Side Channel Analysis for Reverse Engineering (SCARE)

An Improved Attack Against a Secret A3/A8 GSM Algorithm

Christophe Clavier

Gemplus S.A., Card Security Group
La Vigie, ZI Athélia IV, Avenue du Jujubier, F-13705 La Ciotat Cedex, France
christophe.clavier@gemplus.com — http://www.gemplus.com/smart/

Abstract. Side-channel analysis has been recognized for several years as a practical and powerful means to reveal secret keys of [publicly known] cryptographic algorithms. Only very recently this kind of cryptanalysis has been applied to reverse engineer a non-trivial part of the specification of a proprietary (i.e., secret) algorithm. The target here is no longer the value of secret key but the secret specifications of the cryptographic algorithm itself.

In a recent paper, Roman Novak (2003) describes how to recover the value of one (out of two) substitution table of a secret instance of the A3/A8 algorithm, the GSM authentication and session-key generation algorithm. His attack presents however two drawbacks from a practical viewpoint. First, in order to retrieve one substitution table (T_2), the attacker must know the value of the other substitution table (T_1). Second, the attacker must also know the value of secret key K.

In this paper, we improve Novak's attack and show how to retrieve *both* substitution tables (T_1 and T_2) *without any prior knowledge about the secret key.* Furthermore, as a side-effect, we also recover the value of the secret key.

With this contribution, we intend to present a practical SCARE (Side Channel Analysis for Reverse Engineering) attack, anticipate a growing interest for this new area of side-channel signal exploitation, and remind, if needed, that security cannot be achieved through obscurity alone.

Keywords: GSM Authentication, A3/A8, Reverse Engineering, Substitution Table, Side Channel Analysis

1 Introduction

Secure implementations of cryptographic algorithms on security devices such as smart-cards have been carefully studied, particularly since side-channel attacks were launched by P. Kocher [Koc96]. This kind of attacks derives information

about the execution of a sensitive algorithm, either from timing, power consumption or electromagnetic emanation measurements. The signal exploitation may range from simple observations, to more advanced statistical analyses. A simple observation allows distinguishing the rough structure of the algorithm —e.g. the number of round— or detecting the presence/absence of specific instructions or blocks of instructions. Statistical analyses come close to hypothesis testings, either by noise reduction averaging and enhancement of small signal contribution for differential techniques (DPA, DEMA) ([KJJ99,QS00]), or by more global and robust model fitting for correlation-based analyses (CPA, CEMA) [BCO03]. Though numerous variations of these cryptanalytic techniques have been proposed for several years, the target was inevitably the recovery of some sensitive data of a user (e.g., a private key).

Recently, Novak [Nov03] exploited side-channel leakage in order to obtain non-trivial details concerning the secret specifications of a block-cipher algorithm. The targeted algorithm was one of the many proprietary instances of the A3/A8 GSM authentication and session key generation algorithm. This opened a breach in a new kind of cryptanalytic attacks: the *Side-Channel Analysis for Reverse Engineering (SCARE) attacks*.

Without disclosing any further details about the targeted algorithm than those found in [Nov03], we present two practical improvements on Novak's attack. The attack described in [Nov03] recovers the entries of a secret substitution table from the knowledge of the other secret substitution table and the secret key. The SCARE attacks presented in this paper allow to recover the entries of the two secret substitution tables as well as the secret key from scratch.

Our attacks highlight that security cannot be achieved through obscurity alone. This is even more true because of the SCARE attacks. The security level of proprietary cryptographic algorithms (i.e., with secret specifications) is usually lower than publicly scrutinized algorithms. As SCARE attacks may allow to disclose the secret specifications (substitution tables in our case), those algorithms have then more chances to succumb to (classical) attacks.

The rest of this paper is organized as follows. In Section 2, we review the principles behind Novak's attack as well as its underlying assumptions. We then propose a graph interpretation of it, allowing a discussion on its theoretical feasibility. The next two sections describe our main contributions. Section 3 explains how to recover substitution table T_1 from the sole knowledge of secret key K and Section 4 explains how to get rid of the secret key. In Section 5, we discuss the threat of SCARE attacks and suggest practical counter-measures. Finally, we conclude in Section 6.

