Defensive Leakage Camouflage

E. Brier[1], Q. Fortier[2], R. Korkikian[3,4]
D. Naccache[2,4], G. Ozari de Almeida[3,4], A. Pommellet[2]
K. W. Magld[5], A. H. Ragab[5], and J. Vuillemin[2,5]

[1] Ingenico
1, rue Claude Chappe, BP 346, F-07503 Guilherand-Granges, France.
[2] École normale supérieure, Département d'informatique
45, rue d'Ulm, F-75230, Paris Cedex 05, France.
[3] Altis Semiconductor
224 Boulevard John Kennedy, F-91100, Corbeil-Essonnes, France.
[4] Sorbonne Universités – Université Paris II
12 place du Panthéon, F-75231, Paris Cedex 05, France.
[5] King Abdulaziz University, Jeddah, Saudi Arabia

Abstract. We consider the transfer of digital data over a *leaky* communication channel, that releases side-channel emissions and prevent the attacker from accurately measuring these emissions. The method pairs each secret key k with a camouflage value v and simultaneously transmits both k and v over the channel. This releases an emission $e(k, v)$. We wish to select the camouflage values $v(k)$ as a function of k in a way that makes the quantities $e(k, v(k))$ as *indistinguishable* as possible. We model the problem and show that optimal camouflage values can be effectively derived from a limited amount of *a priori* measures over emission traces (just as the attacker will do), under very weak physical assumptions. Consequently, the model is applicable across a wide range of readily available technologies.
We propose a statistical analysis of camouflage, in one, two and more dimensions. We discuss algorithms for inferring the best camouflage values from actual emission traces. Our algorithms are efficient for low dimensions (say up to 4) and heuristic beyond.
We provide some experimental results obtained on some memories, buses and IO emissions from other tamper-proof black-boxes.

1 Introduction

In addition to its usual complexity postulates, cryptography silently assumes that secrets can be physically protected in tamper-proof locations. All cryptographic operations are physical processes where data elements must be represented by physical quantities in physical structures. These physical quantities must be stored, sensed and combined by the elementary devices (gates) of any technology from which we build tamper-resistant machinery.

At any given point in the evolution of a technology, the smallest logic devices must have a definite physical extent, require a certain minimum time to perform their function and dissipate a minimal switching energy when transiting from one state to another. Energy is also dissipated statically, *i.e.* in the absence of any switching.

During the last twenty years, the research community devised a wealth of sophisticated methods for retrieving secret information from circuits by measuring their energy emanations.

In practice, most of the internal variables in current VLSI have an energy foot-print which is small enough to be invisible at the space/time resolution of all available energy measures. Yet, some other variables (*e.g.* capacitive internal bus, IO pad, ...) have an energy footprint

which is large enough for the corresponding signal to be externally measurable in a statistically reliable way. We refer the reader to [1] for a detailed presentation of the *statistical power model* applied here, and an analysis of classical counter-measures against side-channel attacks.

A number of authors, *e.g.* [1], rely on the switching-model where each bit of an isotropic channel dissipates the very same switching energy as any other. This work assumes no a-priori model (such as the switching-model) of the channel's emissions. Our *a posteriori* models exclusively result from actual measured emissions, with anisotropy.

1.1 Content

Unlike many previous works that analyzed the exploit leakage during complex cryptographic computations, we what is probably the simplest setting of all: the emanations from a passive channel through which bits are transmitted, and we only make two physical assumptions:

Leaky The side-channel emanations can be experimentally measured, from inside or outside the chip. We grant the attack and defense with the same measures, at equal resolution.

Small The spacial resolution of these measurements extends beyond the physical area of the channel. We thus forbid the attacker from individually probing any channel bits, or even from separating the channel's power footprint into lower and upper half-width.

Despite these assumptions, the proposed methodology remains applicable across a wide range of existing technologies, where two components communicate through a *leaky* bus.

The proposed defense pairs each secret key k with a camouflage value v and simultaneously transmit k and v through the channel. This releases a physical side-channel emanation $e(k, v)$ which can be measured as such by both the attacker and the defender.

We address the following question:

How can a defender pair each key k with a value v which makes $e(k, v)$ as *indistinguishable* as possible from $e(k', v')$, for all other keys k'?

The crux of this paper is the definition of *indistinguishability*, given the measured emission traces.

