Optimal Recovery of Secret Keys from Weak Side Channel Traces

Werner Schindler[1] and Colin D. Walter[2]

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI)
Godesberger Allee 185-189, 53175 Bonn, Germany
Werner.Schindler@bsi.bund.de
[2] Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom
Colin.Walter@rhul.ac.uk

Abstract. It should be difficult to extract secret keys using weak side channel leakage from embedded crypto-systems which employ standard counter-measures. Here we consider the case of key re-use with randomised exponent recoding. An optimum strategy is presented and proved, but it has the disadvantage of impracticality for realistic key sizes. Developed from the basis of an optimal decision strategy, some modified, computationally feasible versions are studied for effectiveness. This shows how to modify existing algorithms and pick their parameters for the best results.

Key Words. Side channel leakage, power analysis, optimal strategy.

1 Introduction

The academic parents of this work are the optimal strategies of Schindler [10] and the algorithms of Walter [13] for deducing secret keys using weak side channel leakage from an exponentially based public key crypto-system in which the key is re-used a number of times in some form. The optimal decision strategy is practically feasible if the key can be guessed in small parts as in [10], for instance, while under the present conditions it remains feasible only for artificially small key sizes. However, the algorithms discussed in [13] still remain feasible for typical cryptographic key sizes. Here a comparison of the two approaches leads to i) the determination of the best parameters to choose in existing computationally feasible algorithms and ii) the identification of points in the development of the algorithm where it seems impossible to derive a computationally feasible method from the optimal algorithm. Although standard hardware countermeasures nowadays make side channels extremely weak in embedded systems, the methods here were used in simulations to recover keys using substantially weaker leakage than has been reported in the past.

M. Parker (Ed): IMACC 2009, LNCS **5921**, pp. 446–468, 2009.

The earliest published work on such secret key recovery is that of Kocher *et al.* [6, 7]. For RSA and similar crypto-systems, (unprotected) classical exponentiation algorithms employ the same sequence of multiplicative operations every time the key is re-used, and this allows leakage (in a certain sense) to be averaged over many traces to guess the key in small portions, such as bit by bit. With enough data, the operation types can be determined as squarings or multiplications, and this is sufficient to yield the key for the binary exponentiation algorithm. Randomised recoding of the exponent causes operations corresponding to the same key bit to be mis-aligned and variable for different uses of the key. A number of such re-codings exist [9, 8, 4, 15] and they were generally believed to lead to much more secure systems – attacks on such re-codings seemed to require significantly stronger side channel leakage to succeed than is the case where Kocher's attack applies. However, this no longer seems to be the case. Karlof & Wagner [5], Green *et al.* [3] and Walter [13] provide increasingly robust details for attacking such systems without using such strong leakage, thereby emphasising the need to combine a number of counter-measures rather than relying on just one or two to defeat the opponent. None of these works provides justification for the efficacy of their algorithms. Thus there is a gap between what has been achieved at a computationally feasible level and what has been derived theoretically. Here we describe a search to bridge that gap by explaining the computationally efficient choices in terms of the optimal strategies described by Schindler in [10]. As a result, much more powerful means of exploiting the leakage are now identified.

2 The Leakage Model

The context of the side channel attack is the repeated use of a randomised exponentiation algorithm for computing C^K in any cryptographic group where K is a fixed secret key which is not blinded by a random multiple of the group order[1], and C is an unknown ciphertext (or unknown plaintext) which may vary and may be blinded[2]. Of course, all the following considerations apply equally well to randomised scalar multiplications in additive groups (as in ECC).

The adversary is assumed to know all the details of the exponentiation algorithm. Use of the key provides him with a side channel trace for the exponentiation itself, but no further information is assumed: in particular, he is not expected to be able to choose or see any direct input to the exponentiation, nor view any output, nor usefully observe any pre- or post-processing.

It is assumed that occurrences of multiplicative operations in the exponentiation can be identified accurately from the corresponding side channel trace, but that their identities as squares or multiplications (and, in the case of methods with pre-computed tables, multiplications by particular table entries) can only be determined with a substantial degree of inaccuracy [1]. The adversary's aim

[1] Another standard counter-measure to Kocher's averaging of side channel traces.

[2] The base in the exponentiation is frequently unknown due, for example, to "Rivest" blinding [2] or because of an unknown modular reduction when applying the CRT.

is to discover K using computationally feasible resources. The multiplicative operations are assigned probabilities that they represent squares or multiplications as a result of previous experience by the adversary. For this he uses knowledge of the stochastic behaviour of the operations in the side channel, and the extent to which this behaviour varies.

In order to model noisy measurement data in the simulations we assumed that these probabilities were distributed binomially and independently for all multiplications with mean probabilities depending on the true type of the operation, so that some were known correctly with high probability, but most were known with little confidence. However, Theorem 1 also covers more general leakage scenarios.

3 The Randomised Exponentiation

Examples of the randomised exponentiation algorithms which can be attacked in the way described here include those of Liardet-Smart [8], Oswald-Aigner [9] and Ha-Moon [4, 15]. Their common, underlying basis is a recoding of the binary representation of the key K into a form

$$R = ((...(r_{m-1}2^{m_{m-2}} + r_{m-2})2^{m_{m-3}} + ... + r_2)2^{m_1} + r_1)2^{m_0} + r_0$$

for digits r_i and exponents m_i in some fixed, pre-determined sets $\mathcal{D} \subseteq \mathbb{Z}$ (which, for convenience, contains 0[3]) and $\mathcal{M} \subseteq \mathbb{N}^+$ respectively. In this *recoding*, both r_i and m_i are selected according to some finite state automaton (FA) which has the bits of K and the output from a random number generator (RNG) as inputs. For convenience, we assume the bits of K are consumed by the FA from least to most significant. Different bit streams from the RNG result in different recodings R of K.

The exponentiation C^R begins with the pre-calculation of the table $\{C^d \mid d \in \mathcal{D}, d \neq 0\}$. Then for $i = m-2, m-3, ...2, 1, 0$ the main iterative step of the exponentiation consists of m_i squarings followed by a multiplication by the table entry C^{r_i} when $r_i \neq 0$. This results in a sequence of multiplicative operations which is most easily presented using r_i to denote multiplication by C^{r_i} and m_i copies of 0 to denote the m_i squarings. We call this the *recoding sequence* for R, and it belongs to $\mathcal{D}^*$. For example, the exponent $K = 13_{10} = 1101_2$ may have a recoding $R = (1.2^2+3)2^1+\bar{1}$ which gives the operation sequence 100301, the recoding sequence associated with the recoding R. For convenience (e.g. in Section 7), the leading recoded digit is translated in the same way as the others into multiplicative operations of the recoding sequence even when that digit is 0; alternatives in processing the leading bits are ignored. (This matches the situation where the exponentiation algorithm begins with 1 instead of C.) However, the leading digits are invariably treated differently in practice, and appropriate modifications need to be made in the methods here to handle them properly.

[3] Recodings which do not allow 0 in representations are not excluded here, but we want to include it for another use, namely to represent a squaring operation.

Thus, we are using the same set $\mathcal{D}$ to represent both the set of recoding digits and the set of corresponding recoding operations. In our examples, we will make the distinction clearer by using, for example, 'S' for the squaring operation and reserving '0' for the digit. We will rarely be working with recodings. It will be much more frequently be with recoding sequences.

The exponentiation algorithms of interest are assumed to have the property that perfect knowledge of the multiplication/squaring sequences (without necessarily knowledge of the choice of respective table entries in the case of multiplications) for a small number of recodings R of K yields enough information to reconstruct the secret key K with at most a small number of ambiguities. This is the case for the algorithms listed above: attacks on them using such information are described in [11], [12] and [15] respectively. The theory here, however, allows for the possibility of distinguishing between the use of different table entries in the multiplications, and this enables one to deal with recodings for which only non-zero digits are used.

4 The Optimal Decision Strategy

The optimal decision strategy for discovering the secret key K begins by identifying a key K^* with the highest probability of having generated the observed side channel leakage. If the most likely key candidate fails the attacker tries the key candidates that are ranked $2, 3, \ldots$. This maximises the use of known information about K and hence minimises the effort in searching for K.

For each use of the secret key K, the side channel leakage leads to a best guess G at the recoding that was used: the locations of multiplicative operations are identified, and the most likely digit values are selected for those operations. Associated with a set of these recoding guesses there are optimal choices K^* for the key value – those which maximise how well the key collectively matches the guessed recodings.

Remark 1. To be successful in real-world attacks we must assume there is enough leakage to ensure that the correct key is among the best, i.e. most probable, fits to the recoding guesses for otherwise it will be computationally infeasible to find it. If not correct, the most plausible keys typically are at least related to the correct one, which means that their bit representations or their recodings are similar to that of the correct key in some sense. E.g., long sequences of bits in these most probable keys may either be identical to those in the correct key or, depending on the recoding scheme, related to them in very specific, predictable ways[4]. Hence the errors in using a most probable key to predict the correct key should also be relatively few in number, generally isolated, and effectively independent. Consequently, virtually all errors will be equally easy to correct although finding them may not be so easy.

