A Pre-processing Composition for Secret Key Recovery on Android Smartphone

Yuto Nakano, Youssef Souissi, Robert Nguyen, Laurent Sauvage, Jean-Luc Danger, Sylvain Guilley, Shinsaku Kiyomoto, Yutaka Miyake

▶ **To cite this version:**

Yuto Nakano, Youssef Souissi, Robert Nguyen, Laurent Sauvage, Jean-Luc Danger, et al.. A Pre-processing Composition for Secret Key Recovery on Android Smartphone. 8th IFIP International Workshop on Information Security Theory and Practice (WISTP), Jun 2014, Heraklion, Crete, Greece. pp.76-91, 10.1007/978-3-662-43826-8_6 . hal-01400921

HAL Id: hal-01400921

https://inria.hal.science/hal-01400921

Submitted on 22 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Pre-processing Composition for Secret Key Recovery on Android Smartphone

Yuto Nakano[1], Youssef Souissi[2], Robert Nguyen[2], Laurent Sauvage[2], Jean-Luc Danger[2], Sylvain Guilley[2], Shinsaku Kiyomoto[1], Yutaka Miyake[1]

[1] KDDI R&D Laboratories Inc.
2-1-15 Ohara, Fujimino, Saitama 356-8502, Japan
{yuto,kiyomoto,miyake}@kddilabs.jp
[2] Secure-IC S.A.S.
37-39, rue Dareau 75014 Paris, France
{youssef.souissi,robert.nguyen,laurent.sauvage,jean-luc.danger,sylvain.guilley}@secure-ic.com

Abstract. Simple Side-Channel Analyses (SSCA) are known as techniques to uncover a cryptographic secret from one single spied waveform. Up to now, these very powerful attacks have been illustrated on simple devices which leakage was obvious. On more advanced targets, such as high-end processors of smartphones, a simple visual analysis of the waveforms might not be sufficient to read the secret at once. In this paper, we detail and explain how a composition of time-frequency pre-processings manages to extract the relevant information from one signal capture of an asymmetric cryptographic operation (RSA and ECC) running on an Android system. The lesson is that side-channel countermeasures must be applied even on advanced platforms such as smartphones to prevent secret information theft through the electromagnetic (EM) waveforms.

Key words: Simple Side-channel Attack, Time-frequency Pre-processing, Asymmetric Cryptography, RSA, ECC, Android smartphone.

1 Introduction

Side-channel attacks (SCA) are becoming more and more serious threats to secure systems as the latter can be broken even if the underlying cryptographic algorithms are mathematically secure. In fact, these non-invasive attacks consist in exploiting the physical properties (e.g. electromagnetic, power, acoustic or time leakage) of a device when running some cryptographic process, in order to recover a sensitive information. The idea of power consumption based SCA was first introduced by Kocher [16]. Then, other side-channels have been exploited such as electromagnetic (EM) emanations [10], which are a more powerful source of leakage and can be performed without a physical contact to the device. Basically, SCA can be classified into *advanced* and *simple* attacks. Advanced SCA require a lot of waveforms to be statistically analysed. In some recent scientific papers (*e.g.* [3]), this kind of SCA is also called *vertical* attack

as the analysis focuses on the statistical behaviour of one (or some) time samples over many acquired waveforms. In spite of their effectiveness, in some real world situations, advanced SCA might not be applicable. As a matter of fact, the acquisition of many waveforms for the analysis is not possible because of the physical protections and software limitations (*e.g.* filters, sensors, number of trials to enter a password, etc) made to protect the target device. Besides, for some cryptographic RSA or ECC based protocols (*e.g.* key session generation like DH, RSA and ECDH protocols; or digital signature like ECDSA) or some protected implementations, the secret is generated only once. Thus, the attack must deal with only one single waveform. For this purpose, Simple SCA (SSCA) have been developed to work on such restrictive situations. SSCA are also called *horizontal* attacks as they exploit the information provided by the whole time samples within the waveform. Basically, the SSCA is usually performed over the time domain [25, 4, 14, 8, 16]. The frequency domain can be used to perform a noise-filter as a first step in the analysis in order to identify the secret patterns in the time domain as a second step. In the same context, it is shown that vertical SCA can be applied separately over each domain. As a matter of fact, Aboulkassimi et al. [1] proposed two different vertical SCA on PDA mobile device: the first attack, called Spectral Density Approach (SDA), aims at performing a vertical attack (correlation analysis) based on the power spectrum density representation (*i.e.* frequency domain) to overcome the problem of waveforms misalignment; and the second attack, called Template Resynchronisation Approach (TRA), consists in performing a correlation vertical attack over the time domain after resynchronising the waveforms.