Section 3 uses statistical analysis to show that the best defense is to choose the camouflage values v gathering the emanations $e(k, v)$ from all keys k into the smallest ball spanning all keys/colors.

Before, section 2 exploits this geometrical characterization to extract optimal camouflage values from actual power traces. The algorithms presented compute the optimal efficiently in low dimensions, and through heuristics beyond.

Section 4 provides some experimental results and measures regarding the proposed defense strategies, including the key special case of a memory bus.

2 Models and Algorithms

We let e_d represent the physical emission (*e.g.* power consumption) which results from the transfer of data $d < N = 2^n$ through the n bits of a digital channel (*e.g.* a memory bus). We assume that e_d can be measured with equal precision by both the attacker and the defender.

The defender relies on first building a set (*i.e.* data-base, trace) of statistically meaningful emission measures $\mathcal{E}$. Each point $e \in \mathcal{E}$ is assigned to the data point $d = e \cdot d < N$ which contributes emission $e = e_d$.

Based on $\mathcal{E}$, the defense assigns s channel bits to the secret keys $k < S = 2^s$, and the remaining $n - s$ for the camouflage value $v_k < 2^{n-s}$.

We let $d(k, v) < N = 2^n$ represent the actual data, *i.e.* the bits of k and v_k mapped/interleaved onto the channel, and we note by $e(k, v) = e_{d(k,v)}$ an emission measured while transferring k and v through the channel.

The vector $V = [v_0 \cdots v_{S-1}]$ represents all camouflage values. It is chosen to make all emanations due to the secret bits look "as alike as possible". The choice is based on the actual measures $\mathcal{E}$.

We assign a unique *color* $e \cdot c = k$ to each point $e \in \mathcal{E}$, indexed by its key $k < V$. We view the trace $\mathcal{E}$ as a multi dimensional cloud of colored point.

A *color-spanning ball* is a subset $B \subset \mathcal{E}$ of the traces which contains at least one emission per color from each and every color.

The traces $\mathcal{E}$ upon which the defense is built cover all N data transfers. The traces $A(V)$ which the attack may build only cover the S pairs k, v_k, *i.e.* the color-spanning ball:

$$A(V) = \bigcup_{k<S} \{e \in \mathcal{E} : e \cdot d = d(k, v_k)\}.$$

The best strategy for defense is to minimize the *size* $\|A\|$ of the color-spanning ball $A = A(V)$ which remains exposed to attack.

Our aim is thus to compute/extract from $\mathcal{E}$ a *smallest color-spanning ball* S such that

$$\|S\| = \min_V \|A(V)\|,$$

i.e. S has the least size for all choices of V, and the absolute difference $|e - e'|$ between two scalar $e, e' \in \mathcal{E}$ is used as a similarity measure.

2.1 One Dimension

We assume here that e_d is a scalar - say the average power consumption carefully sampled after the clock edge during the transfer of data $d < N = 2^n$, and we measure the N reference traces:

$$\mathcal{E} = \{e_0 \cdots e_{N-1}\}.$$

Choosing the camouflage values $V = [v_0 \cdot v_{S-1}$ reduces the emissions to

$$A(V) = \{e(k, v_k) : k < S\},$$

and our goal is to minimize $\|A(V)\| = \max_k(e(k, v_k)) - \min_k(e(k, v_k))$.

Let $P = [p_0 \leq p_1 \leq \cdots \leq p_{N-1}]$ represent the sequence of emissions in $\mathcal{E}$, sorted by scalar value, and indexed by colors $p_i \cdot \mathbf{c}$.

A *color-spanning segment* is an interval in $[p_0; p_{N-1}]$ which contains at least one emission value of each color. We construct a *smallest* color-spanning segment in time proportional to the number N of measurements in the sorted sequence E.

Let S_i be the smallest color-spanning segment whose point of greatest value - that is, the rightmost extremity of the segment - is p_i. The smallest color spanning segment is clearly one of the S_i, for some value i. Thus we compute all the S_i, and output that of minimal length: it is the smallest color-spanning segment.

We define for each color c and index i

$$\text{last}_i(c) = \max_{\substack{c=p_j\cdot\mathbf{c} \\ j\leq i}} (p_j)\,,$$

i.e. the emission p_j of color c which closest to, and to the left $p_j \leq p_i$ of p_i - if any. We also consider $\text{first}(i) = \min_c(\text{last}_i(c))$.