[4] E.g. two keys which are bit-wise complements of each other can have almost identical recodings, as may two keys of which one is almost the same as a shift of the other.

Definition 1.
(i) Let $\mathcal{K} \subseteq \mathbb{F}_2{}^*$ denote the set of all admissible keys in binary representation and $\mathcal{R} \subseteq \mathcal{D}^*$ the set of all possible recoding sequences of keys for the chosen recoding scheme. Recoding to an operation sequence is defined by a map $\phi\colon \mathcal{K}\times\mathcal{Z} \to \mathcal{R}$ where $z \in \mathcal{Z}$ denotes a random number in a finite set $\mathcal{Z}$. The set $\phi(K, \mathcal{Z}) \subseteq \mathcal{R}$ of possible recoding sequences of K is denoted $\mathcal{R}(K)$ or $\mathcal{R}_K$.
(ii) The set of possible recoding sequence guesses which can be deduced from side channel leakage is denoted by $\mathcal{G}$ where $\mathcal{R} \subseteq \mathcal{G} \subseteq \mathcal{D}^*$, and $\mathrm{len}(G)$ is the number of elements in a guess $G \in \mathcal{G}$ when viewed as a sequence over $\mathcal{D}$. The i^{th} element of G is g_i where the index runs from $\mathrm{len}(G)-1$ (the most significant operation) down to 0 (the least significant operation).
(iii) The random variable $X_{\mathcal{Q}}$ assumes values in the set $\mathcal{Q}$.

Clearly the subsets $\mathcal{R}(K)$ are disjoint for different K's – each recoding sequence specifies exponentiation to a particular power, and that power is K. We seek to determine $K \in \mathcal{K}$ from guesses $G_1, \ldots, G_N \in \mathcal{G}$ based on side channel leakage with the disadvantage that the G_j are probably inconsistent because of erroneous operation deductions, i.e. they may not all represent the same key K; some G_j may even represent 'impossible' recoding sequences (i.e. not belonging to any K).

Starting with key $K \in \mathcal{K}$ a guessed recoding sequence G may be interpreted as the result of two consecutive random experiments. The first step (recoding to an operation sequence) is determined by a random number z (and K, of course). The second step (adding noise), namely *recoded operation sequence* $\to$ *recoding guess*, is determined by a hidden parameter $y \in \mathcal{Y}$ which represents the influence of noise of various forms such as that arising from the measurement process itself or from implemented countermeasures. Formally, the guessing step $R \mapsto G$ can be expressed by a function $\psi\colon \mathcal{R}\times\mathcal{Y} \to \mathcal{G}$, $(R, y) \mapsto \psi(R, y)$ since the guessed recoding sequence G depends on the actual recoding sequence $R = \phi(K, z) \in \mathcal{R}$ but not directly on $K \in \mathcal{K}$ or z.

The random variable $X_{\mathcal{K}}$ describes the selection of the key K during initialisation of the attacked cryptosystem. Without loss of generality we may assume the probability $\eta(K)$ of each key is non-zero:

$$\eta(K) \stackrel{\text{def}}{=} \mathrm{Prob}(X_{\mathcal{K}} = K) > 0 \qquad \text{for all } K \in \mathcal{K}. \tag{1}$$

The distributions of $X_{\mathcal{K}}$ and $X_{\mathcal{Z}}$ and, of course, the applied recoding scheme determine the distribution ν of the random variable $X_{\mathcal{R}} \stackrel{\text{def}}{=} \phi(X_{\mathcal{K}}, X_{\mathcal{Z}})$. The random variable $X_{\mathcal{G}}$ quantifies the distribution of random recoding sequences that are guessed by the attacker. Theorem 1 considers the situation where an attacker observes N re-uses of the key K. We assume that the associated random variables $X_{\mathcal{K}}, X_{\mathcal{Z},1}, \ldots, X_{\mathcal{Z},N}, X_{\mathcal{Y},1}, \ldots, X_{\mathcal{Y},N}$ are independent.

Theorem 1. *(i) Given recoding sequence guesses $G_1, \ldots, G_N \in \mathcal{G}$, the optimal decision strategy $\tau_* : \mathcal{G}^N \to \mathcal{K}$ selects a key $K^* \in \mathcal{K}$ that maximises the expres-*

sion

$$\sum_{j=1}^{N} \log \left(\sum_{R \in \mathcal{R}(K)} \nu(R) \mathrm{Prob}(X_{\mathcal{G},j}{=}G_j \mid X_{\mathcal{R}}{=}R) \right) - (N{-}1) \log\left(\eta(K)\right) \;=\; (2)$$

$$\sum_{j=1}^{N} \log \left(\sum_{R \in \mathcal{R}(K)} \nu(R \mid X_{\mathcal{K}}{=}K) \mathrm{Prob}(X_{\mathcal{G},j}{=}G_j \mid X_{\mathcal{R}}{=}R) \right) + \log\left(\eta(K)\right). \quad (3)$$

If this maximum is attained for several keys the first key is chosen under any pre-selected order on $\mathcal{K}$.

(ii) Assume that the adversary is able to detect whenever an operation of the recoded sequence R *is carried out and guesses the types of these operations independently to obtain* $G \in \mathcal{G}$. *Assume also that the conditional probabilities* $p(g|r) \stackrel{\mathrm{def}}{=} \mathrm{Prob}($*guessed op*^{n} *type is* g *given the true op*^{n} *type is* $r)$ *for guessed operations* g *do not depend on the position of the operation* r *but only on the operation types* g, $r \in \mathcal{D}$. *Then* $\mathrm{len}(R) = \mathrm{len}(G)$, *and the optimal decision* K^* *with respect to the* N *such guesses* $G_1, \ldots, G_N$ *maximises*

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \mathrm{len}(R)=\mathrm{len}(G_j)}} \nu(R) \prod_{i\,=\,0}^{\mathrm{len}(G_j)-1} p(g_{j,i} \mid r_i) \right) - (N{-}1) \log\left(\eta(K)\right) = \quad (4)$$

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \mathrm{len}(R)=\mathrm{len}(G_j)}} \nu(R \mid X_{\mathcal{K}}{=}K) \prod_{i\,=\,0}^{\mathrm{len}(G_j)-1} p(g_{j,i} \mid r_i) \right) + \log\left(\eta(K)\right). \quad (5)$$

(iii) Assume that $\mathcal{D} = \{0, 1, \overline{1}\}$. *Suppose also that (ii) holds with* $p(0|0) = 1$, $p(1|\overline{1}) = p(\overline{1}|1) = q \in [0, 0.5]$ *and* $p(0|1) = p(0|\overline{1}) = 0$. *Then the optimal decision* K^* *maximises*

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \mathrm{len}(R)=\mathrm{len}(G_j), \\ \mathrm{supp}_0(R)=\mathrm{supp}_0(G_j)}} \nu(R) \left(\frac{q}{1-q}\right)^{\mathrm{Ham}(R,G_j)} \right) - (N{-}1) \log\left(\eta(K)\right) = \quad (6)$$

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \mathrm{len}(R)=\mathrm{len}(G_j), \\ \mathrm{supp}_0(R)=\mathrm{supp}_0(G_j)}} \nu(R \mid X_{\mathcal{K}}{=}K) \left(\frac{q}{1-q}\right)^{\mathrm{Ham}(R,G_j)} \right) + \log\left(\eta(K)\right) \quad (7)$$

where $\mathrm{supp}_0(R)$ *and* $\mathrm{supp}_0(G_j)$ *are the sets of positions in* $R = (r_{\mathrm{len}(R)-1}, \ldots, r_0)$ *and* $G_j = (g_{j,\mathrm{len}(R)-1}, \ldots, g_{j,0})$ *for which the type of operation is a squaring, i.e.* $r_i = 0$. *Further,* $\mathrm{Ham}(R, G_j)$ *denotes Hamming distance, viz. the number of positions for which the operations in* R *and* G_j *differ.*

(iv) If $X_{\mathcal{K}}$ is uniformly distributed on $\mathcal{K}$ the terms "$(N-1)\log(\eta(K))$" in (2), (4), (6) and "$\log(\eta(K))$" in (3), (5), (7) may be omitted.

(v) For any constant $c > 0$, multiplying the probability $\nu(R)$ in (2), (4) or (6) (resp. the conditional probability $\nu(R \mid X_{\mathcal{K}} = K)$ in (3), (5) or (7)) by c for all $R \in \mathcal{R}$ does not change the optimal decision strategy in (i), (ii), (iii) or (iv), respectively, but may simplify concrete calculations.

Proof. To prove (2) we apply Theorem 3(i) from the Appendix (originally proved as Theorem 1(i) in [10]) with $\Theta = \mathcal{K}$, $\Omega = \mathcal{G}^N$, $\mu =$ the counting measure on $\mathcal{G}^N$, and the loss function $s(\theta, a) \stackrel{\text{def}}{=} 0$ if $\theta = a$ and $s(\theta, a) \stackrel{\text{def}}{=} 1$ otherwise since, as noted at the start of this section, any kind of false key guess is equally harmful.