2 Retrieving Table T_2 with Known Key K and Known Table T_1

2.1 Description of Novak's attack

As for any GSM A3/A8 instance, the cryptographic algorithm attacked by Novak in [Nov03] takes a 16-byte challenge $M = (m_0, \ldots, m_{15})$ and a 16-byte secret

key $K = (k_0, \ldots, k_{15})$ on input, and produces a 32-bit message authentication code S_{RES} and a 64-bit voice ciphering session key K_C.

Novak's attack relies on three assumptions:

Assumption 1 (Observational). *The attacker is supposed to be able to detect by side-channel analysis when two intermediate values at some predefined point during the execution of the algorithm are the same.*

Assumption 2 (Prior structural knowledge). *The attacker is supposed to know the structure of the very beginning of the proposed target algorithm. Namely, he must know that the application of function f (as depicted in Fig. 1) to each one of the pairs $\{(m_i, k_i)\}_{i=0,\ldots,15}$ forms the very first operation performed onto the input data.*

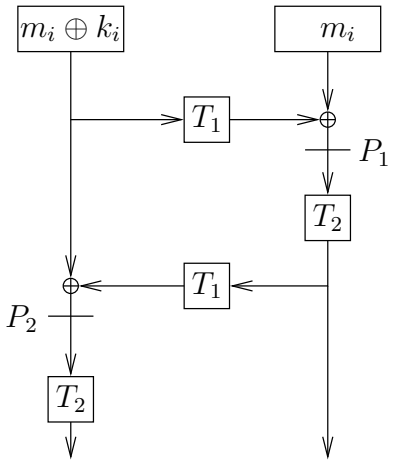

Fig. 1. Synopsis of function f

Note that in his goal to retrieve the value of the substitution table(s), the attacker does not need to know more about the structure of the algorithm. In particular, he needs not to know the number of rounds and (if applicable) the number of sub-rounds per round.[1] The attacker needs neither to known how the diffusion process in the algorithm is performed.

Assumption 3 (Prior data knowledge). *In order to retrieve table T_2, the attacker is supposed to know the whole content of table T_1 and secret key K.*

[1] A widely used and now public GSM algorithm, the COMP128, owns a round structure divided in 5 similar sub-rounds.

This last assumption is by far the main limitation in Novak's attack. In particular, it is hardly plausible that an attacker managed to get the content of T_1, and didn't know T_2. Furthermore, the required knowledge of the secret key constitutes an additional barrier.

In the sequel, *iteration* i refers to an application of f to a pair (m_i, k_i) in the first layer. An observation of two intermediate values for different iterations (resp. for the same iteration) is called a *cross-iteration* (resp. *intra-iteration*) observation.

Definition 1. *Let $\mathcal{R}^{(2)}_{\alpha,\beta}$ be the set of input pairs (m_α, m'_β) producing two identical values at point P_2 (cf. Fig. 1) for iterations α and β, namely*

$$\mathcal{R}^{(2)}_{\alpha,\beta} = \big\{(m_\alpha, m'_\beta) : T_1(T_2(T_1(m_\alpha \oplus k_\alpha) \oplus m_\alpha)) \oplus (m_\alpha \oplus k_\alpha) \\ = T_1(T_2(T_1(m'_\beta \oplus k_\beta) \oplus m'_\beta)) \oplus (m'_\beta \oplus k_\beta)\big\} \ .$$

By Assumption 1, it is possible to collect the set of pairs

$$\mathcal{R}^{(2)} = \bigcup_{\alpha,\beta} \mathcal{R}^{(2)}_{\alpha,\beta}$$

for which an intermediate equality occurs at point P_2 in the first layer.

We note that, by incrementing all message bytes in parallel, only 256 invocations of the algorithm are needed for that purpose.