When it is defined, segment $S_k = [\text{first}(i); p_i]$ spans all colors: by construction, the color $\text{first}(i)\cdot\mathbf{c}$ of its least value appears exactly once. It follows that there is no smaller color-spanning segment ending with p_i.

This allows the following linear time algorithm to compute every S_i, extract the minimal length, and return the smallest color-spanning segment.

Algorithm 1: Smallest color spanning segment algorithm

```
for i ∈ [0; N] do
    last[p[i] · c] ← i ;
    first ← min(last) ;
    if Every entry of last is properly defined then
        S[i] ← [first; p(i)] ;
        L[i] ← p(i).value − p(first).value ;
return S[argmin_k(L[i])]
```

Here S stands for the list of the computed segments S_i, and L for the length of these segments.

2.2 More Dimensions

We now consider the general case where e is a d-dimensional vector, *e.g.* a power consumption sample taken at d different clock cycles. The set $\mathcal{E}$ of traces is now a d-dimensional cloud of colored points, the color being defined by the key k. As a measure of similarity between a set of given points, the size of the smallest enclosing balls of these points makes sense. We must therefore determine the smallest ball containing at least one point of each color - that will, from now on, be called the smallest color-spanning ball.

As a finite set of points, our cloud is contained in a minimum enclosing d-dimensional rectangle R. We assume that the sides of this rectangle are parallel to the coordinate axis.

The general algorithm We can apply a divide and conquer method to the current problem. Let B be a ball of radius r such that B contains at least one point of each color (B is not necessarily optimal). If r is less than half the maximum length l of the sides of R, we split R along its longest side so that each sub-spaces has size $\frac{l}{2} + r$ on the coordinate corresponding to the maximum side of R, as described in Figure 1. Let R_1 and R_2 be the two equally sized sub-spaces obtained this way.

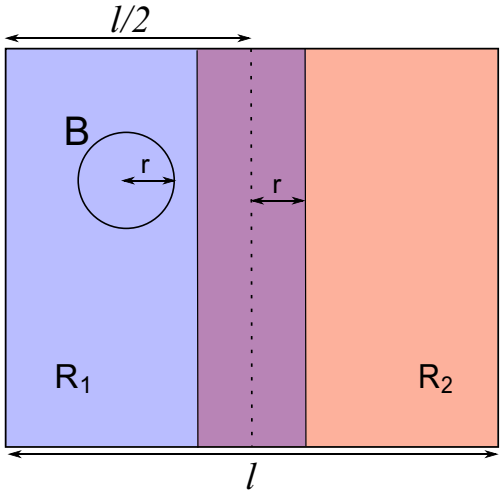

Fig. 1. Definition of the sub-rectangles

The smallest color-spanning ball is fully contained in R_1 or R_2, by construction, so that we can recursively find the smallest color-spanning balls in R_1 and R_2, and keep the smallest ball of the two. This solution will indeed be the smallest color-spanning ball of the original problem.

If the ball B is too large to split the space (for example, if every point is contained in R_1) we use a brute force method instead: if there are c colors we compute, for each c-tuple of points with different colors, the smallest ball containing these points. We then take the minimal ball. There are several methods to find such a ball: [2] describes a simple linear (in expectation) algorithm in two dimensions and Welzl ([3]) showed how to find it in all dimensions in linear time, if the dimension is fixed. However the complexity with respect to the dimension may be exponential.

Algorithm 2: Smallest color-spanning ball computation

Find a color-spanning ball B in the rectangle R with a heuristic method ;
Split R in two sub-rectangles R_1 and R_2 according to B ;
if R_1 *or* R_2 *contains every point* **then**
    Use a brute force method on the current rectangle R ;
else
    Compute recursively the respective smallest color spanning balls R_1 and R_2 of the two sub-rectangles R_1 and R_2 ;
    return $B' =$ *the smallest of the two color-spanning balls* R_1 *and* R_2

The main problem is the choice of B: we want it small enough to significantly reduce the search space of the divide and conquer method, especially for the final brute force, while being easy enough to compute. In the next subsection, we discuss the choice of a heuristic method to compute a small (but not always smallest) color-spanning ball.