From the signal processing theory viewpoint, the information content is the same in both domains, but its representation is not. This has a special flavour for side-channel analysis. In fact, the combination of both domains (*i.e.* time-frequency analysis) should provide better description of the secret leakage. More importantly, signal processing theory has provided us with powerful tools that can be used for this purpose, like the Short-Time Fourier Transform (STFT or spectrogram) and wavelet transforms. However, despite their great efficiency, the usage of these tools is not democratised for SCA. To our best knowledge, in the general context of SCA, very few scientific papers and reports [24] [11] [12] [22] have analysed the side-channel leakage from its time-frequency representation: in [24], Vuagnoux proposed to spy the electromagnetic (EM) activity of computer keyboards based on independent pre-processings. In fact, they used the STFT just as first and independent step to visualise the overall EM activity. The extraction of the secret itself is performed thanks to another tool that is a frequency filter applied directly on the non-noisy baseband waveforms. In addition to this, an anechoic room is used to obtain a good success rate. The time-frequency side-channel analysis has been also addressed by Gebotys et al. in [11]. They proposed a vertical attack, called DSA, on an AES-192bit (Advanced Encryption Standard) and ECC-192bit implementation-based PDA mobile device. Technically, the attack is mainly based on the computed spectrograms of acquired waveforms. However, a lot of EM waveforms (around 1100) are necessary to recover

the secret key. Similarly to [1], the main idea behind DSA is to get round the problem of misaligned waveforms. In [12], Genkin et al. have recently proposed an improved acoustic SCA on RSA-4096bit implementation, based on the spectrograms of acoustic leakages, emitted at very low frequencies. It is noteworthy that the proposed attack requires many acoustic waveforms, based on chosen ciphertexts, and follows several steps (*e.g.* waveforms classification, etc) to recover the RSA secret key. Besides, Souissi et al. in [22], have described several side-channel applications (vertical and horizontal SCA) based only on wavelet transforms. However, they briefly showed the power of such tool in SSCA context. Note that, in our paper, we will focus on the merits of Short-Time Fourier transform that is faster and easier to compute than wavelet transform.

Our contributions In this paper, our contributions are four-fold:

1. We propose a composition of a time-frequency SSCA to localise and characterise the secret leakage. By contrast to [11] [1], the proposed attack is **horizontal** and requires only **one single** EM measurement to recover the whole secret. Such attack is suitable for real implementations of asymmetric cryptography.
2. We show how classic SSCA, like Wiener and average mobile filtering, are unable to localise the secret patterns within a sample RSA waveform.
3. We perform the proposed SSCA on a modern device that is an Android smartphone clocked at 832 MHz. In this context, some publications pointed out the possibilities of SCA on smartphones and PDAs [11] [1] [15]. Kenworthy and Rohatgi [15] analysed side-channel vulnerabilities of RSA-Chinese Remainder Theorem (CRT) and elliptic curve point multiplication. They showed that the private key of both cryptosystems can be recovered from a single EM waveform using a specific equipment. Actually, an Icom IC-R7000 VHF-UHF receiver is used to demodulate the baseband waveform. The main issue, not addressed by the authors, is the way to find out the most appropriate frequency bandwidth related to the secret leakage. Moreover, note that this characterisation requires many measurements. On the other hand, the method we propose, instantaneously provides such characterisation, using software pre-processing on a single baseband (or raw) waveform. Our approach is more flexible as it is not limited by the range of frequencies to analyse that is necessary to be verified by the features of the demodulator equipment. Obviously, it is a real challenge to deal with such modern devices. Indeed, in the literature, we found that some studies (*e.g.* [26]) failed to mount a complete SSCA on smartphones.
4. We target asymmetric cryptographic algorithms. Indeed, we show how to break the security of basic RSA and ECC-based Android applications. More importantly, we note that we did not target home made applications. In fact, the targets are the cryptographic functions provided by JCE the default library of Android that is the most commonly used by the developers community.