Let $\text{Prob}_K(A)$ be the probability of the event A if K is the correct key. Then the optimal decision strategy $\tau^*\colon \mathcal{G}^N \to \mathcal{K}$ assigns to the guess tuple $\omega = (G_1, \ldots, G_N)$ a key $K^* \in \mathcal{K}$ which minimises

$$\sum_{K' \in \mathcal{K}} s(K', K^*)\eta(K')\text{Prob}_{K'}((X_{\mathcal{G},1}, \ldots, X_{\mathcal{G},N}) = \omega) \quad =$$
$$\sum_{K' \in \mathcal{K}} \eta(K')\text{Prob}_{K'}((X_{\mathcal{G},1}, \ldots, X_{\mathcal{G},N}) = \omega) - \eta(K^*)\text{Prob}_{K^*}((X_{\mathcal{G},1}, \ldots, X_{\mathcal{G},N}) = \omega)$$

or, equivalently, maximises $\eta(K^*)\text{Prob}_{K^*}((X_{\mathcal{G},1}, \ldots, X_{\mathcal{G},N}) = \omega)$. Recall that $X_{\mathcal{K}}$, $X_{\mathcal{Z},1}, \ldots, X_{\mathcal{Z},N}, Y_{\mathcal{Z},1}, \ldots, Y_{\mathcal{Z},N}$ are independent and $X_{\mathcal{G},j} = \psi(\phi(X_{\mathcal{K}}, X_{\mathcal{Z},j}), X_{\mathcal{Y},j})$. For each fixed $K \in \mathcal{K}$ and $\omega = (G_1, \ldots, G_N) \in \mathcal{G}^N$ we have

$$\begin{aligned}
\text{Prob}_K((X_{\mathcal{G},1}, \ldots, X_{\mathcal{G},N}) = \omega) \quad &= \quad \prod_{j=1}^{N} \text{Prob}_K(X_{\mathcal{G},j} = G_j) \\
&= \quad \prod_{j=1}^{N} \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{Z},j} = z_j)\ \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{Z},j} = z_j, X_{\mathcal{K}} = K) \\
&= \quad \prod_{j=1}^{N} \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{Z},j} = z_j)\ \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = \phi(K, z_j)) \\
&= \eta(K)^{-N} \prod_{j=1}^{N} \sum_{z_j \in \mathcal{Z}} \text{Prob}(X_{\mathcal{K}}{=}K, X_{\mathcal{Z},j}{=}z_j)\ \text{Prob}(X_{\mathcal{G},j}{=}G_j \mid X_{\mathcal{R}}{=}\phi(K, z_j)) \\
&= \eta(K)^{-N} \prod_{j=1}^{N} \sum_{R \in \mathcal{R}(K)} \nu(R)\ \text{Prob}(X_{\mathcal{G},j} = G_j \mid X_{\mathcal{R}} = R)
\end{aligned}$$

since $\phi(K, \mathcal{Z}) = \mathcal{R}(K)$ and the sets $\mathcal{R}(K')$ are mutually disjoint for different keys K'. As the logarithm function is strictly increasing, the last formula implies (2). Moreover, since the sets $\mathcal{R}(K')$ are mutually disjoint, for any $R \in \mathcal{R}(K)$ we have $\nu(R \mid X_{\mathcal{K}} = K) = \nu(R)/\eta(K)$, which proves (3). Substituting the additional conditions from (ii) into (2) and (3) yields (4) and (5).

Assertion (iii) follows from (ii) since, when $\mathrm{supp}_0(R) = \mathrm{supp}_0(G_j)$,

$$\begin{aligned}\prod_{i\,=\,0}^{\mathrm{len}(G_j)-1} p(g_{j,i}, r_i) &= q^{\mathrm{Ham}(R,G_j)}(1-q)^{\mathrm{len}(G_j)-|\mathrm{supp}_0(G_j)|-\mathrm{Ham}(R,G_j)} \\ &= \left(\frac{q}{1-q}\right)^{\mathrm{Ham}(R,G_j)} (1-q)^{\mathrm{len}(G_j)-|\mathrm{supp}_0(G_j)|}\ .\end{aligned}$$

Otherwise the product is zero. Since the last factor only depends on G_j (fixed) but not on R it can be factored out in (4) and (5). In particular, it has no impact on the location of the maximum, and thus may be dropped. Similarly, in (iv) $\eta(K)$ is assumed to be the same for all $K \in \mathcal{K}$ and thus may be dropped. Assertion (v) follows immediately from the functional property $\log(ab) = \log(a) + \log(b)$ of the logarithm function. ∎

Remark 2. (i) Theorem 1(i) is very general. It covers any recoding scheme as long as the random variables $X_{\mathcal{K}}, X_{\mathcal{Z},1}, \ldots, X_{\mathcal{Z},N}, X_{\mathcal{Y},1}, \ldots, X_{\mathcal{Y},N}$ are independent. In particular, there are no restrictions on maps $\phi\colon \mathcal{K} \times \mathcal{Z} \to \mathcal{R}$ or $\psi\colon \mathcal{R} \times \mathcal{Y} \to \mathcal{G}$. (ii) In the case of several best candidates in Theorem 1 (which should be a rare event in practice) the rule that one of them is selected according to some predefined order has 'technical reasons', namely, to ensure the measurability of the decision strategy. Alternatively, one could pick one of the best key candidates at random (defining a randomised decision strategy).

Definition 2. *With regard to expression (2), define the (weighted) "credibility" function* cred: $\mathcal{G} \times \mathcal{K} \to R$ *by*

$$\mathrm{cred}(G, K) \stackrel{\mathrm{def}}{=} \sum_{R \in \mathcal{R}(K)} \nu(R)\mathrm{Prob}(X_{\mathcal{G}} = G \mid X_{\mathcal{R}} = R). \tag{8}$$

A large value for this credibility function implies that G is a likely recoding sequence guess for key K.

Corollary 1. *If η is uniformly distributed on $\mathcal{K}$ (i.e. if all keys are equally likely) for recoding sequence guesses $G_1, \ldots, G_N$ the optimal decision strategy selects a key $K^* \in \mathcal{K}$ that maximises $\sum_{j=1}^{N} \log(\mathrm{cred}(G_j, K))$ for $K \in \mathcal{K}$.*

As in theorem Theorem 1(ii), suppose that $\mathrm{Prob}(X_{\mathcal{G}} = G \mid X_{\mathcal{R}} = R) = \prod_{0 \le i < \mathrm{len}(G)} p(g_i | r_i)$. Then the computation of $\mathrm{cred}(G, K)$ is computationally feasible because recodings are performed by a finite automaton (FA) which typically has very few states. The state of the finite automaton incorporates the difference between the key suffix which has been read and the recoded key suffix which has been generated. The finite automaton reads the next key bit and uses a random input to decide the next recoded digit to output and which transition to make to its next state. For each state s of the automaton, define

$$\mathrm{cred}(G, K', s) \stackrel{\mathrm{def}}{=} \sum_{R \in \mathcal{R}(K')} \nu(R, s)\mathrm{Prob}(X_{\mathcal{G}}{=}G | X_{\mathcal{R}}{=}R) \tag{9}$$

where $\nu(R, s)$ corresponds to the FA reaching state s after generating the recoding R of key suffix K'. All the component functions can be evaluated iteratively by processing the bits of K sequentially (here from right to left): $\nu(R, s)$ is given by the product of the state transition probabilities for the path through the FA to s which yields R, and, if $K'' = d||K'$ has leading (i.e. most significant) bit d, then $\mathrm{cred}(G, K'', s')$ can be expressed easily as a linear combination of values $\mathrm{cred}(G, K', s)$ for the suffix K'' of K. The work is then proportional to the length of K and the number of FA states. Algorithm 1 (in the Appendix) provides a concrete example for the Ha-Moon recoding scheme.

5 Traces

The rest of this paper investigates the extent to which the main theorem (Theorem 1) applies to real-world scenarios and, in subsequent sections, enables the computationally feasible recovery of a secret key K. From here on, it is assumed that side channel leakage is sufficient to identify the trace sections which correspond to individual long integer multiplicative operations (or elliptic curve point operations) when the key is used. Given that a standard counter-measure to timing attacks is to ensure that these operations always take the same number of clock cycles, it should be relatively straight-forward to divide the trace correctly from knowledge of the expected number of operations.

The decision about whether a section of side channel leakage represents a particular type of multiplicative operation is not clear cut. As a result of noise, the decision can only be made with a certain probability of correctness. Initial processing of the leakage from each operation uses templates of the expected leakage from each operation type and takes account of the relative probabilities with which the particular digits of $\mathcal{D}$ occur in the given recoding scheme. This yields a set of probabilities, one for each $r \in \mathcal{D}$ and summing to 1, that the operation involved digit r. For convenience, we define a *trace* to be the result of this pre-processing:

Definition 3. *$\mathcal{T} \subseteq ([0,1]^{|\mathcal{D}|})^*$ denotes the set of traces considered by an attacker. The i^{th} element t_i of $T \in \mathcal{T}$ is a list of probabilities, one for each $r \in \mathcal{D}$, that the corresponding side channel measurement represents the operation r.*

Formally, for power attacks for example, each operation $r \in \mathcal{D}$ induces a probability distribution on the set of all possible power traces. The exact probability vector t_i follows from the convex combination of these probability distributions with regard to the probabilities with which the particular operations occur (generally) in the recoding scheme. In real-world scenarios the probability vectors t_i can hardly be determined exactly, but roughly estimated.