Each one of these pairs relates two different entries of T_2 by the equation:

$$T_1(T_2(x_\alpha)) \oplus T_1(T_2(x'_\beta)) = d \tag{1}$$

with

$$\begin{cases} x_\alpha = T_1(m_\alpha \oplus k_\alpha) \oplus m_\alpha \\ x'_\beta = T_1(m'_\beta \oplus k_\beta) \oplus m'_\beta \\ d = (m_\alpha \oplus k_\alpha) \oplus (m'_\beta \oplus k_\beta) \end{cases} \tag{2}$$

where x_α, x'_β and d are known from Assumption 3.

Each of these relations gives the opportunity to link together (if not already done) two different T_2 entries. This decrements by one the degree of freedom (d.o.f.) of the table, that is, the number of remaining independent T_2 entries.

Once the d.o.f. of T_2 is reduced to 1, all T_2 entries are deductible from any of them, e.g. from $T_2(0)$. There so remain 256 possible T_2 candidates. The attacker is then able to identify the correct T_2 value by DPA-like [KJJ99] or CPA-like (correlation based) [BCO03] techniques.

Alternatively, and if he knows all other details of the algorithm, the attacker may also identify the correct T_2 value by a classical plaintext/ciphertext comparison. We stress that this knowledge is not mandatory for none of the attacks presented in this paper. Only the knowledge of the structure of the first layer (Assumption 2) is merely required to mount the attacks.

2.2 Graph interpretation

In this section, we propose an interpretation of the attack in terms of graph, discuss some implementation aspects, and present justifications for its theoretical feasibility.

Observing equal intermediates at point P_2 links together, by parameter d, two T_2 entries with indices x and x'. This basic relationship suggests a graph interpretation of the current knowledge that the attacker acquired so far about the constraints on T_2.

The graph of constraints on T_2 is a labelled non-oriented graph $\mathcal{G}^{(2)}$ whose vertices are the indices of the different T_2 entries, and where an edge labelled d between two vertices x and x' (noted $x \overset{d}{\frown} x'$) means that $T_2(x)$ and $T_2(x')$ are linked together by the relation:

$$T_1(T_2(x)) \oplus T_1(T_2(x')) = d \ . \tag{3}$$

At the beginning of the attack, graph $\mathcal{G}^{(2)}$ contains no edge, each vertex being apart from each other. This means that each entry is a priori independent of all others (except for the fact that they are all different from each other since T_2 is a permutation). There are 256 different connected components, each containing only one vertex. The d.o.f. of T_2 is then also equal to 256.

Each time a relation like Eq. (3) is to be exploited, the attacker connects vertices x and x' (if they were not) by a d-labelled edge. This results in a graph containing one less connected component. The correspondence between the number of independent entries in T_2 (the d.o.f.) and the number of connected components in $\mathcal{G}^{(2)}$ thus still persists.

Proposition 1 (Edge transitivity). *If* $x \overset{d_1}{\frown} x'$ *and* $x' \overset{d_2}{\frown} x''$ *then* $x \overset{d_1 \oplus d_2}{\frown} x''$.

Proof. Trivial. □

This proposition shows that it is possible to make each connected component of $\mathcal{G}^{(2)}$ fully connected.

2.2.1 Practical exploitation of observations

From a practical viewpoint, a possible and memory-efficient way to manage and maintain the information contained in $\mathcal{G}^{(2)}$ is to define two 256-byte arrays, say `comprep` and `delta`, such that:

- `comprep`$[x]$ represents the identifier of the connected component $comp(x)$ of x. By convention, it is defined as the least vertex belonging to $comp(x)$. This vertex may be thought of as the representative of $comp(x)$.
- `delta`$[x]$ represents the d parameter of the relation linking the vertex x and the representative of $comp(x)$ (`comprep`$[x]$). In particular,

$$\forall\, x \in \mathcal{G}^{(2)}, \texttt{delta}\big[\texttt{comprep}[x]\big] = 0 \ .$$

The exploitation process of the relations starts with `comprep` $= \{0, 1, \ldots, 255\}$ and `delta` $= \{0, 0, \ldots, 0\}$, meaning that each vertex forms a connected component by itself, of which it is obviously the representative.

