $0..M{-}1$ which correspond to the MMM inputs and the variation in output within the bounds given in Theorem 1. Then $\pi_{mu} = pr(Z{+}ABR^{-1}{\ge}M)$ $= \frac{1}{M^3}\sum_{Z=0}^{M-1}\sum_{A=0}^{M-1}\sum_{B=0}^{M-1}(Z{+}ABR^{-1}{\ge}M) = \frac{1}{M^3}\sum_{A=0}^{M-1}\sum_{B=0}^{M-1}ABR^{-1}$. So $\pi_{mu} \approx \frac{1}{4}MR^{-1}$ because M is large.

On the other hand, suppose π_{sq} is the probability that the final subtraction takes place when MMM is used to square a uniformly distributed input. For A and Z as above, $\pi_{sq} = pr(Z{+}A^2R^{-1}{\ge}M) = \frac{1}{M^3}\sum_{Z=0}^{M-1}\sum_{A=0}^{M-1}(Z{+}A^2R^{-1}{\ge}M)$ $= \frac{1}{M^3}\sum_{A=0}^{M-1}A^2R^{-1}$, whence $\pi_{sq} \approx \frac{1}{3}MR^{-1}$.

The difference between π_{mu} and π_{sq} means that multiplications can be distinguished from squares if a sufficiently large sample of exponentiations is available. π_{mu} and π_{sq} are given approximately by averaging the entries in the rows of Q.

If the binary or "square-and-multiply" exponentiation method is used, then the pattern of squares and multiplies given by these row averages reveals the bits of the secret exponent D.

If the m-ary or sliding windows method is used for the exponentiation then it is also necessary to distinguish between multiplications corresponding to different exponent digits. This is done by using the fact that in the jth exponentiation, the same pre-computed multiplier $C_j^{(i)}$ is used whenever the exponent digit is i. Let π_{ij} be the probability that the MMM input $A = C_j^{(i)}$ induces the conditional subtraction when the argument B is uniformly distributed on $0..M{-}1$. For Z as before, $\pi_{ij} = pr(Z{+}C_j^{(i)}BR^{-1}{\geq}M) = \frac{1}{M^2}\sum_{Z=0}^{M-1}\sum_{B=0}^{M-1}(Z{+}C_j^{(i)}BR^{-1}{\geq}M)$ $= \frac{1}{M}\sum_{B=0}^{M-1}C_j^{(i)}BR^{-1} = \frac{1}{2}C_j^{(i)}R^{-1}$. The $C_j^{(i)}$ are uniformly distributed as j varies. So the average value of π_{ij} as j varies is, by definition, π_{mu}. Also, the average value of $\pi_{ij}{}^2$ as j varies is $\pi^{(2)} = \frac{1}{M}\sum_{C=0}^{M-1}\frac{1}{4}C^2R^{-2} \approx \frac{1}{12}M^2R^{-2}$.

The distance between two rows of Q is defined here as the average Hamming distance between corresponding entries. This is, in a sense, independent of the sample size N, i.e. the number of columns. Thus the expected distance between two rows which correspond to the *same* exponent digit i is $d_{ii} = \frac{2}{N}\sum_j \pi_{ij}(1{-}\pi_{ij})$. Its average value is therefore $\overline{d_{eq}} = \overline{d_{ii}} = 2(\pi_{mu}{-}\pi^{(2)}) \approx 2(\frac{1}{4}MR^{-1} - \frac{1}{12}M^2R^{-2}) = MR^{-1}(\frac{1}{2} - \frac{1}{6}MR^{-1})$, which is independent of N and, indeed, of i.

Now assume that the distributions of $C_j^{(i)}$ and $C_j^{(i')}$ are independent if $i \neq i'$. This is reasonable since the RSA crypto-system relies on the fact that application of a public exponent $E{=}3$ to any ciphertext C should randomly permute values modulo M. Then, if two rows of Q correspond to *distinct* digits i and i', their distance apart is approximately $d_{ii'} = N^{-1}(\sum_j \pi_{ij}(1{-}\pi_{i'j}) + \sum_j \pi_{i'j}(1{-}\pi_{ij}))$. The average value of this is $\overline{d_{neq}} = \overline{d_{ii'}} = 2(\pi_{mu}{-}\pi_{mu}{}^2) \approx MR^{-1}(\frac{1}{2}{-}\frac{1}{8}MR^{-1})$.