Using the above definition of a trace, we need to convert each trace $T_j \in \mathcal{T}$ into a guess $G_j \in \mathcal{G}$ before applying Theorem 1(ii) to this situation. The straightforward strategy is to treat each operation separately and to select the most likely candidate $g_{j,i} \in \mathcal{D}$, i.e. the operation that maximises $t_{j,i}$. Formally, the conditional probabilities $p(g_{\ldots}|r_{\ldots})$ used in Theorem 1 are given by first averaging

over sections of side channel traces (such as those described in the previous paragraph) which correspond to the execution of the same multiplicative operation $r_{\ldots} \in \mathcal{D}$ at the same position within the recoding sequence. Then, secondly, these conditional probabilities are averaged over the different positions in the recoding sequence. Practically, these conditional probabilities can simply be estimated by applying this procedure (assigning trace sections, deciding on the most probable operation etc.) to a sample of side channel traces with known recodings.

For many applications this strategy should be appropriate. However, depending on the concrete scenario it may have two difficulties. The first is that the probabilities in Theorem 1(ii) neither depend on the particular side channel trace nor (which may be of less importance) on the position i. In contrast to Theorem 2 below, Theorem 1(ii) applies averaged probabilities. An advantage of Theorem 1(ii) and, in particular, of Theorem 1(iii) is their simple formulae. But using Theorem 2 and traces avoids the second difficulty, namely that of explicitly determining the best values G_j to use in advance[5].

Formally, for $N = 1$, Theorem 2 may be viewed as a special case of Theorem 1(i), which considers the whole recoding sequence R_1 and the guess G_1 at the same time. (Note that each t_{j,i,r_i} may formally be replaced by a trace- and position-dependent conditional probability $p_{j,i}(g_{j,i} \mid r_i)$ with some guess $g_{j,i}$.) A straight-forward generalisation of Theorem 1(i), allowing trace-dependent conditional probabilities $\mathrm{Prob}_j(X_{\mathcal{G},j}{=}G_j \mid X_{\mathcal{R}}{=}R)$, comprises the general case $N \geq 1$. In particular, the proof of Theorem 1 can easily be adapted to Theorem 2. Theorem 1(iv) and (v) also remain valid for traces.

Theorem 2. *Given traces $T_1, \ldots, T_N \in \mathcal{T}$, the optimal decision strategy $\tau_* : \mathcal{T}^N \to \mathcal{K}$ selects a key $K^* \in \mathcal{K}$ that maximises the expression*

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K): \\ \mathrm{len}(R)=\mathrm{len}(T_j)}} \nu(R) \prod_{i=0}^{\mathrm{len}(T_j)-1} t_{j,i,r_i} \right) - (N{-}1) \log\left(\eta(K)\right). \qquad (10)$$

If this maximum is attained for several keys the first is chosen under any preselected order on $\mathcal{K}$.

6 Application of the Main Theorem

In Sections 4 and 5 we determined optimal decision strategies. To demonstrate the power of its optimal decision strategy, Theorem 1(ii) was applied to small key sizes with the Ha-Moon recoding scheme using Algorithm 1 of the Appendix. Algorithm 1 allows one to compute efficiently the credibility function $\mathrm{cred}(G, K)$ for any required (G, K) and key size. The Ha-Moon recoding scheme maps the binary representation of key K into a representation that uses the digits 0, 1 and -1. Rather than using $\mathcal{D} = \{0, 1, \overline{1}\}$ also for the recoding sequences, to avoid

[5] Indeed, choosing the $g_{j,i}$ first and independently for all positions may lead to impossible guesses which are not recoding sequences of any key.

confusion we will write these operation sequences using 'S' to denote a squaring, 'M' a multiplication by the base C, and '$\overline{M}$' a multiplication by C^{-1}. Then digits 0, 1, and -1 in the recoding correspond to the operation sequences 'S', 'SM' and '$S\overline{M}$' respectively.

A large number of stochastic simulations were performed. In each simulation we generated: a random key $K \in \mathcal{K} \stackrel{\text{def}}{=} \{0,1\}^n \setminus \{(0,\ldots,0)\}$ where n denotes the length of K, N random recodings which delivered operation sequences $R_1,\ldots,R_N$ and some related recoding guesses $G_1,\ldots,G_N$. More precisely, the guess G_j was derived from the operation sequence R_j by replacing the correct operations independently with the conditional probabilities $p(\text{'}M\text{'}|\text{'}S\text{'}) = 0.2$, $p(\text{'}\overline{M}\text{'}|\text{'}S\text{'}) = 0.1$, $p(\text{'}\overline{M}\text{'}|\text{'}M\text{'}) = 0.2$, $p(\text{'}S\text{'}|\text{'}M\text{'}) = 0.1$, $p(\text{'}M\text{'}|\text{'}\overline{M}\text{'}) = 0.2$ and $p(\text{'}S\text{'}|\text{'}\overline{M}\text{'}) = 0.1$, which corresponds to noise in the side channel traces. Hence $p(\text{'}Y\text{'}|\text{'}Y\text{'}) = 0.7$ for each operation 'Y'. Given $G_1,\ldots,G_N$ we applied Algorithm 1 exhaustively to all $2^n - 1$ admissible keys K'. For each row in Table 1 we performed 100 simulations. The integers in the column entitled "The correct key was ranked 2", for instance, give the number of simulations out of 100 for which the correct key was ranked second for the given parameter set (Key length, N), e.g. 5 out of 100 for key length 15 and $N = 10$.

		The correct key was ranked					
Key length	N	1	2	3-9	10-99	100-999	> 1000
15	2	9	2	19	38	27	7
15	3	10	7	26	36	19	2
15	5	45	12	20	18	4	1
15	10	84	5	9	1	1	0
20	10	57	20	20	2	1	0

Table 1. Ha-Moon recoding scheme: simulation results for short keys.

These results clearly underline the strength of the optimal decision strategy. However, a major problem with the optimal decision strategy is that it is computationally infeasible for keys of cryptographic size – there are too many keys to search for the best one. Unlike in, e.g., [6, 7] or in several examples in [10] we cannot handle single key bits or small blocks of key bits either independently or at least sequentially. Instead, the optimal strategy does not describe how to recover *part* of a key, only how to recover the *whole* key. In general it is not possible to associate the key bits independently yet correctly with recoding operations. Where such an association is possible, the key bits might be recovered independently and in parallel or sequentially with effort which is linear rather than exponential in the key length.

7 Incremental Key Construction

General complexity and other issues (*see* [13], §1) seem to dictate that the best fit (i.e. most probable) key that it is feasible to find has to be generated sequentially bit by bit. We take this approach here, following [13], and this enables the processing of partial keys to be closely related to the theory already presented for full length keys.

It makes sense to determine the bits in the order that recoding takes place, which is assumed here, for convenience, to start at the least significant (rightmost) end. So key suffixes of increasing length will be generated. This is done using corresponding suffixes of 'traces' in the sense of Definition 3. In order to maximise the expression in Theorem 2, each key suffix must normally have, for each trace, a very good fit between one (or more) of its recoding sequences and the suffix of the trace with the same length. In subsequent sections we try to justify this reasonable assumption while working out its implications. If a sufficiently large set of good key suffixes are processed the best fit key suffix should always be included, so that the best key emerges at the end of the algorithm. However, the longer the key the more likely there is to be a suffix which is too poor a fit to be retained, and our algorithm will fail more frequently in these cases.

Suppose we select a key suffix $K^{(n)}$ of length n which yields the best match to all the leakage from that point onwards, i.e. the best match to trace suffixes. Formally, under the optimal decision strategy model $K^{(n)}$ provides the maximum of

$$\sum_{j=1}^{N} \log \left(\sum_{\substack{R \in \mathcal{R}(K'): \\ \mathrm{len}(R) \le \mathrm{len}(T_j)}} \nu(R) \prod_{i=0}^{\mathrm{len}(R)-1} t_{j,i,r_i} \right) - (N-1) \log \left(\eta(K') \right) \qquad (11)$$

over all keys K' of n bits. (There are some minor issues with end conditions which we will ignore.) If all bits of $K^{(n)}$ are accepted as belonging to the near best-fit full length key K then a decision is being made on the initial, i.e. leftmost, bits of $K^{(n)}$ which does not take into account all available information. Recoding choices are not made independently of previous input; they depend on the state of the recoding finite automaton arising from the previous input, and the influence can persist measurably for several bits. Hence we should ignore the first λ bits, say, of $K^{(n)}$ and choose only its last $n-\lambda$ bits. Thus, the obvious algorithm is to compute iteratively for $n = 1, 2, 3...$ the fitness of every $K^{(n)}$ which extends the previously chosen $K^{(n-\lambda-1)}$, select the best, and use that to determine $K^{(n-\lambda)}$.