Each time a relation is to be exploited, function `AddRelationToGraph`(x, x', d) defined in Fig. 2 is called, which possibly modifies the graph structure by merging together the connected components of x and x', if they were distinct.

Input: $\mathcal{G}$ given by `comprep` and `delta` arrays
(x, x', d) an observational $x \overset{d}{\frown} x'$ relation
Output: $\mathcal{G}$ with $comp(x)$ and $comp(x')$ merged together

if (`comprep`$[x] =$ `comprep`$[x']$) **then return** $\mathcal{G}$
if (`comprep`$[x] >$ `comprep`$[x']$) **then** `swap`(x, x')
for all $y \in comp(x') \setminus \{x'\}$ **do**
  `AddPointToGraph`$(x, y, d \oplus$ `delta`$[x'] \oplus$ `delta`$[y])$
endfor
`AddPointToGraph`(x, x', d)
return $\mathcal{G}$

Fig. 2. `AddRelationToGraph`(x, x', d) function

Input: $\mathcal{G}$ given by `comprep` and `delta` arrays
(x, y, d) a $x \overset{d}{\frown} y$ relation
Output: $\mathcal{G}$ with y added to $comp(x)$

if (`comprep`$[x] =$ `comprep`$[y]$) **then return** $\mathcal{G}$
`comprep`$[y] \leftarrow$ `comprep`$[x]$
`delta`$[y] \leftarrow$ `delta`$[x] \oplus d$
return $\mathcal{G}$

Fig. 3. `AddPointToGraph`(x, y, d) function

This operation must preserve both the convention that `comprep`$[x]$ is minimal in $comp(x)$, and the property induced by Proposition 1. It also ensures that all connected components are fully connected.

The process stops, either if there is no more relation to be exploited, or if the graph is fully connected which is detectable by the fact that `comprep` $= \{0, 0, \ldots, 0\}$. In this later case, `delta` contains all the information required to infer a possible candidate for T_2 from each possible value t_0 of its first element $T_2(0)$. The attack then succeeds.

Note that in the case of an unfinished attack, the number of possible candidates for T_2 rapidly grows with the number of remaining connected components in `comprep` (d.o.f.). This may become prohibitive if the number of remaining component is not small enough. This motivates a study of the connectivity of $\mathcal{G}^{(2)}$ after having exploited all relations.

2.2.2 Resulting connectivity of $\mathcal{G}^{(2)}$

We first evaluate the number of relations that may be collected for the attack when all possible message bytes are inputed at all possible iterations.

For any arbitrary secret key K, let

$$l_K = \#\{k_i\}$$

denote the number of distinct bytes of K. Let also $g(m,k)$ denote the value at point P_2 when m and k are the input bytes of function f, and

$$\mathcal{S}(z) = \{(m,k) : k = k_i \text{ for some } i, \text{ and } g(m,k) = z\} \ .$$

Finally, let $s(z) = \#\mathcal{S}(z)$.

To each pair of elements $\big((m_\alpha, k_\alpha), (m'_\beta, k_\beta)\big)$ of $\mathcal{S}(z)$ corresponds an edge (x_α, x'_β) to be added to the graph of constraints on T_2. The number of such edges for a given z is $\binom{s(z)}{2}$ but a lot of them can be deduced from others thanks to the transitivity property (Proposition 1). So, they are not to be counted as new relations. If we get rid of this transitivity property, the number of edges brought to the graph $\mathcal{G}^{(2)}$ by $\mathcal{S}(z)$ is $s(z) - 1$.[2] The total number of such edges amounts to

$$n_K = \sum_z (s(z) - 1) = 256 \cdot l_K - 256 = 256 \cdot (l_K - 1) \ .$$

Now, assuming that $g(m,k)$ behaves like a random function, the sets $\{x = T_1(m \oplus k) \oplus m : (m,k) \in \mathcal{S}(z)\}_z$ behave like random samples of vertices, and the evolution of $\mathcal{G}^{(2)}$ can be modeled as a random graph process.