It is also possible to compare two squarings or a multiplication with a squaring. In an exponentiation, except perhaps for the first one or two squarings, the inputs to these would be independent. For a square and a multiplication involving exponent digit i, the expected distance between the rows of Q is $d_{sq,mu} = N^{-1}(\sum_j \pi_{ij}(1{-}\pi_{sq}) + \sum_j \pi_{sq}(1{-}\pi_{ij}))$. The average value of this is $\overline{d_{sq,mu}} = \pi_{mu}{+}\pi_{sq}{-}2\pi_{mu}\pi_{sq} = MR^{-1}(\frac{7}{12} - \frac{1}{6}MR^{-1})$. For two squares the expected distance between the (different) rows of Q is $d_{sq,sq} = 2N^{-1}\sum_j \pi_{sq}(1{-}\pi_{sq})$. The average value of this is $\overline{d_{sq,sq}} = 2\pi_{sq}(1{-}\pi_{sq}) = MR^{-1}(\frac{2}{3} - \frac{2}{9}MR^{-1})$.

Observe that $\overline{d_{eq}}$, $\overline{d_{neq}}$, $\overline{d_{sq,mu}}$ and $\overline{d_{sq,sq}}$ must all be distinct because $M < R$. As variance in these distances is proportional to $\frac{1}{N}$, the distance between two rows of Q will tend to one of these four distinct values as the sample size increases, making it easier to determine whether the rows represent, respectively, two multiplications corresponding to the same exponent digit, two multiplications corresponding to different exponent digits, a squaring and a multiplication, or two squarings.

It is easy to develop a suitable algorithm to traverse the rows of Q and classify all the multiplicative operations into subsets which represent either squarings or the same exponent digit. One such algorithm was given in [20]. The classification might, for example, minimise the sum of the differences between the expected and actual distances between all pairs of rows. The set in which a pre-multiplication lies determines the exponent digit associated with that set. There are a few consistency checks which might highlight any errors, such as enforcing exactly one pre-multiplication in each set of multiplications, and squarings having to appear only in multiples of $\log_2 m$ consecutive operations. This enables the secret exponent key D to be reconstructed with sufficiently few errors to enable its precise determination providing the sample size N is large enough.

5 Doubling the Key Length

Suppose the key length is increased. Does the timing attack become more or less successful when the ratio M/R, the base m and the sample size N are kept the same? We will assume that the detection rate for the conditional subtraction is unchanged because the detection method is unspecified. However, it seems likely that the subtractions will be easier to spot for longer keys since, although the same detectable operations are performed in both cases, there are more of them. The detection assumption means that, by counting only the subtractions in each row of Q, the same proportion of errors will be made in classifying an operation as a square or a multiply. Doubling n will then double the number of such errors. However, using *pairs* of rows rather than *single* rows for this classification improves the likelihood of classifying multiplications correctly.

First note that the distributions for the four types of distances between two rows are independent of n because the row length N is unchanged and the probability of a conditional subtraction is unchanged. Suppose the rows have already been roughly partitioned into one set for each non-zero exponent digit and one set for squares (m subsets in all). A row is classified, or its classification checked, by taking the distance between it and each of these sets. This distance is the average between the chosen row and each row of the group. Doubling n doubles the size of the group and so provides *twice* the number of distances from the row. So, as the other parameters are unchanged, the average distance from the row to the group will have *half* the variance when the key length is doubled. This will markedly reduce the probability of mis-classifying a row.