This algorithm with the above formula was applied to the (first) Ha-Moon recoding scheme [4], but so far without success. Even worse, under the leakage model described in [13], the resulting "best fit" full length key was indistinguishable from a randomly chosen key – on average half the bits were incorrect. This spurred an investigation into what modifications are necessary to obtain useful results as it is known (e.g. from [3, 5, 13]) that it *is* possible to recover the secret key when leakage is weak. It was hoped that the optimal decision strategy would lead to a much more powerful algorithm.

8 Simulation Experiments

With the Ha-Moon scheme [4] as a test case, the formula (11) was modified in a large number of ways to discover what choices lead to a useful, computationally feasible algorithm. By Theorem 1(iv) choice of a uniform key space $\mathcal{K}$ allowed the formula to be simplified by ignoring the $\log(\eta(K))$ term.

Of all the modifications attempted, only one seemed essential to obtain anything better than how a random key choice would perform, and that was substituting *Max* for the second $\sum$ in (11). Thus we became interested in iteratively finding the n-bit suffix $K^{(n)}$ which closely maximises

$$\sum_{j=1}^{N} \log \left(\text{Max}\{ \ \nu(R) \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i} \mid R \in \mathcal{R}(K^{(n)}),\ \text{len}(R) \le \text{len}(T_j) \} \right). \quad (12)$$

(Note that (12) moves the search for a maximum within subsets of recoding sequences to individual recoding sequences. This is computationally more easy.) With sufficiently strong leakage and enough traces, this change results in a significant portion of the predicted bits of K being correct. However, it exposes the factor $\nu(R)$ which gives undue weight to shorter recoding sequences. In the original formula (10) recoding sequences which were too short could not arise because of the constraint $\text{len}(R) = \text{len}(T_j)$. Now, with only an upper bound on the length, some re-balancing is necessary. Normalising using $\text{len}(R)$ yields the much more frequently successful formula

$$\sum_{j=1}^{N} \log \left(\text{Max}\{ \prod_{i=0}^{\text{len}(R)-1} t_{j,i,r_i}{}^{1/\text{len}(R)} \mid R \in \mathcal{R}(K^{(n)}),\ \text{len}(R) \le \text{len}(T_j) \ \} \right). \quad (13)$$

We could have added a weighting that accounted more for the expected length of a recoding sequence for an n-bit key which has given trace length for the full key, but did not do so.

It was noted above that only the final $n-\lambda$ bits should be chosen from $K^{(n)}$. This provides the possibility of treating the first λ bits differently from the later ones. In particular, the two modifications described for (11) can be applied only to the recoding operations corresponding to the last $n-\lambda$ bits or only to the operations corresponding to the first λ bits. However, this different treatment of the two sequences of bits did not give better results.

The formula (13) and related algorithm are now close to what was used in [13]. The main difference is that here we have a product of probabilities to maximise rather than the following sum of distances to minimise:

$$\sum_{j=1}^{N} \text{Min}\{ \ \frac{1}{\text{len}(R)} \sum_{i=0}^{\text{len}(R)-1} (1-t_{j,i,r_i}) \ \mid \ R \in \mathcal{R}(K^{(n)}),\ \text{len}(R) \le \text{len}(T_j) \ \}. \quad (14)$$

However, the absence of the "log" from this formula suggests that the contributions from each trace might be either multiplied or added together. So a

promising alternative to (13) might be found in

$$\frac{1}{N}\sum_{j=1}^{N} \mathrm{Max}\{ \prod_{i=0}^{\mathrm{len}(R)-1} {t_{j,i,r_i}}^{1/\mathrm{len}(R)} \mid R \in \mathcal{R}(K^{(n)}),\ \mathrm{len}(R) \le \mathrm{len}(T_j)\ \} \quad (15)$$

which is normalised with respect to the number of traces N. In practice, the simulations did work a little better for this formula than for (13), so that it became our final choice. Moreover, this "most probable" version proved to be significantly better than the "best fit" version (14) which appears in [13] unless a very specific choice of parameters is made in [13].

9 An Ordered Search

The algorithm of the previous section rarely identifies the correct key although, with enough traces and sufficiently strong leakage, the resulting key $K^* = K^{(\mathrm{len}(K))}$ should yield a close to maximal value for (2), as should the correct key. The remaining problem is therefore to organise a computationally feasible ordered search of the key space to find the correct key K from K^*.

In a simulation, a quick scan of this close-to-most-probable key K^* shows that, unless the strategy failed, most of the key bits are correct, or differ from the correct ones according to specific patterns which make no difference to the likelihood of the key (such as a sequence of one or more inverted bits). Thus, the correct key might be found by allowing for all the related patterns of equal probability and by looking at each bit and obtaining an estimate of confidence in its correctness. This estimate is naturally based on the values returned by the expression (15) which was used to choose the bit initially. Then the search of the key space is performed by changing more and more of those bits for which confidence is lowest (and any subsequent related pattern) until the correct key is found.

With the last $n-\lambda-1$ bits of K^* already chosen, there are $2^{\lambda+1}$ choices for $K^{(n)}$ which are supplied to (15). There are 2^λ cases for which bit $n-\lambda$ is 0, and 2^λ cases for which bit $n-\lambda$ is 1. Let $\mathrm{cred}(0, n-\lambda)$ and $\mathrm{cred}(1, n-\lambda)$ denote the maximum values returned by (15) when evaluated over these two partial key subsets. Whichever is larger determines the choice of bit $n-\lambda$, and so it is natural to use some measure of their proximity to provide a confidence value for the bit decision. Out of several possibilities,

- an effective discriminant was found to be the larger of the two ratios $\mathrm{cred}(1, n-\lambda)/\mathrm{cred}(0, n-\lambda)$ or $\mathrm{cred}(0, n-\lambda)/\mathrm{cred}(1, n-\lambda)$.

The larger this ratio, the greater the confidence that the bit is chosen correctly.

When evaluating (15) over the 2^λ keys of interest, the variance of $\mathrm{cred}(b, n-\lambda)$ decreases as n increases. This implies that, on average, the confidence function will return smaller values with increasing key size. Of course, this is what happens, and it corresponds to the fact that bits are predicted with decreasing accuracy as n becomes larger. A re-scaling to compensate for this is the modification

to (15) which results in the following expression for maximising:

$$\frac{1}{N}\sum_{j=1}^{N} \mathrm{Max}\{ \prod_{i=0}^{\mathrm{len}(R)-1} {t_{j,i,r_i}}^{n/\mathrm{len}(R)} \mid R \in \mathcal{R}(K^{(n)}),\ \mathrm{len}(R) \le \mathrm{len}(T_j)\ \} \quad (16)$$

This re-scaling may give some benefit when applied, but was usually marginally poorer than (15) over all the choices of parameters we tried, such as variations in the level of leakage and in the key lengths.

10 Complexity

This section considers the space and time complexity for obtaining a near best-fit key K^* using expression (15). We are assuming that all necessary pre-processing has already been performed to derive traces in the sense of Definition 3 from the side channel measurements and template information.

Our model assumes 1 unit of time for any arithmetic operation involving one or two bits. It also assumes one unit of time for moving (reading, writing etc.) or copying a single bit and one unit of space to store a single bit. This is not quite realistic since address sizes and wire connect grow as the logarithm of the volume of data, but is adequate for the quantities of data under consideration. Moreover, since most bits are effectively random and we avoid storing multiple copies of data, there are no data compression techniques which are likely to be useful.

For convenience, we further assume that all probabilities can be stored with sufficient accuracy using $O(1)$ bits. In practice our simulations worked using 32-bit real arithmetic without any hint of problems. Overall this complexity model enables one to run small trial cases first, and scale up using the complexity results to obtain good approximations to the space and time requirements for an attack on secret keys of cryptographic size.

Following earlier notation, we have a key K^* of n_K bits which is constructed bit by bit, N traces, and λ leading bits of the suffixes which we ignore when selecting the next bit of K^*. At any one time we have decided the least significant $n - \lambda - 1$ bits of K^*. We have $2^{\lambda+1}$ possibilities for the remaining bits of the n-bit suffix and hence that number of keys to consider when determining the next bit of K^*.

Recall our assumption that recodings are converted into sequences of operations with digit 0 generating one operation (a squaring) and non-zero digits generating two (a squaring and a multiplication). Consequently, when the recoding finite automaton (FA) has read a suffix of n bits, the resulting recoding sequences may have any length from n up to $2n$. Suppose also that the recoding FA has F possible states. These states include, but are not limited to, storing the shifted difference (i.e. carry or borrow) between the value of the key suffix read by the FA and the recoding output by it. (For the Ha-Moon scheme this is always 0 or 1, and the corresponding FA has two states.) Whenever two recoding sequences have the same length and left the FA in the same state, we can ignore

the one with the smaller value of the product of trace values in (15). It cannot give the maximum, nor contribute to any maximum when the FA recodes more bits. Hence there are only up to $F(n+1)$ sequences which need to be maintained in order to select the best recoding.