This kind of structure and the evolution of its components have been deeply studied in graph theory. An asymptotic result by P. Erdős and A. Rnyi [ER60] states that a random graph with n vertices and $m \sim \frac{1}{2} n \log n$ random edges is almost certainly connected when $n \to \infty$.

For $n = 256$, the graph is connected once $m \approx 710$ edges are put on it.

Given that ([MOV97], p.53):

$$\Pr(l_K = t) = \left\{ {16 \atop t} \right\} \frac{\prod_{k=0}^{15}(256-k)}{256^{16}}$$

where $\left\{ {16 \atop t} \right\}$ denotes the Stirling number of the second kind, we have

$$\Pr(l_K \geqslant 13) = \Pr(n_K \geqslant 3072) \geqslant 0.999 \ .$$

[2] We neglect the very unlikely case when $s(z) = 0$.

This indicates that there are much more relations than needed for $\mathcal{G}^{(2)}$ to be connected. This has been confirmed by many simulations with random permutations T_1 and T_2. The exploitation of only intra-iteration relations always suffices to obtain a degree of freedom equal to 1.

3 Retrieving Table T_1 with Known Key K

The attack presented in [Nov03] assumes the ability to detect equalities of intermediate results at point P_2; the exploitation of such equalities making it possible to recover T_2.

Likewise, we show in this section that it is possible to recover T_1 by observing equalities of intermediate results at point P_1 (see Fig. 1).

Compared to Novak's attack, our attack relies on the same observational and structural prior knowledge assumptions (Assumptions 1 and 2). But the prior data knowledge assumption (Assumption 3) is weakened and replaced with the following one:

Assumption 3' ([Weakened] prior data knowledge). *In order to retrieve table T_1, the attacker is supposed to know secret key K.*

Definition 2. *Let $\mathcal{R}^{(1)}_{\alpha,\beta}$ be the set of input pairs (m_α, m'_β) producing two identical values at point P_1 (cf. Fig. 1) for iterations α and β.*

Similarly to § 2.1, each pair $(m_\alpha, m'_\beta) \in \mathcal{R}^{(1)}_{\alpha,\beta}$ relates two different entries of T_1 by the equation:

$$T_1(x_\alpha) \oplus T_1(x'_\beta) = d \tag{4}$$

with

$$\begin{cases} x_\alpha = m_\alpha \oplus k_\alpha \\ x'_\beta = m'_\beta \oplus k_\beta \\ d = m_\alpha \oplus m'_\beta \end{cases} . \tag{5}$$

Here again, parameters x_α, x'_β and d are known by the attacker (Assumption 3').

We should make a specific remark concerning this case:[3]

Proposition 2. *$\forall \alpha, \beta \in \{0, \ldots, 15\}$, we have $\mathcal{R}^{(1)}_{\beta,\beta} = \mathcal{R}^{(1)}_{\alpha,\alpha} \oplus (k_\alpha \oplus k_\beta)$.*

[3] By abuse of notations, for a vector $\boldsymbol{x}$ and a scalar δ, $\boldsymbol{x} \oplus \delta$ means that each component of $\boldsymbol{x}$ is $\oplus$-ed with δ, i.e., if $\boldsymbol{x} = (x_0, \ldots, x_t)$ then $\boldsymbol{x} \oplus \delta = (y_0, \ldots, y_t)$ with $y_i = x_i \oplus \delta$.