There are two main types of error to consider, namely those which are detectable through inconsistencies and those which are not. Inconsistencies arise when squares do not appear in sequences of m or multiplications are not separated by squares. For convenience, suppose that the inconsistent errors can be corrected with computationally feasible effort for both key lengths n and $2n$ because this is a minor part of the total cost. Hence it is assumed that a possible pattern of squares and multiplies has been obtained. For simplicity, we assume this is the correct pattern. Ambiguities also appear when a multiplication appears to be near to two or more different subsets of rows or near to none. The attacker then knows to check each of up to m possibilities for that multiplication. However, his main problem is the number of incorrect, but consistent decisions.

A mis-classified row has to be too far away from its correct subset *and* too close to one other set in order to be assigned to an incorrect exponent digit. For convenience, suppose that average distances from one row to the subset of rows representing a given exponent digit are normally distributed, and that appropriate scaling is performed so that the distances are $N(0,1)$. (Although the distances are bound within the interval $[0,1]$, the previous section shows their averages are not close to either end, and so increasing the sample size N will improve the match with a normal distribution.) Let Z be a random variable with such a distribution, and let δ be the (similarly scaled) distance at which the row is equally likely to be in the group as not in it. (A similar discussion is given in more detail in §7 for the other attack where these notions are made more precise.) Then the mis-classification occurs with probability $pr(Z{>}\delta)^2$. Since δ is inversely proportional to standard deviation it is dependent on the key length. Thus $\delta = \delta_n$ increases by a factor $\sqrt{2}$ when the key length is doubled.

Suppose p_n is the probability of correctly classifying one multiplication. Since keys with $2n$ bits require twice as many multiplications on average, we need $p_n < {p_{2n}}^2$ for the attack to become easier for the longer key. From the above, $p_n = 1{-}pr(Z{>}\delta_n)^2$ where Z is $N(0,1)$ and so the condition becomes $1{-}pr(Z{>}\delta)^2 < (1{-}pr(Z{>}\sqrt{2}\delta)^2)^2$. A quick glance at tables of the normal distribution shows that this is true providing $\delta > 0.616$. In other words, longer keys are easier to attack if distances between rows of Q are accurate enough. This just requires the sample size N to be large enough, or the experimental methods to be made accurate enough, or, indeed, n to be large enough. In conclusion, with all other aspects fixed there appears to be an optimal key length providing maximum security against this attack with shorter and longer keys being more unsafe.

However, with more leaked data per exponent bit as key length increases, it is not impossible that the attack may be developed further so that there is no longer an optimal secure key length and all increases in key length become unsafe. For example, some exponentiations are more helpful than others in discriminating between one exponent digit and any others because the associated multiplicands are unusually large or unusually small. These instances become apparent while performing the attack just described. Then a weighted Hamming distance which favours these cases should improve the correct detection of the corresponding exponent digit. Increasing key length provides more useful data for detecting such cases, further decreasing the strength of longer keys.

6 The Power Analysis Attack

The other attack considered here is one based on data-dependent power and/or EMR variation from the smartcard multiplier [20]. The long integer multiplication $A{\times}B$ requires every product of digits $a_u{\times}b_v$ to be computed. For each index u, the power or EMR traces from the multiplier are averaged as v ranges over its s values. In general, the digits of B are sufficiently random for this averaging process to provide a trace which is reasonably indicative of the Hamming weight of a_u. Concatenating these averaged traces for all a_u provides a single trace from which the vector of the Hamming weights of the digits of A is obtained with reasonable accuracy. Unless s is small, the Euclidean distance between the Hamming weight vectors for different, randomly chosen values of A has much smaller variance than its average. So this distance enables equal arguments A to be identified and unequal ones to be distinguished. By defining distance between power traces via the Hamming weight vectors in this way, the attacker can expect to distinguish between, or identify, the multipliers $C^{(i)}$ used in the modular multiplications of an exponentiation. This enables the exponent digits to be discovered and hence the secret key D to be determined.