The iterative step starts at $n = \lambda + 1$ in order to determine the bit of index 0. If there were just one trace then, after incrementing n, the general induction step would start with a set of best recoding sequences for the least significant $n - \lambda - 1$ bits of K^* and the corresponding values for the product of traces values needed in (15). There are up to $(n-\lambda)F$ of these, so $O((n-\lambda)F)$ space is required for this. In fact, this is needed for all N traces, so $O((n-\lambda)FN)$ space is used to hold the data required by the induction step. A three dimensional array is used for this in order to have direct access in unit time to the data elements. The total space order is also enough to include the decided bits of K^*.

Extending the recoding sequences from representing a suffix of length $n-\lambda-1$ to a suffix of length n just means generating the same set of data incrementally for longer suffixes. A depth first traversal is made of the binary tree of depth $\lambda+1$ representing the remaining choices for the n-bit suffix of K^*. Along each branch of the tree one such data set needs to be stored at each node. So $O(nFN\lambda)$ space is required for data storage during the induction step. At each node in the tree, each recoding sequence of the parent node's data set is extended by processing the key bit value which labels the current node. The FA generates $O(F)$ choices which are used to create or update the items in the data set of this node. This means $O(F)$ time per recoding sequence, and a total of $O(nF^2N)$ per node of the tree. With $2^{\lambda+2} - 2$ nodes to treat, the time order is $O(2^\lambda nF^2N)$.

If, instead, we are able to hold all the data sets for all the nodes in the above tree simultaneously, then we do not need to regenerate the same data for up to λ consecutive values of n. Instead we traverse the leaves of the tree from the preceding value of n and generate the data for the leaves of the tree for the current value of n. However, as this still means traversing $2^{\lambda+1}$ nodes, the time order is not reduced although there will be a speed-up by a factor of about 2.

Whenever the data set for a leaf of the tree has been generated, we have the value of the product term in (15) for all the $O(Fn)$ recoding sequences associated with each trace. Hence the maximum value can be obtained for each trace, and the sum over all traces calculated. This does not add to the time complexity as it requires $O(1)$ time per sequence. Finally, we must obtain the maximum value of (15) over the $2^{\lambda+1}$ key choices in order to determine bit $n - \lambda$ of K^*. Again, this does not add to the time order. If we want to rank the bits in order of certainty, then the ratio of credibility values is obtained at this point by taking the maximums over the two sub-trees corresponding to the two choices for bit $n - \lambda$.

Thus, neglecting special processing for the most significant λ bits of K^* (when we must decrease λ rather than increase n), the iterative process to compute K^* takes $O(n_K FN)$ space and $O(2^\lambda n_K F^2 N)$ time.

11 Numerical Results

Direct comparison of the improved algorithms here with the results of [13] is made difficult by the fact that [13] limits the key search to a fixed maximum number of recoding states[6] whereas here the number is not limited. Specifically, in (15) the maximum is given by incrementally extending a set of best recoding sequences whenever another key bit is decided, and there must be a best recoding sequence for every recoding state that might have been reached so far. For an n-bit key suffix, the Ha-Moon algorithm could have generated a recoding sequence of up to $2n$ operations and left the recoding finite automaton in one of two states. Hence there are up to $4n$ best recoding sequences per trace which need to be extended in order to extract the maximum, and restricting this number turns out to be detrimental. Hence, in order to generate comparative values we modified our simulation to use Walter's metric but without his bound on recoding states. On its own, this modification arising from the optimal strategy approach led to the majority of the improvement in performance tabulated below – witness the first two rows in Table 2.

Unlike the example in Section 6 we assume a side-channel leakage which only allows us to distinguish between squarings and multiplications with some certainty but not between the two multiplicative operations 'M' and '$\overline{M}$' corresponding to digits 1 and -1. This is easy to model. The i^{th} element t_i of a trace T is a set of three probabilities, one for each operation type: $t_i = \{p_{i0}, p_{i1}, p_{i\overline{1}}\}$ in which we ensure that $p_{i1} = p_{i\overline{1}}$. We selected this scenario to have a fair comparison with the results in [13].

Let L denote the average level of side channel leakage, that is, the fraction of square or multiply operations which are independently guessed correctly. So $L = \frac{1}{2}$ means no leakage, when blind guessing makes half the bits correct, and $L = 1$ corresponds to full leakage, when all operations and hence all bits are completely determined[7]. For several realistic cryptographic key lengths and numbers of traces, Table 2 gives the probability that all the incorrectly guessed bits of the most likely key are among the 24 bits which the credibility metric shows are most likely to be in error (*see* Section 9). It is computationally feasible to correct all errors in such guessed keys and thereby recover the true key. The last column explores the possibility that the most probable key guesses are those for which the bit errors are confined to the 24 most dubious positions. This new measure is not in previous literature, and clearly indicates that the adversary can select cases for which the method is more likely to succeed. He collects sets of side channel data for a number of different keys, computes the best-fitting key in each case, and then selects the 10%, say, which yield the highest value for (15). Then, in parallel for the selected keys, he modifies more and more of the

[6] The recoding state is a pair consisting of the state of the recoding finite automaton and the number of operations it has generated.

[7] Guessing all operations to be squarings in the binary exponentiation algorithm makes two thirds of the operations correct (on average), not a half. Here we are modelling the noise which is added to the actual recoding sequence, changing the average correctness of the decision between squaring or multiplication from 1 to L.

most dubious bit values in decreasing order of likelihood until a correct key is found.

Method	Leakage Level L	Key Length	No. of Traces	Av. No. Bit Errors	Fraction with all errors in worst 24	Fraction for 10% best-fitting Keys
[13][1]	0.7	192	5	–	0.0027	–
[13][2]	0.7	192	5	23.2	0.011	0.020
Here	0.7	192	5	20.7	0.013	0.014
[13][2]	0.7	384	10	19.0	0.003	0.002
Here	0.7	384	10	18.2	0.003	0.010
[13][2]	0.7	1024	40	13.5	0.3	0.3
Here	0.7	1024	40	13.4	0.28	0.28
”	0.6	192	64	30.9	0.19	0.43
”	0.6	384	128	39.4	0.12	0.24

Table 2. Fraction of key guesses with all bit errors in the 24 most dubious positions for Ha-Moon recoding [4]. [1]= original for 10 recodings, [2]= improved version (see text).

12 Conclusion

A computationally feasible algorithm has been presented for determining the secret key used repeatedly in exponentiations where there is weak side channel leakage and randomised recoding has been employed in an attempt to nullify the effect of that leakage. The algorithm was derived from an optimal decision strategy and is an improvement over prior techniques. Using it, it is easy both to determine which results have few bit errors, and to locate the potential bit errors. Hence it is frequently possible to recover the key using much weaker leaked data than before. The derivation also highlights points where significant modifications had to be made to the optimal strategy to obtain a computationally feasible algorithm.

References

1. E. Brier & M. Joye, *Weierstraß Elliptic Curves and Side-Channel Attacks*, Public Key Cryptography (Proc. PKC 2002), D. Naccache & P. Paillier (editors), LNCS **2274**, Springer-Verlag, 2002, pp. 335–345.
2. D. Chaum, *Blind Signatures for Untraceable Payments*, Advances in Cryptology – Crypto ’82, D. Chaum, R.L. Rivest & A.T. Sherman (editors), Plenum Press, New York, 1983, pp. 199–203.
3. P. J. Green, R. Noad & N. Smart, *Further Hidden Markov Model Cryptanalysis*, Cryptographic Hardware and Embedded Systems – CHES 2005, J.R. Rao & B. Sunar (editors), LNCS **3659**, Springer-Verlag, 2005, pp. 61–74.

4. J. C. Ha & S. J. Moon, *Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. K. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2002, pp. 551–563.
5. C. Karlof & D. Wagner, *Hidden Markov Model Cryptanalysis*, Cryptographic Hardware and Embedded Systems – CHES 2003, C. D. Walter, Ç. K. Koç & C. Paar (editors), LNCS **2779**, Springer-Verlag, 2003, pp. 17–34.
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.-Y. Liardet & N. P. Smart, *Preventing SPA/DPA in ECC Systems using the Jacobi Form* Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. K. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 391–401.
9. E. Oswald & M. Aigner, *Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. K. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, pp. 39–50.
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. C. D. Walter, *Breaking the Liardet-Smart Randomized Exponentiation Algorithm*, Proc. Cardis '02, San José, November 2002, Usenix Association, Berkeley, 2002, pp. 59–68.
12. C. D. Walter, *Issues of Security with the Oswald-Aigner Exponentiation Algorithm*, Topics in Cryptology – CT-RSA 2004, T. Okamoto (editor), LNCS **2964**, Springer-Verlag, 2004, pp. 208–221.
13. C. D. Walter, *Recovering Secret Keys from Weak Side Channel Traces of Differing Lengths*, Cryptographic Hardware and Embedded Systems – CHES 2008, E. Oswald & P. Rohatgi (editors), LNCS **5154**, Springer-Verlag, 2008, pp. 214–227.
14. H. Witting, *Mathematische Statistik I*, Teubner, Stuttgart 1985.
15. S.-M. Yen, C.-N. Chen, S. J. Moon & J. C. Ha, *Improvement on Ha-Moon Randomized Exponentiation Algorithm*, Information Security and Cryptology – ICICS 2004, C. Park & S. Chee (editors), LNCS **3506**, Springer-Verlag, 2005, pp. 154–167.