2 A Pre-processing Composition for Secret Recovery

2.1 SSCA Types and Tools: Overview

In the literature, SSCA come basically with four types of analysis according to the implementation difficulty level, the used tools and the noise nature. Before going further, it is noteworthy that two kinds of noise can be considered: the measurement noise that is caused by surrounded electronic components and external environment; and the algorithmic noise that is generated by the whole activity of the target circuit except of course the one that is related to the secret. Now, the four types of SSCA can be described as follows:

1. *SSCA first type*: it is the simplest one as it is based only on a visual detection of the secret patterns. Such type of analysis is possible only when the useful information is not perturbed by the noise [16]. It is generally applied on fully controlled platforms often used for demonstration or academic purposes.
2. *SSCA second type*: it involves the usage of pre-processing tools. This SSCA basically deals with the measurement noise in order to make easier the visual detection of secret patterns. In the real world (*i.e.* when performing the attack on real devices), acquired waveform is usually noisy. Hence, pre-processing is essential. We note that the core idea behind this paper mainly turns around this type of SSCA. In what follows, we list commonly used techniques:
   - Frequency filtering (High & Low pass filters),
   - Time linear & non-linear filtering (e.g. Wiener or Kalman filtering [23]),
   - High order statistics [17],
   - Threshold-based Wavelets transforms [7] [22].

   Generally, when acquired EM waveforms are very noisy, the extraction of the secret key is a challenging task as the sensitive leakage is not properly localised. Indeed, in such situation, those techniques usually fail to recover the secret.
3. *SSCA third type*: it statistically analyses (*e.g.* linear correlation, time samples combination, collision analysis, etc) the whole time information within the waveform in order to extract the secret [4] [14] [8]. Such SSCA type is useful to efficiently target some protected implementations like blinding RSA.
4. *SSCA fourth type*: it combines both pre-processing and statistical analysis. These SSCA are generally used to deal with noisy and protected real world implementations.

2.2 Theoretical Background: DFT, STFT and Noise-filtering

In this section, we recall some theoretical definitions and basics necessary to properly understand the proposed attack. From the signal processing viewpoint, the acquired EM waveform can be represented as a sequence of N discrete time samples. We denote by $\{y[0], y[1], ..., y[N-1]\}$ such sequence. Hence, the discrete

Fourier transform (DFT) returns N frequency components that can be written as follows (Eq.1):

$$Y[k] = \sum_{n=0}^{N-1} y[n]\, \omega_N^{kn}, \tag{1}$$

where $\omega_N = e^{-j\frac{2\pi}{N}} = \cos(\frac{2\pi}{N}) - j\sin(\frac{2\pi}{N})$ and $k = \{0, 1, ..., N-1\}$ is the frequency index. For the sake of clarity, the DFT can be represented in a matrix form such that (Eq.2):

$$\begin{bmatrix} Y[0] \\ Y[1] \\ Y[2] \\ \vdots \\ Y[k] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_N^1 & \omega_N^2 & \cdots & \omega_N^{(N-1)} \\ 1 & \omega_N^2 & \omega_N^4 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{(N-1)} & \omega_N^{2(N-1)} & \cdots & \omega_N^{(N-1)^2} \end{bmatrix} \times \begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ \vdots \\ y[k] \end{bmatrix}. \tag{2}$$