Proof. $\forall m, m' \in \{0, \ldots, 255\}$, we have

$$\begin{aligned}
(m, m') \in \mathcal{R}^{(1)}_{\alpha,\alpha} &\iff T_1(m \oplus k_\alpha) \oplus m = T_1(m' \oplus k_\alpha) \oplus m' \\
&\iff T_1\big((m \oplus (k_\alpha \oplus k_\beta)) \oplus k_\beta\big) \oplus (m \oplus (k_\alpha \oplus k_\beta)) \\
&\qquad = T_1\big((m' \oplus (k_\alpha \oplus k_\beta)) \oplus k_\beta\big) \oplus (m' \oplus (k_\alpha \oplus k_\beta)) \\
&\iff (m \oplus (k_\alpha \oplus k_\beta), m' \oplus (k_\alpha \oplus k_\beta)) \in \mathcal{R}^{(1)}_{\beta,\beta} \\
&\iff (m, m') \oplus (k_\alpha \oplus k_\beta) \in \mathcal{R}^{(1)}_{\beta,\beta} \ .
\end{aligned}$$

□

This implies for the attacker that information about T_1 brought by the relation set $\mathcal{R}^{(1)}_{\alpha,\alpha}$ is the same as the one brought by each other $\mathcal{R}^{(1)}_{\beta,\beta}$. Thus, it is worth exploiting only one of the 16 intra-iteration relation sets. Hopefully, such a remark does not apply to cross-iteration relation sets. Each one of the cross-iteration relation sets is a priori informative. Compared to the case where the attacker retrieves T_2 by observing at point P_2, the number of relation sets to be exploited is reduced from $16 + \binom{16}{2} = 136$ to $1 + \binom{16}{2} = 121$. This does not represent a noticeable penalty to mount the attack.

The exploitation process of the relations is the same as in Section 2.

The graph $\mathcal{G}^{(1)}$ of constraints on T_1 gathers all edges $x \overset{d}{\frown} x'$ where $T_1(x)$ and $T_1(x')$ are linked together by the relation:

$$T_1(x) \oplus T_1(x') = d \ . \tag{6}$$

The discussion about the connectivity of $\mathcal{G}^{(1)}$ is essentially the same as in § 2.2.2 —$g(m, k)$ being defined as the value at point P_1, and vertices x being equal to $m \oplus k$ instead of $T_1(m \oplus k) \oplus m$.

$\mathcal{G}^{(1)}$ is still modeled as a random graph, but there are slightly less available random edges due to Proposition 2. Nevertheless, simulations confirmed that this number of observed relations is, by far, large enough to mount the attack successfully.

4 Retrieving Table T_1 without Key K

In the case where he does not know secret key $K = (k_0, \ldots, k_{16})$, the attacker can make guesses about its successive bytes.

Making a guess g_0 about k_0, the attacker is able to exploit relations belonging to $\mathcal{R}^{(1)}_{0,0}$. More generally, making a guess $\boldsymbol{g_t} = (g_0, \ldots, g_{t-1})$ about the first t bytes $\boldsymbol{k_t} = (k_0, \ldots, k_{t-1})$ of the key, the attacker is able to exploit all relations in

$$\mathcal{R}^{(1)}_t \overset{\text{def}}{=} \bigcup_{0 \leqslant \alpha, \beta < t} \mathcal{R}^{(1)}_{\alpha,\beta} \ .$$

For any guess $\boldsymbol{g_t}$, let $\mathcal{G}^{(1)}(\boldsymbol{g_t})$ denote the graph of constraints on T_1 after having exploited all relations in $\mathcal{R}^{(1)}_t$, and assuming that $\boldsymbol{k_t} = \boldsymbol{g_t}$.

Graph $\mathcal{G}^{(1)}(\boldsymbol{g_t})$ is said to be *inconsistent* whenever the edge transitivity property (Proposition 1) is not verified; otherwise, it is said to be *consistent*. For any incorrect guess $\boldsymbol{g_t}$, the odds for graph $\mathcal{G}^{(1)}(\boldsymbol{g_t})$ to be inconsistent increase with t. This suggests an in-width first searching algorithm to retrieve T_1.