In detail, each digit product $a_u{\times}b_v$ contributes $|a_u|{+}|b_v|{+}x$ to the trace-averaging process where $|d|$ is the Hamming weight of digit d and x is an instance of a random variable X which represents measurement errors and variation caused by the initial condition of the multiplier and other hardware. Without loss of generality, we assume $\mu_X{=}0$. Averaging over the s digits of B provides $\frac{1}{s}|B|{+}|a_u|{+}x_s$ as the uth co-ordinate of the vector for $A{\times}B$. Here the average for x_s is the same as for X (namely 0) but, with realistic independence assumptions, the variance is less by a factor s. As the s digits have k bits each and their Hamming weights are binomially distributed, this has a mean of $\frac{k}{2}{+}|a_u|$ and variance $\frac{k}{4s}+\frac{1}{s}\sigma_X^2$. Thus, overall, the coordinates have mean k and variance $\frac{k}{4}(1{+}\frac{1}{s}){+}\frac{1}{s}\sigma_X^2$. Now, comparing the vectors from two independent multipliers $C^{(i)}$ and $C^{(i')}$, the mean square difference in the uth co-ordinate is $\frac{k}{2}(1{+}\frac{1}{s}){+}\frac{2}{s}\sigma_X^2$, leading to a mean Euclidean distance between the vectors of $\sqrt{\frac{k}{2}(s{+}1){+}2\sigma_X^2}$. However, if the multipliers are equal, i.e. $i = i'$, then the Euclidean distance between the two vectors contains no contribution from $C^{(i)}$ and $C^{(i')}$. So its mean is derived entirely from the variance $\frac{k}{4s}+\frac{1}{s}\sigma_X^2$ in each co-ordinate, namely $\sqrt{\frac{k}{2}{+}2\sigma_X^2}$. Hence there is a $\sqrt{s{+}1}$-fold difference in size between the distances between vectors for the same and for different multiplicands when the data is "clean". Other variation from measurement error and hardware initialisation is only significant for small s, which is not the case here.

These vectors are used to form a matrix $Q = (q_{ij})$ similar to that in the timing attack: q_{ij} is the weight derived from the jth digit of the ith multiplication. As before, the distances (now Euclidean) between rows are used to distinguish squares from multiplies, and identify rows corresponding to the same exponent digits. Squares have no arguments in common with other operations, so that they have distances from all other rows which behave in the same way as described above for distances between multiplications for different exponent digits; they are not close to any other rows in the way that rows are for multiplications associated with the same exponent digit. Thus, as before, the attacker can potentially determine the secret key D.

7 Increasing the Key Length

The formulae in the previous section make explicit some of the results of increasing the key length n, and hence also the number of digits s. First, the trace averaging process is improved for individual entries in Q by reducing their variance. Secondly, as a consequence, the larger number of columns in Q increases the Euclidean distance between rows corresponding to different exponent digits without changing the distance between rows corresponding to the same digit. This enables each row to be classified more accurately. Thirdly, as in the timing attack, the larger number of rows in Q reduces the variance in the average distance of the row from the sets of rows which represent the various exponent digits. This, too, enables rows to be classified more accurately. So, as well as the increased difference in average separation, doubling the key length halves the variance in the separation since the sets for each exponent digit contain twice as many rows on average. At a theoretical level, this more than squares the probability of mis-classifying a digit so that the total number of digit errors actually *decreases* when key length increases. The question, as before, is whether such improved accuracy in classification really does achieve this in practice.

Modelling the attack by randomly generating Hamming weights is potentially inaccurate for several reasons. For example, bus encryption should hide Hamming weight effectively. Secondly, the multiplier does not necessarily yield the Hamming weight of inputs with much accuracy. Lastly, the multiplier does not operate independently on different input digits: processing one digit of a long integer input sets the multiplier in a biased state which affects the amount of switching when the next digit is processed.

So it was decided to assume the attacker made observations of EMR from, and power use by, the multiplier which would enable him to estimate the number of gates being switched in the multiplier. A model was built of the typical gate layout in a 2^k-bit multiplier using full adders and half adders in a Wallace tree without Booth re-coding. Random long integers were generated, and their digits fed sequentially into the multiplier as in a long integer multiplication. Gate switching activity was counted for each clock cycle, and the averaging and concatenation processes of the previous section were used to generate a row of the matrix Q. In this way m-ary exponentiation was modelled and a large number of values obtained for Q. Key length was varied to obtain an insight into general behaviour and confirm the improved feasibility of the attack as n increases.