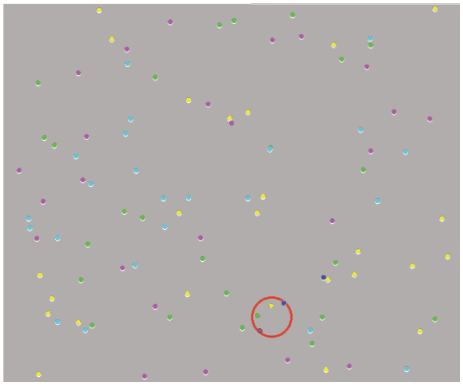

Fig. 2. Example of an output from the program, in 2 dimensions.

Heuristics and Methods We used the following method to find a first ball B: let p_1 be a point (for example the closest point to the center of R) with color 1 (assuming the colors are numbered from 1 to c). If we assume p_1, ..., p_k are properly defined, p_{k+1} is a point of color $k+1$ and with minimal distance to the barycenter of the previously computed points p_1, ..., p_k. If we assume that there are n points in each color, the complexity of this process is $O(n \times c)$. B is not necessarily optimal, as shown in Figure 3.

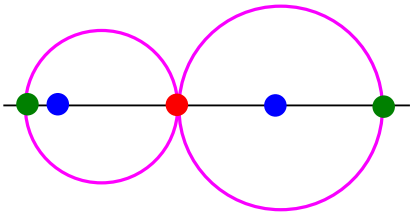

Fig. 3. The optimal ball (left) is different from the ball found by the heuristic (right) if the heuristic consider first the red color, then blue and finally green

2.3 Implementations

Algorithms were implemented in C++[6] in a straightforward manner. A function

```
bool smallest_ball(points, space, output)
```

splits space and points as explained above (using a ball found with `find_ball_barycenter`) and calls recursively `smallest_ball` on the smaller spaces, until this process stops to decrease

[6] the code is available at http://perso.ens-lyon.fr/quentin.fortier/color_ball.html

the size of the problem. We then use Miniball[7], a C++ software for computing smallest enclosing balls of points in arbitrary dimensions (without the need for a ball to contain at least one point of each color) in a brute force manner. The description of Miniball can be found in [4], and it is based on [3].

All presented timing results were measured on a (Dell inspiron 1520 with Intel Core 2 Duo T7300) Processor (2.0GHz, 4MB L2 cache, 2Go memory) with compilation achieved through Visual C++ 2008 (all optimization flags set for maximum speed).

Total number of points	2 colors	3 colors	4 colors	5 colors
100	11ms	39ms	309ms	2214ms
1000	164ms	1156ms	10s	147s
10000	2233ms	16s	160s	
100000	27s	188s	2240s (37min)	
1000000	287s	1937s (32min)	$>$ 1 hour	$>$ 1 hour

Table 1. Running time for points randomly chosen in the 4-dimensional unit cube, averaged over 10 runs

Total number of points	2 colors	3 colors	4 colors	5 colors
100	8ms	11ms	43ms	211ms
1000	96ms	221ms	833ms	7s
10000	946ms	3s	11s	81s
100000	10s	31s	145s	953s
1000000	109s	327s		

Table 2. Running time for points randomly chosen in the 3-dimensional unit cube, averaged over 10 runs

As shown in Table 1, in practice the algorithm is linear in the number of points and exponential in the number of colors.

3 Why Use Euclidean Distance?

Let $\{m_0(t), \ldots, m_{n-1}(t)\}$ be a database of n reference power consumption traces[8], the sample $m_{i,t} = m_i(t)$ corresponds to the power consumption at instant t caused by the manipulation of data element i. Let μ_t be the average power consumption at time t and σ_t the standard deviation at time t, *i.e.*:

$$\mu_t = \frac{1}{n}\sum_{i=0}^{n-1} m_{i,t} \qquad \sigma_t = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(m_{i,t} - \mu_t)^2}$$

[7] http://www.inf.ethz.ch/personal/gaertner/miniball.html
[8] defined over some discrete time interval $t \in [0, \ldots, \tau - 1]$.

Let a_t be an unidentified power measurement made by an attacker. The attacker's problem consists in finding the $m_{k,t}$ that *best reassembles* a_t. The following section will justify why, classically for doing so, an attacker would naturally compute for $0 \leq i \leq n-1$ the quantities:

$$\text{score}(i) = \sum_{t=0}^{\tau-1} \frac{(a_t - m_{i,t})^2}{\sigma_t^2} \tag{1}$$

and output the guess k corresponding to the $m_{k,t}$ whose score is the lowest *i.e.*:

$$\text{score}(k) = \min_{0 \leq i < n} \Big(\text{score}(i)\Big)$$

This formula is justified in the next section in the special case where the $m_{i,t}$ samples are t-wise independent.