At depth t, a set $\mathcal{C}_t$ contains all guesses $\boldsymbol{g_t}$ (together with their corresponding graph $\mathcal{G}^{(1)}(\boldsymbol{g_t})$) for which $\mathcal{G}^{(1)}(\boldsymbol{g_t})$ is consistent. At depth $t+1$, when guessing byte k_t with each possible value g_t, each graph $\mathcal{G}^{(1)}(\boldsymbol{g_t})$ in $\mathcal{C}_t$ is constrained with all relations in $\mathcal{R}^{(1)}_{t+1} \setminus \mathcal{R}^{(1)}_t$. Each such updated graph $\mathcal{G}^{(1)}(\boldsymbol{g_{t+1}})$ is then stored (together with $\boldsymbol{g_{t+1}} = \boldsymbol{g_t} \cup \{g_t\}$) in $\mathcal{C}_{t+1}$, provided that it is consistent.

Before going further, we first give a slight generalization of Definition 2:

Definition 2'. *For any* $\boldsymbol{k} \stackrel{\text{def}}{=} \boldsymbol{k_{16}} = (k_0, \ldots, k_{15})$*, let* $\mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k})$ *be the set of input pairs* (m_α, m'_β) *producing two identical values at point* P_1 *for iterations* α *and* β*, when the secret key is* $\boldsymbol{k}$*.*

Proposition 3. $\forall \alpha, \beta \in \{0, \ldots, 15\}, \forall \delta \in \{0, \ldots, 255\}$*, we have*

$$\mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k} \oplus \delta) = \mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k}) \oplus \delta \ .$$

Proof. $\forall m, m' \in \{0, \ldots, 255\}$, we have

$$\begin{aligned}
(m, m') \in \mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k}) &\iff T_1(m \oplus k_\alpha) \oplus m = T_1(m' \oplus k_\beta) \oplus m' \\
&\iff T_1\big((m \oplus \delta) \oplus (k_\alpha \oplus \delta)\big) \oplus (m \oplus \delta) \\
&\qquad = T_1\big((m' \oplus \delta) \oplus (k_\beta \oplus \delta)\big) \oplus (m' \oplus \delta) \\
&\iff (m \oplus \delta, m' \oplus \delta) \in \mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k} \oplus \delta) \\
&\iff (m, m') \oplus \delta \in \mathcal{R}^{(1)}_{\alpha,\beta}(\boldsymbol{k} \oplus \delta) \ .
\end{aligned}$$

□

Given that each T_1 entry $x = m \oplus k$ depends linearly on m, Proposition 3 implies a kind of equivalence classes of pairs (table, key). For any δ, the same set of observed relations may suggest a given value for T_1 if the secret key is $\boldsymbol{k}$, as well as a table deduced from the previous one by $\oplus$-ing its indices with δ if the secret key is $\boldsymbol{k} \oplus \delta$. Otherwise stated, if the secret key $\boldsymbol{k}$ is compatible with observations then each secret key $\boldsymbol{k} \oplus \delta$ is also compatible.

The main implication is that exploiting equalities of intermediate values at point P_1 will, at best, disclose the value of T_1 up to its first element (as in the previously described attacks), but also up to a $\oplus$ of its indices by a constant δ.

Taking this property into account, the algorithm described above must be modified in that only one guess about k_0 (say $g_0 = 0$) needs to be considered. The rest of the algorithm remains unchanged.

Note that without any death of guesses which reveal their graph as inconsistent, the number of guesses to be considered would increase exponentially with the depth t. One may wonder whether this in-width search process indeed requires a prohibitive number of guesses $\boldsymbol{g_t}$ to be considered, or if incorrect

guesses prove themselves to be inconsistent so rapidly that the attack becomes practicable.

Here again, simulations showed that the recovery of the relative value of T_1 (up to a $\oplus$ of entries with $T_1(0)$, and up to a $\oplus$ of indices with $\delta = k_0$) by this in-width guessing process is actually effective. At depth $t = 2$, only few (say less than 20) incorrect guesses remain alive, and once $t = 3$ or 4 only the graph of the correct relative guess (up to k_0) usually remains consistent. From then, the relative value of T_1 is already known, but the attacker may choose to continue this process and exploit relations implying successive iterations in order to retrieve the remaining relative bytes of the secret key.