Bit length n	32	64	128	256	512	1024	2048
Av btwn same	266	255	234	201	177	176	171
SD btwn same	191	161	137	129	106	110	100
Av min to diff	68.4	146	324	434	843	1453	2153
SD min to diff	53.4	87.9	78	102	140	131	118
%age errors	83	71	16	1.8	0.02	0.00	0.00
SDs btwn avs	–	–	0.84	2.02	5.41	10.6	18.2
p_c (lowr. bnd.)	–	–	0.439	0.711	0.9932	0.999...	0.999...

Table 1. Gate Switch Statistics for 32-bit Multiplier with $m = 8$.

Figures from the simulation of 8-ary exponentiation with a standard 32-bit multiplier and various key lengths are given in Table 1. This is the largest multiplier likely to be found in current smartcards. Standard m-ary exponentiation was used, not the sliding windows version, so that there were $m-1$ pre-computed powers for use in multiplications. These choices of k, m and algorithm make the correct association of exponent digits more difficult than is typically the case. So the setting here provides one of the most unfavourable scenarios for the attack, except for the absence of measurement noise. Moreover, the refinement of making comparisons between every pair of rows was neglected: just making comparisons of a row with each of the pre-computed multiplications was enough to establish the principle that longer key lengths may be less secure.

The column headings in Table 1 provide the key length: the number of bits n used in both the modulus M and the exponent D. Values are chosen to illustrate the effect of doubling key length. Although the smaller values are totally insecure, they allow a much clearer appreciation of the overall trends, especially in the last three rows of the table where values tend to their limits very quickly. The first line of data provides the average distance between vectors formed from multiplications which correspond to equal, exponent digits. These confirm the theory of the previous section: for equal digits, the average distance is essentially constant except for a slight decline arising from reduced noise as key length increases. The second line records the standard deviation in the figures of the first line. These also show a steady decrease as n increases. They are consistently about two thirds of the associated means over the given range.

The third and fourth lines provide the average distance, and its standard deviation, of a multiplication from the nearest pre-computed case which corresponds to a different exponent digit. If exponent digits are assigned to multiplications on the basis of the nearest pre-computation trace, these lines give a basis for estimating the number of errors that would be made. The percentage of such errors encountered in the simulations is given in the following line. For the very smallest cases, the nearest pre-computation vector is, on average, *nearer* than that corresponding to the same exponent digit. So a large number of errors are made. For the 128-bit or larger keys that are encountered in practice, the nearest pre-computation is usually the correct one. Due to its marginally different definition, the average tabulated in line 3 behaves slightly differently from the average between any pair of multiplications corresponding to different exponent digits. However, as with the formula in the previous section, this distance increases markedly with the key length, so that multiplications corresponding to different exponent digits are distinguished more easily. The standard deviations in line 4 varied noticeably between different samples generated by the multiplier simulation even for large samples (with the largest s.d. being around 50% greater than the smallest), but there seems to be a gradual trend upwards.

Both lines of standard deviations are included in order to calculate how many distances might be incorrectly classified as corresponding to equal or unequal digits. The average was taken of the two standard deviations in each column and the number of them which separate the two averages in the column was computed. This is tabulated in the second last line. The final line of the table contains an estimate from normal distribution tables for the probability p_c that the nearest trace of a pre-computation correctly determines the exponent digit associated with a multiplication, given that the operation is indeed a multiplication rather than a squaring. This assumes that the distances between traces sharing the same multiplicand are normally distributed with the tabulated expectation and variance and, similarly, that the minimum distance between one trace and a set of $m-2$ traces, all with different multiplicands, is normally distributed with the given expectation and variance.