A Appendix

A.1 Algorithmic Treatment of Ha-Moon Recodings

In this subsection we provide an efficient algorithm to compute $\mathrm{cred}(G, K)$, defined by (8), for the Ha-Moon recoding scheme [4] for any fixed $G \in \mathcal{G}$ and $K \in \mathcal{K}$. In the recoding step the binary representation K is mapped onto a recoding which is a sequence of digits in $\{-1, 0, 1\}$ with length $\mathrm{len}(K)$ or $\mathrm{len}(K)+1$. Again, we follow a standard convention in which the letters 'S', 'M', and '$\overline{M}$' denote a squaring, a multiplication by the base C, and a multiplication by C^{-1}, respectively. So, to obtain the corresponding operation sequence, apart from the

most significant digit (= 1), each digit of the recoding is substituted as follows: $0 \mapsto$ 'S', $1 \mapsto$ ('S','M'), and $-1 \mapsto$ ('S','$\overline{M}$'), yielding an element $R \in \mathcal{R}(K)$. In particular, the recoding sequence has length $\leq 2\text{len}(K)$. Note that the probabilities $\nu(R)$ are not identical for all $R \in \mathcal{R}(K)$ in the Ha-Moon recoding scheme.

Assumption 1: We consider the scenario from Theorem 1(ii). We assume further that $K \in \mathcal{K} \stackrel{\text{def}}{=} \{0,1\}^n$, and that $X_{\mathcal{K}}$ is uniformly distributed on $\mathcal{K}$, i.e., each admissible key is equally likely. If the next recoding step is not unique the recoding algorithm decides with probability $\frac{1}{2}$ for one of the two alternatives.

Definition 4. *(Ha-Moon recoding scheme [4].) For $0 \leq v \leq \text{len}(K)$ and $K \in \mathcal{K}$ the term $\mathcal{R}(K, v, \text{len}, c_v) \subseteq \mathcal{R}(K)$ denotes the subset of recoding sequences that require exactly* len *operations ('S', 'M', '$\overline{M}$') and the carry bit c_v to encode the v least significant bits of K. Guesses of recoding sequences $G = (g_j)_{0 \leq j < \text{len}(G)}$ and recoding sequences $R = (r_j)_{0 \leq j < \text{len}(R)}$ are extended to length $2n$ by defining a further symbol, ∞, and setting $g_j \stackrel{\text{def}}{=} \infty$ for $j \geq \text{len}(G)$ and $r_j \stackrel{\text{def}}{=} \infty$ for $j \geq \text{len}(R)$. Further, the conditional probability function $p(\cdot \mid \cdot)$ is extended so that $p(\infty|\infty) \stackrel{\text{def}}{=} 1$ and $p(a|\infty) \stackrel{\text{def}}{=} p(\infty|a) \stackrel{\text{def}}{=} 0$ for $a \in \{$'S','M','$\overline{M}$'$\}$. This allows one to increase the upper bound of the product in (4) beyond $\text{len}(R) - 1$ to $2n - 1$. For $v \geq 0$ the 'intermediate' credibility function associated with the guess G and any subset $M \subseteq \mathcal{R}(K) = \bigcup_{v \leq \text{len} \leq 2v;\ c \in \{0,1\}} \mathcal{R}(K, v, \text{len}, c)$ is*

$$\text{cred}_v(G, M) \stackrel{\text{def}}{=} \sum_{R \in M} \nu(R) \prod_{i=0}^{\text{len}_v(R)-1} p(g_i \mid r_i). \tag{17}$$

The term $\text{len}_v(R)$ denotes the number of operations in R which are used to encode the key bits $k_{v-1}, ..., k_0$. (In particular, for all $R \in \mathcal{R}(K, v, \text{len}, c)$ it is $\text{len}_v(R) = \text{len}$.)

During recoding, the Ha-Moon scheme generates either one or two operations per bit of K, and a carry bit equal to 0 or 1. Hence for given v the length of a recoding sequence ranges from v to $2v$, and so $\mathcal{R}(K)$ is the disjoint union of $2(v + 1)$ (possibly empty) subsets $\mathcal{R}(K, v, \text{len}, c)$ with $v \leq \text{len} \leq 2v$ and $c \in \{0, 1\}$. Algorithm 1 below computes sequentially the $\text{cred}_v(G, \cdot)$-values for increasing parameters v and subsets $\mathcal{R}(K, v, \text{len}, c) \subseteq \mathcal{R}(K)$, which finally yields the desired value $\text{cred}(G, K)$. The definition of the subsets $\mathcal{R}(K, v, \text{len}, c)$ and the $\text{cred}_v(\cdot, \cdot)$-function are closely related to the components of definition (9) used in the generic description of an algorithm for the efficient computation of $\text{cred}(G, K)$. Indeed, both approaches are essentially equivalent. In definition (9) 'truncated' recoding sequences are considered, and the product in (17) only considers these operations. The recoding sequences are elements of $\mathcal{R}(K)$ but the probabilities of all recoding sequences with a fixed suffix add up to the probability of the truncated recoding sequence. However, the approach chosen in the appendix is more convenient for Algorithm 1 which splits and merges subsets of $\mathcal{R}(K)$.

The goal of Algorithm 1 is the efficient computation of $\text{cred}(G, K)$ for arbitrary but fixed $G \in \mathcal{G}$ and $K \in \mathcal{K}$.

Remark 3. Our implementation of Algorithm 1 requires the binary representation of key K to contain at least one '1', while Theorems 1 and 2 clearly also cover the zero key. In our simulation experiments presented in Section 6 we restricted the key space to $\{0,1\}^n \setminus \{(0,\ldots,0)\}$ for simplicity.

Algorithm 1. Initialise by setting $\mathrm{cred}_0(G,\mathcal{R}(K,0,0,0)) \stackrel{\mathrm{def}}{=} \mathrm{Prob}(X_{\mathcal{K}} = K)$. (Note that $\mathcal{R}(K) = \mathcal{R}(K,0,0,0)$.) Assume that we know the 'truncated' credibilities $\mathrm{cred}_v(G,\mathcal{R}(K,v,u,c_v))$ for all $u \in \{v,\ldots,2v\}$ and $c_v \in \{0,1\}$. The next task is therefore to consider key bit k_v and to determine the values $\mathrm{cred}_{v+1}(G, \mathcal{R}(K,v+1,u,c_{v+1}))$ for $v+1 \le u \le 2(v+1)$ and $c_{v+1} \in \{0,1\}$.

Induction Step (over v):

For each $len \in \{v,\ldots,2v\}$ and $c_v \in \{0,1\}$:

Case A: If $(k_v,c_v) = (0,0)$ the next digit (viewed from right to left) in any recoding is 0 and thus 'S' is the next operation in the recoding sequence. In particular, $\mathrm{cred}_{v+1}(G,\mathcal{R}(K,v{+}1,\,len{+}1,0) \cap \mathcal{R}(K,v,len,0)) = p(g_{len}\,|\,\text{'}S\text{'})\mathrm{cred}_v(G,\mathcal{R}(K,v,\,len,0))$.

Case B: If $(k_v,c_v) = (1,0)$ the next digit of the recoding is 1 or -1 (and thus $c_{v+1} = 0$ or $c_{v+1} = 1$, respectively), each with probability $\frac{1}{2}$. Consequently, $\mathrm{cred}_{v+1}(G,\mathcal{R}(K,v{+}1,len{+}2,0) \cap \mathcal{R}(K,v,len,0)) = \frac{1}{2}p(g_{len+1} \mid \text{'}S\text{'})p(g_{len} \mid \text{'}M\text{'}) \times \mathrm{cred}_v(G,\mathcal{R}(K,v,len,0))$ and $\mathrm{cred}_{v+1}(G,\mathcal{R}(K,v{+}1,len{+}2,1) \cap \mathcal{R}(K,v,len,0)) = \frac{1}{2}p(g_{len+1} \mid \text{'}S\text{'})p(g_{len} \mid \text{'}\overline{M}\text{'})\mathrm{cred}_v(G,\mathcal{R}(K,v,len,0))$.

Cases C and D: The cases $(k_v,c_v) = (0,1)$ and $(k_v,c_v) = (1,1)$ are treated analogously with $\mathcal{R}(K,v,len,1)$ in place of $\mathcal{R}(K,v,len,0)$. As in Case B above the set $\mathcal{R}(K,v,len,1)$ splits into two subsets if $(k_v,c_v) = (0,1)$, i.e. in Case C.