Finally, the relative values of T_1 and K are retrieved, and the attacker only needs to identify by DPA-like or CPA-like techniques (2^{16} candidates about T_1) which $(T_1(0), k_0)$ defines their correct absolute values.

We thus explained how an attacker may proceed to recover T_1 and K from no particular prior data knowledge. This step may then be followed by the basic Novak's attack in order to retrieve T_2 as well.

5 Counter-measures

By enhancing Novak's work, the attacks presented in the previous sections make possible to recover the two substitution tables of a secret algorithm. The exposure of such design details represents a threat at the *system level* —as opposed to the user level threat in a classical key recovery scenario. As the attacks need to be performed only once, the secrecy of the algorithm specifications directly relates to the protection offered by the *weakest available product* implementing this A3/A8 GSM algorithm.

Fortunately, there are counter-measures preventing our attacks. Side-channel leakage may be reduced via hardware features (including current scrambler or dual-rail logic). Time randomization may be introduced by hardware (e.g., dummy cycles) or software (e.g., random delays) means, making harder the comparison of waveforms at specific points. Finally, masking all intermediate values, which is the usual counter-measure against statistical analysis, should efficiently thwart our attacks, provided that the randomization is refreshed at every invocation. We point the synergy provided by the combination of these protections, each one making it difficult to bypass each other. A soon as such counter-measures are properly implemented, the observational assumption (Assumption 1) will not stand anymore, and the attacker will be defeated.

6 Conclusion

A SCARE attack presented in [Nov03] allows an attacker to recover the value of a substitution table T_2 which is part of the secret specifications of a GSM A3/A8 authentication and session key generation algorithm.

We proposed a graph interpretation of this attack and proved, under the random graph model, that the set of relations collectible by side-channel observation is large enough to infer the whole table up to its first element.

Noticing that this first attack needs the knowledge of another substitution table T_1 used in this algorithm as well as the knowledge of the secret key K (Assumption 3), we presented a similar way to retrieve T_1 from the sole knowledge of secret key K, and we then improved this attack to recover T_1 without even knowing secret key K, which is also recovered as a by-product.

Our proposed attacks have been validated by simulation. Providing that the observational assumption (Assumption 1) discussed in [Nov03], and a weak prior structural knowledge assumption (Assumption 2) are satisfied, our attacks allow to recover both substitution tables T_1 and T_2 (as well as secret key K for our last attack), without additional prior data knowledge.

We stress that, unlike classical attack scenarios in which the target is generally one's cryptographic secret key, SCARE attacks are *one-shot attacks* in that they jeopardize the specifications of the algorithm once for all. Should these specifications be made public, a further analysis by cryptography researchers may then reveal design flaws which will in turn threat all the users within the system. The security of a system being the one of its weakest link, this type of attacks demonstrates the need for a *generalization* of carefully designed implementations. Above all, it illustrates, one more time, the necessity to abandon the meaningless *security through obscurity alone* philosophy.

This contribution, together with [Nov03], opens new perspectives for side-channel analysis applied to reverse engineering.

References

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.

[BCO03] Éric Brier, Christophe Clavier, and Francis Olivier. Optimal statistical power analysis. Cryptology ePrint Archive, Report 2003/152, 2003.

[ER60] P. Erdős and A. Rnyi. On the evolution of random graphs, Magyar Tud. Akad. Mat. Kut. Int. Kzl. **5** (1960), 17–61.

[Koc96] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[MOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC Press, 1997.

[Nov03] Roman Novak. Side-Channel Attack on Substitution Blocks. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security*

(ANCS '03), volume 2846 of *Lecture Notes in Computer Science*, pages 307–318. Springer-Verlag, 2003.

[QS00] J-J. Quisquater and D. Samyde. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions, the SEMA and DEMA methods. Presented at the rump session of EUROCRYPT 2000, Bruges, Belgium, May 14–18, 2000.