Thus, let Z_c be a random variable with distribution $N(\mu_c, {\sigma_c}^2)$ which gives the distance to the correct trace, and let Z_d be a random variable with distribution $N(\mu_d, {\sigma_d}^2)$ which gives the distance to the nearest incorrect trace (one for a different digit). Then the probability of a correct decision is $p_c \approx Pr(Z_c < Z_d)$. Since, by the table, the means are so many standard deviations apart for typical values of n, a good approximation is given by

$$Pr(Z_c < Z_d) \;\approx\; Pr\left(Z_c < \tfrac{\sigma_c\mu_d+\sigma_d\mu_c}{\sigma_c+\sigma_d}\right) \times Pr\left(\tfrac{\sigma_c\mu_d+\sigma_d\mu_c}{\sigma_c+\sigma_d} < Z_d\right)$$

This yields $p_c \approx Pr(Z < \frac{\mu_d-\mu_c}{\sigma_d+\sigma_c})^2$ where Z is an $N(0,1)$ random variable. The last line of the table gives these values direct from tables of the normal distribution. This is approximately the probability of identifying the correct exponent digit given that the operation is a multiplication. It is consistent with the observed number of errors recorded in the table.

These probabilities goes up much faster than the square root as key length is doubled: $p_c^{(128)} = 0.4386$, $p_c^{(256)} = 0.7114$, $p_c^{(512)} = 0.9932$ and $p_c^{(1024)} = 1 - 2.5\times10^{-7}$ easily satisfy $p_c^{(128)} < \{p_c^{(256)}\}^2$, $p_c^{(256)} < \{p_c^{(512)}\}^2$ and $p_c^{(512)} < \{p_c^{(1024)}\}^2$. This means that it is easier to identify the exponent digits correctly for a pair of multiplications where the key has $2n$ bits than it is to identify the exponent digit correctly for a single multiplication where the key has only n bits. Thus fewer errors will be made for the $2n$-bit key. Indeed, the total number of predicted errors decreases rapidly towards zero over the range of the table. Thus, since squares are detected in a very similar way (they are not close to any of the pre-computed powers), at least in the simulation it becomes easier to deduce the full secret exponent as key length increases.

The analysis above has not taken into account comparisons between all possible pairs of product traces – distances between pairs of computation stage operations can be evaluated as well. As noted earlier, each row of Q can be compared with $O(n)$ other rows instead of just $O(1)$ rows. This decreases the variances involved and thereby improves the above decision process as n increases. Hence, as key length increases, a full scale attack will become even easier to process correctly than is indicated by the table.

8 A Particular Example

Consider the case from Table 1 which is closest to that in a typical current smartcard, namely a key length of 1024 bits. This will require about 1023 squares, which will occur in 341 triplets for $m = 2^3$, and about $7/8\times1023/3 \approx 298$ multiplications. By examining the exponent from left to right, of all the squares it is necessary to identify only the first of each triplet correctly. Hence there are approximately $341+298 = 639$ operations to identify as squares or multiplications, after which each multiplication must be associated with a digit value.

Let μ_{sq} and μ_{mu} be the average distances of a square and multiplication from the nearest of the $m-1$ pre-computation traces and let σ_{sq} and σ_{mu} be the corresponding standard deviations. A multiplication is assumed if, and only if, the distance is less than $\frac{\sigma_{mu}\mu_{sq}+\sigma_s\mu_{mu}}{\sigma_{mu}+\sigma_{sq}}$. The probability p_{sm} of this being the correct decision is then $Pr(Z < \frac{\mu_{sq}-\mu_{mu}}{\sigma_{sq}+\sigma_{mu}})$ for an $N(0,1)$ random variable Z. For larger m as here, $\mu_{mu} \approx \mu_c$ and $\mu_{sq} \approx \mu_d$ so that $p_{sm} \approx \sqrt{p_c}$. So all the squares and multiplications are identified correctly with probability at least ${p_{sq}}^{639} \approx {p_c}^{319.5}$. Correct determination of the exponent digits for the 298 or so multiplications is done with probability about ${p_c}^{298}$. Hence, without making any use of the considerable data given by comparing the $(1023{+}298)^2/2$ or so pairs of computation phase traces, deduction of the correct exponent will occur with probability at least about ${p_c}^{298+319.5} \approx (1-2.5\times10^{-7})^{617.5} \approx 0.9998$.