When this has been completed for all values $len \in \{v,\ldots,2v\}$ and $c_v \in \{0,1\}$, the credibility values for the subsets $\mathcal{R}(K,v+1,len',c_{v+1})$ are easily computed. For each $v+1 \le \mathrm{len}' \le 2(v+1)$ and $c_{v+1} \in \{0,1\}$ 'related' subsets are merged to give: $\mathrm{cred}_{v+1}(G,\mathcal{R}(K,v{+}1,len',c_{v+1})) = \sum \mathrm{cred}_{v+1}(G,\mathcal{R}(K,v{+}1,\,len',c_{v+1}) \cap \mathcal{R}(K,v,len'',c''))$, where the sum extends over all $(len'',c'') \in \{len'{-}1, len'{-}2\} \times \{0,1\}$. This equality holds because the subsets on the right side of the equation have a disjoint union equal to the set on the left side. Clearly, $\mathrm{cred}_v(G,\emptyset) = 0$ and $\mathrm{cred}_{v+1}(G,\emptyset) = 0$, and this enables instances of $\mathrm{cred}_{v+1}\ (G,\mathcal{R}(K,v{+}1,len',c'))$ to be evaluated without taking a sum over subsets when the second parameter is the empty set.

This completes the induction step from v to $v+1$.

This procedure is continued until $v = \mathrm{len}(K) - 2$ (inclusively). The most significant bit of K then needs special treatment because different operations are used to initialise the exponentiation. Finally, $\mathrm{cred}(G,K) = \mathrm{cred}_{\mathrm{len}(K)}(G, \mathcal{R}(K,\mathrm{len}(K),\mathrm{len}(G),0))$. □

Note that for $R \in \mathcal{R}(K)$ we have $\nu(R) = \mathrm{Prob}(X_{\mathcal{K}} = K)2^{-\mathrm{bif}(R)}$ where $\mathrm{bif}(R)$ stands for the number of 'bifurcations' in generating the recoding sequence R from K, i.e. the number of positions where two recoding choices are possible, i.e. when $(k_v,c_v) = (0,1)$ or $(1,0)$. This causes the factor $\frac{1}{2}$ for Cases B and C.

A.2 Statistical Decision Theory

This subsection provides a brief introduction to statistical decision theory as far as is relevant to understand the concepts of Theorems 1 and 2. We omit all mathematical details. For a more comprehensive treatment we refer the interested reader to [10], Section 2, or to textbooks in this field (e.g. [14]). Reference [10] illustrates the merits of statistical decision theory for side-channel analysis by several examples.

Our focus is side-channel analysis. We interpret side-channel measurements as *realisations* of random variables, i.e. as values assumed by these random variables. The relevant part of the information is covered by noise but an attacker clearly aims to exploit the available information in an optimal way. Statistical decision theory quantifies the impact of the particular pieces of information on the strength of a decision strategy, and so the search for the optimal decision strategy can be formalised.

Formally, a statistical decision problem is given by a 5-tuple $(\Theta, \Omega, s, \mathcal{DST}, A)$. The statistician (in our context the attacker) observes a sample $\omega \in \Omega$ that he interprets as a realisation of a random variable X with unknown distribution p_θ. On basis of this observation he guesses the parameter $\theta \in \Theta$ where Θ denotes the *parameter space*, i.e., the set of all admissible hypotheses (= possible parameters). Further, the set Ω is called the *observation space*, and the letter A denotes the set of all admissible alternatives the statistician can choose. In the following we assume $\Theta = A$ with finite sets Θ and A.

Example 1. ([10], Example 1)
(i) Assume that the attacker guesses a particular (single) RSA key bit and that his decision is based upon N measurements. Then $\Theta = A = \{0, 1\}$. For timing attacks, we may assume $\Omega = \mathbb{R}^N$ while $\Omega = \mathbb{R}^{TN}$ for power attacks where T is the number of relevant measurement points per power trace.
(ii) Consider a power attack on a DES implementation where the attacker guesses a particular 6-bit subkey that affects a single S-box in the first round. Then $\Theta = A = \{0, 1\}^6$.

The term $\mathcal{DST}$ denotes the set of all decision strategies between which the statistician can choose. A *(deterministic) decision strategy* is given by a mapping $\tau\colon \Omega \to A$. This means that if the statistician applies decision strategy τ he decides on $\tau(\omega) \in A = \Theta$ whenever he observes $\omega \in \Omega$. (It may be noted that for certain statistical applications it is reasonable to consider the more general class of randomised decision strategies ([10], Remark 1(i)). For our purposes we need only concentrate on deterministic decision strategies.)

Finally, the *loss function* $s\colon \Theta \times A \to [0, \infty)$ quantifies the harm of a wrong decision, i.e., $s(\theta, a)$ gives the loss if the statistician decides on $a \in A$ although $\theta \in \Theta = A$ is the correct parameter. In the context of side-channel attacks the loss function quantifies the efforts to detect, to localise and to correct a wrong decision, usually a wrong guess of a key part. Clearly, $s(\theta, \theta) \stackrel{\text{def}}{=} 0$ since a correct guess does not cause any loss. We point out that for some side-channel attacks

certain types of errors are easier to detect and correct than others ([10], Sect. 6). The optimal decision strategy takes such phenomena into account.

Assume that the statistician uses the deterministic decision strategy $\tau\colon \Omega \to A$ and that θ is the correct parameter. The expected loss (= average loss if the hypothesis θ is true) is given by the *r*isk function

$$r(\theta,\tau) \stackrel{\text{def}}{=} \int_{\Omega} s(\theta,\tau(\omega))\, p_\theta(d\omega). \tag{18}$$

In the context of side-channel attacks one can usually determine (at least approximate) probabilities with which the particular parameters occur. This is quantified by the so-called *a priori distribution* η, a probability measure on the parameter space Θ.

Example 2. ([10], Example 2)
(i) (Continuation of Example 1(i).) Assume that k exponent bits remain to be guessed and that the attacker knows that r of them are 1. If the secret key was selected randomly it is reasonable to assume that a particular exponent bit is 1 with probability $\eta(1) = r/k$.
(ii) (Continuation of Example 1(ii).) Here $\eta(x) = 2^{-6}$ for all $x \in \{0,1\}^6$.

Assume that η denotes the a priori distribution. If the statistician applies the deterministic decision strategy $\tau\colon \Omega \to A$ the expected loss is given by

$$R(\eta,\tau) \stackrel{\text{def}}{=} \sum_{\theta\in\Theta} r(\theta,\tau)\eta(\theta) = \sum_{\theta\in\Theta} \int_{\Omega} s(\theta,\tau(\omega))\, p_\theta(d\omega)\, \eta(\theta). \tag{19}$$

A decision strategy τ' is *optimal against* η if it minimises the right-hand term. Such a decision strategy is also called a *Bayes strategy* against η.

Theorem 3. *([10], Theorem 1(i), (iii))*
Assume that $(\Theta, \Omega, s, \mathcal{DST}, A)$ defines a statistical decision problem with finite parameter space $\Theta = \{\theta_1, \ldots, \theta_t\} = A$ where $\mathcal{DST}$ contains all deterministic decision strategies. Further, let μ denote a σ-finite measure on Ω with $p_{\theta_i} = f_{\theta_i} \cdot \mu$, i.e. p_{θ_i} has μ-density f_{θ_i}, for each $i \le t$.
(i) The deterministic decision strategy $\tau\colon \Omega \to A$,

$$\tau(\omega) \stackrel{\text{def}}{=} a \quad \text{if } \sum_{i=1}^{t} s(\theta_i,a)\eta(\theta_i)f_{\theta_i}(\omega) = \min_{a'\in A}\left\{\sum_{i=1}^{t} s(\theta_i,a')\eta(\theta_i)f_{\theta_i}(\omega)\right\} \tag{20}$$

is optimal against the a priori distribution η. (If the minimum is attained for several decisions, we chose $a \in A$ according to any pre-selected order on A.)
(ii) Assume that $C \subseteq \Omega$ with $p_{\theta_i}(C) = p > 0$ for all $\theta_i \in \Theta$. Then (i) and (ii) remain valid if f_θ is replaced by the conditional density $f_{\theta|C}$.

Remark 4. ([10], Remark 2)
(i) A σ-finite measure μ on Ω with the properties claimed in Theorem 3 always exists (e.g. $\mu = p_{\theta_1} + \cdots + p_{\theta_t}$).

(ii) For $\Omega = \mathbb{R}^n$ the well-known Lebesgue measure λ_n is σ-finite. (The Lebesgue measure on $\mathbb{R}^n$ is given by $\lambda_n([a_1, b_1] \times \cdots \times [a_n, b_n]) = \prod_{i=1}^{n}(b_i - a_i)$ if $b_i \geq a_i$ for all $i \leq n$.) If Ω is finite or countable the counting measure μ_C is σ-finite. The counting measure is given by $\mu_C(\omega) = 1$ for all $\omega \in \Omega$. In particular, the probabilities $\mathrm{Prob}_\theta(X = \omega) = p_\theta(\omega)$ can be interpreted as densities with respect to μ_C.
(iii) The examples mentioned in (ii) and combinations thereof cover the cases that are relevant in the context of side-channel analysis.

The probability densities f_θ, the a priori distribution η, and the loss function s have an impact on the optimal decision strategy. The probability densities f_θ clearly have most influence, and usually their determination is the tough part of the work.