In general, the samples can be correlated, for instance when the same secret bit is manipulated at two different instants. We analyze this general case later, and propose an explicit formula (2) for the score to minimize, which accounts for correlations between samples.

3.1 Multivariate Normal Distribution

Equation (1) stems from the assumption that, for any fixed i, successive measurements of $m_{i,t}$ follow an independent normal distribution with mean μ_t and standard deviation σ_t, and hence abide by the probability density function:

$$f_{m_t}(x) = \frac{1}{\sigma_t\sqrt{2\pi}} \exp\Big(-\frac{(x-\mu_t)^2}{2\sigma_t^2}\Big)$$

When the $m_{i,t}$'s are independently distributed, the probability density of all measurements m_t over $0 \leq t \leq \tau - 1$ can be expressed as a τ-dimensional multivariate distribution:

$$f_{\boldsymbol{m}}(\boldsymbol{x}) = \prod_{t=0}^{\tau-1} f_{m_t}(x_t) = \frac{1}{(2\pi)^{\tau/2} \prod_{t=0}^{\tau-1} \sigma_t} \exp\Big(-\sum_{t=0}^{\tau-1} \frac{(x_t-\mu_t)^2}{2\sigma_t^2}\Big)$$

where $\boldsymbol{x} = (x_0, \ldots, x_\tau)$.

Note that in the previous equation μ_t and σ_t are the expected value and standard deviation of $m_{i,t}$ over *all* data elements i. For a measurement $m_{i,t}$ corresponding to a specific data element i, we can also assume that $m_{i,t}$ follows a normal distribution with mean $\tilde{\mu_t} = m_{i,t}$ and standard deviation $\tilde{\sigma_t}$; we also assume that the standard deviation $\tilde{\sigma_t}$ around $m_{i,t}$ is the same for all data elements. In this case, the measurement m_t corresponding to data element i has the following distribution:

$$f_{\boldsymbol{m}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\tau/2} \prod_{t=0}^{\tau-1} \tilde{\sigma}_t} \exp\Big(-\sum_{t=0}^{\tau-1} \frac{(x_t-m_{i,t})^2}{2\tilde{\sigma}_t^2}\Big)$$

Additionally, we assume that the standard deviation $\tilde{\sigma}_t$ of m_t around $m_{i,t}$ is proportional to the standard deviation σ_t of m_t when all data values are considered, *i.e.* we assume $\tilde{\sigma}_t = \alpha \cdot \sigma_t$

for all $0 \leq t \leq \tau - 1$ for some $\alpha \in \mathbb{R}$. In this case, the probability density function of the m_t's for data i can be written as:

$$f_i(\boldsymbol{m}) = \frac{1}{(2\pi)^{\tau/2}\alpha^{\tau}\prod_{t=0}^{\tau-1}\sigma_t} \exp\Big(-\sum_{t=0}^{\tau-1}\frac{(m_t - m_{i,t})^2}{2\alpha^2\sigma_t^2}\Big) \;\propto\; \exp\Big(-\frac{\text{score}(i)}{2\alpha^2}\Big)$$

where score(i) is given by equation (1). The probability to obtain measurements m_t from data i is thus a decreasing function of score(i). Given measurement $\boldsymbol{m}$ the most probable candidate is therefore the one with the lowest score.

3.2 Multivariate Normal Distribution: Taking Correlation into Account

We denote by Σ the covariance matrix of the measurements, defined as follows:

$$\Sigma = \text{var}(\boldsymbol{m}) = \text{var}\begin{pmatrix} m_1 \\ \vdots \\ m_\tau \end{pmatrix} = \begin{pmatrix} \text{var}(m_1) & \text{cov}(m_1 m_2) & \cdots & \text{cov}(m_1 m_\tau) \\ \text{cov}(m_1 m_2) & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(m_1 m_\tau) & \cdots & \cdots & \text{var}(m_\tau) \end{pmatrix}$$

where $\text{cov}(X, Y) = \text{E}(XY) - \text{E}(X)\text{E}(Y)$ and $\text{var}(X) = \text{E}(X^2) - \text{E}(X)^2$.