At least theoretically, this very high probability should give cause for concern. In practice, it is to be hoped that measurement noise will substantially reduce the ability to identify the correct multiplicand $C^{(i)}$. Even a modest reduction in the probabilities p_{mu} and p_{sq} would be helpful since both must be raised to a power linear in the number of bits in the key in order to obtain the probability that the key is identified correctly first time without further computing to try the next best alternatives. It is computationally feasible to correct only one or two errors.

9 Counter-measures

Counter-measures for the timing attack are straightforward, and were covered in the introduction: the exponent can be randomly blinded [9] and the subtraction can either be performed every time or be omitted every time [6, 19, 22].

The power analysis attack is harder to perform, but also harder to defeat. Exponent blinding is not a defence. The attack does rely on re-use of the pre-computed powers. Hence performing m-ary exponentiation in the opposite order, namely from least to most significant digit, may be a solution. This can be done without significant extra computation time [15, 23]. For $m = 2$, the normal square-and-multiply algorithm can be modified to square-and-always-multiply, but this is more expensive time-wise.

Apart from hardware counter-measures such as a Faraday cage to shield the processor and capacitors to smooth out the power trace, a larger multiplier also helps. This reduces the number of digits s over which averages are taken and reduces the number s of concatenated traces. Thus it reverses the effect of increased key length on the traces. Moreover, with larger numbers of words sharing the same Hamming weight, it becomes less easy to use the Euclidean metric to separate the different multiplicands. Further, one might use two multipliers in parallel. Montgomery's modular multiplication algorithm naturally uses two.

Then the power used by one might successfully shield observation of the power used by the other. Thus safety can be bought, but perhaps only at a price.

10 Conclusion

Two attacks on smartcard implementations of RSA have been outlined, one a timing attack and the other a power analysis attack. In each case the effect of increasing the key length was studied for its impact on the number of bit errors made in deducing the secret key. For the timing attack, leaked data is proportional to the square of the key length and it appears that there is an optimal secure length with both shorter and longer keys being less safe. For the power analysis attack, leaked data is proportional to the cube of the key length and the analysis shows that longer keys are less safe.

There are a number of both algorithmic and hardware counter-measures which improve resistance against such side channel attacks and they should provide the extra safety that one has been traditionally led to expect from longer key lengths. However, the main conclusion is that counter-measures to cryptanalysis must not be assumed to be suitable as counter-measures to side channel leakage. In particular, increasing key length on its own appears to be quite unsuitable as a counter-measure in embedded RSA cryptosystems.

References

1. R. M. Best, *Crypto Microprocessor that Executes Enciphered Programs*, U.S. Patent 4,465,901, 14 Aug. 1984.
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), Lecture Notes in Computer Science, **1820**, Springer-Verlag, 2000, pp. 175–190.
3. W. Diffie & M. E. Hellman, *New Directions in Cryptography*, IEEE Trans. Info. Theory, **IT-22**, no. 6, 1976, pp. 644–654.
4. T. El-Gamal, *A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Trans. Info. Theory, **IT-31**, no. 4, 1985, pp. 469–472.
5. K. Gandolfi, C. Mourtel & F. Olivier, *Electromagnetic Analysis: Concrete Results*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), Lecture Notes in Computer Science, **2162**, Springer-Verlag, 2001, pp. 251–261.
6. G. Hachez & J.-J. Quisquater, *Montgomery exponentiation with no final subtractions: improved results*, Cryptographic Hardware and Embedded Systems (Proc CHES 2000), C. Paar & Ç. Koç (editors), Lecture Notes in Computer Science, **1965**, Springer-Verlag, 2000, pp. 293–301.
7. D. E. Knuth, The Art of Computer Programming, vol. 2, *Seminumerical Algorithms*, 2nd Edition, Addison-Wesley, 1981, pp. 441–466.
8. Ç. K. Koç, *Analysis of Sliding Window Techniques for Exponentiation*, Computers and Mathematics with Applications, **30**, no. 10, 1995, pp. 17–24.