We assume that the measurements follow a τ-dimensional multivariate distribution with mean $\boldsymbol{\mu}$ and covariance matrix Σ. The probability density function can then be expressed as:

$$f_{\boldsymbol{m}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\tau/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\Big)$$

where $|\Sigma|$ is the determinant of Σ. Note that mean $\boldsymbol{\mu}$ is a τ-vector and Σ is a $\tau \times \tau$-matrix.

Note that in the previous equation $\boldsymbol{\mu}$ and Σ are the expected value and covariance matrix of measurements for *all* data elements i. As previously, for measurements corresponding to a specific data element i, we also assume that the measurements follow a τ-multivariate normal distribution with mean $\tilde{\mu_t} = m_{i,t}$ and covariance matrix $\tilde{\Sigma}$;

We also assume that $\tilde{\Sigma}$ is the same for all data elements. In this case, the measurement $\boldsymbol{m}$ for data element i follows the multivariate distribution:

$$f_{\boldsymbol{m}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\tau/2}|\tilde{\Sigma}|^{1/2}} \exp\Big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{m_{i,.}})'\tilde{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{m_{i,.}})\Big)$$

As previously, we additionally assume that the covariance matrix satisfies $\tilde{\Sigma} = \alpha \cdot \Sigma$ for some $\alpha \in \mathbb{R}$. In this case, the probability density function can be written as:

$$f_{\boldsymbol{m}}(\boldsymbol{x}) = \frac{1}{(2\pi\alpha)^{\tau/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2\alpha}(\boldsymbol{x} - \boldsymbol{m_{i,.}})'\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{m_{i,.}})\Big)$$

which gives:

$$f_{\boldsymbol{m}}(x) = \frac{1}{(2\pi\alpha)^{\tau/2}|\Sigma|^{1/2}} \exp\Big(-\frac{\text{score}(i)}{2\alpha}\Big)$$

where:

$$\text{score}(i) = (\boldsymbol{m} - \boldsymbol{m_{i,.}})' \Sigma^{-1} (\boldsymbol{m} - \boldsymbol{m_{i,.}}) \tag{2}$$

Therefore we obtain that equation (2) is a generalization of equation (1) when taking correlations into account. In other words, to take correlations into account acquire a_t and compute for every i the score as per equation (2), sort the scores by increasing values and bet on the smallest.

Bivariate Example To illustrate the procedure consider the bivariate case where the covariance matrix between variables X and Y is:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$$

where $\text{var}(X) = \sigma_x^2$, $\text{var}(Y) = \sigma_y^2$ and $\text{cov}(X, Y) = \rho\sigma_x\sigma_y$ where ρ is the correlation between X and Y. In this case, we have:

$$\Sigma^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} \frac{1}{\sigma_x^2} & \frac{-\rho}{\sigma_x\sigma_y} \\ \frac{-\rho}{\sigma_x\sigma_y} & \frac{1}{\sigma_y^2} \end{bmatrix}$$

and the probability density function can be written:

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right]\right)$$

In this case, equation (2) gets simplified as follows:

$$s_i = \frac{(a_1 - m_{i,1})^2}{\sigma_1^2} + \frac{(a_2 - m_{i,2})^2}{\sigma_2^2} - \frac{2\rho(a_1 - m_{i,1})(a_2 - m_{i,2})}{\sigma_1\sigma_2}$$

where $\sigma_1 = \text{var}(m_1)$, $\sigma_2 = \text{var}(m_2)$ and ρ is the correlation between m_1 and m_2.

4 Experiments

Experimental results will be included in the final paper.

4.1 Acknowledgments

Part of this work was supported under grant 12-15-1432-HiCi from King Abdulaziz University.

References

1. E. Brier, C. Clavier, F. Olivier. Optimal Statistical Power Analysis. Cryptology ePrint Archive, Report 152, http://eprint.iacr.org/, 2003 .
2. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational Geometry: Algorithms and Applications. Springer, 1997
3. E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, New Results and New Trends in Computer Science, volume 555 of Lecture Notes in Computer Science, pages 359-370. Springer-Verlag, 1991
4. B. Gartner. Fast and robust smallest enclosing balls. In Proc. 7th Annual European Symposium on Algorithms (ESA), volume 1643 of Lecture Notes Comput. Sci., pages 325-338. Springer-Verlag, 1999