9. P. Kocher, *Timing attack on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Proc. Crypto 96, N. Koblitz (editor), Lecture Notes in Computer Science, **1109**, Springer-Verlag, 1996, pp. 104–113.
10. P. Kocher, J. Jaffe & B. Jun, *Differential Power Analysis*, Advances in Cryptology – Crypto '99, M. Wiener (editor), Lecture Notes in Computer Science, **1666**, Springer-Verlag, 1999, pp. 388–397.
11. M. G. Kuhn, *Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP*, IEEE Transactions on Computers, **47**, No. 10, October 1998, pp. 1153–1157.
12. R. Mayer-Sommer, *Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards*, Cryptographic Hardware and Embedded Systems (Proc CHES 2000), C. Paar & Ç. Koç (editors), Lecture Notes in Computer Science, **1965**, Springer-Verlag, 2000, pp. 78–92.
13. T. S. Messerges, E. A. Dabbish & R. H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*, Cryptographic Hardware and Embedded Systems (Proc CHES 99), C. Paar & Ç. Koç (editors), Lecture Notes in Computer Science, **1717**, Springer-Verlag, 1999, pp. 144–157.
14. V. Miller, *Use of Elliptic Curves in Cryptography*, Proc. CRYPTO '85, H. C. Williams (editor), Lecture Notes in Computer Science, **218**, Springer-Verlag, 1986, pp. 417–426.
15. Bodo Möller, *Parallelizable Elliptic Curve Point Multiplication Method with Resistance against Side Channel Attacks*, Information Security – ISC 2002, A. H. Chan & V. Gligor (editors), Lecture Notes in Computer Science, **2433**, Springer-Verlag, 2002, pp. 402–413.
16. P. L. Montgomery, *Modular Multiplication without Trial Division*, Mathematics of Computation, **44**, no. 170, 1985, pp. 519–521.
17. R. L. Rivest, A. Shamir & L. Adleman, *A Method for obtaining Digital Signatures and Public-Key Cryptosystems*, Comm. ACM, **21**, 1978, pp. 120–126.
18. W. Schindler, *A Timing Attack against RSA with Chinese Remainder Theorem*, Cryptographic Hardware and Embedded Systems (Proc CHES 2000), C. Paar & Ç. Koç (editors), Lecture Notes in Computer Science, **1965**, Springer-Verlag, 2000, pp. 109–124.
19. C. D. Walter, *Montgomery Exponentiation Needs No Final Subtractions*, Electronics Letters, **35**, no. 21, October 1999, pp. 1831–1832.
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), Lecture Notes in Computer Science, **2162**, Springer-Verlag, 2001, pp. 286–299.
21. C. D. Walter & S. Thompson, *Distinguishing Exponent Digits by Observing Modular Subtractions*, Topics in Cryptology – CT-RSA 2001, D. Naccache (editor), Lecture Notes in Computer Science, **2020**, Springer-Verlag, 2001, pp. 192–207.
22. C. D. Walter, *Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli*, Topics in Cryptology – CT-RSA 2002, B. Preneel (editor), Lecture Notes in Computer Science, **2271**, Springer-Verlag, 2002, pp. 30–39.
23. C. D. Walter, *MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis*, Topics in Cryptology – CT-RSA 2002, B. Preneel (editor), Lecture Notes in Computer Science, **2271**, Springer-Verlag, 2002, pp. 53–66.
24. *Digital Signature Standard* (DSS), FIPS 186, http://csrc.nist.gov/publications/, US National Institute of Standards and Technology, May 1994.

25. *Data Encryption Standard* (DES), FIPS 46-3, http://csrc.nist.gov/publications/, US National Institute of Standards and Technology, October 1999.
26. *Advanced Encryption Standard* (AES), FIPS 197, http://csrc.nist.gov/publications/, US National Institute of Standards and Technology, November 2001.
27. *Index of National Security Telecommunications Information Systems Security Issuances*, NSTISSC Secretariat, US National Security Agency, 9 January 1998.