According to Eq.2, the DFT allows us to characterise the periodicity of elementary activities within the analysed sequence. Indeed, from the SSCA point of view, the idea being that the waveform is exhaustively compared to different sinusoidal waves. In practice, DFT can be very useful in telling us about the number of most significant patterns in the EM waveform. However, the DFT is not able to detect time varying activities as it provides an averaged value of the frequency over the whole time samples in the sequence. For this purpose, the Short-Time Fourier Transform (STFT) proposes to apply the DFT over a windowed version of the sequence. Indeed, the sequence is segmented into successive time intervals (windows) of fixed length $L \leq N$. Then, the DFT is computed over each window. This way the frequency information is time-localised. Consequently, the STFT can be expressed as follows:

$$Y[k, t_0] = \sum_{n=0}^{L-1} y[t_0 + n] \cdot s[n] \cdot e^{-j\frac{2\pi kn}{L}}, \tag{3}$$

where, k is the frequency index, s is the sliding window and t_0 is the time reference index related to s. It is noteworthy that, the window size determines the time and frequency resolution. As the window size is fixed, a good frequency resolution is obtained to the detriment of a good time resolution, especially when the used window is large. In practice, we slide the window by one time sample to enhance the time resolution. Eventually, the STFT can be seen as a two-dimensional representation showing the information over both, the time and the frequency scales. This way, the STFT allows differentiating the elementary activities of the target component. This means that the STFT is able to deal with the algorithmic noise.

Now, in order to deal with the measurement noise, any basic noise-filtering (*e.g.* mobile average smoother, Wiener filter, Kalman filter, low pass filter, linear regression, etc) can be used. In our attack, we use the simplest noise-filter that is the mobile average. Let l be the window size of such filter. Thus the filter simply

consists in computing the arithmetic mean of all subsequent time intervals, each of length l.

2.3 Composition Scheme and Steps

Based on the previous definitions, we introduce our proposed SSCA tool that performs the analysis based on a composition scheme. We remind the reader that our attack requires only one single EM waveform. Basically, as shown in Fig. 1, the composition scheme is composed of 3 chained pre-processing blocks:

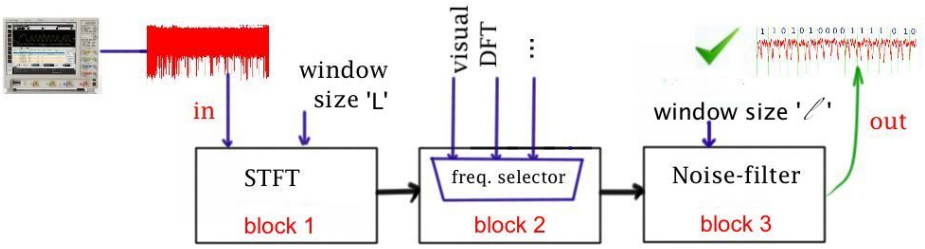

Fig. 1. An illustration of the pre-processing composition scheme.

1. **Block 1: Time-frequency localisation** The first step consists in computing the STFT over the baseband waveform. In the SSCA context, the usage of this tool is justified for multiple reasons:
   - The sliding STFT allows the analysis to keep the **time** component of the signal. As the SSCA is based on demarcation of the sequential pattern, the study of time is mandatory.
   - The **frequency** component created by STFT allows the analysis to forge a local frequency signature. This signature is very relevant as the processes are always performed in a circular manner, with periodic loops. This is true at both application level, especially in cryptography, or at implementation level, especially in software. Inside a loop the activity pattern is specific, hence allowing to discriminate the pattern for SSCA.

   This way, as stated before, the algorithmic noise would be properly analysed.
2. **Block 2: Frequency bandwidth selection** The second step aims at selecting the frequency area of interest, where the secret leakage exists. This can be performed through a simple visual analysis, especially when the internal activities had been efficiently decorrelated by the STFT. Alternatively, advanced techniques like DFT, clustering or probability density estimation can be used. The idea being that we should able to start characterising the secret patterns (*e.g.* periodicity, frequency, classes, shape, etc).
3. **Block 3: Noise-filtering** The third step, definitely removes the measurement noise. Therefore, the full characterisation of the secret patterns becomes an easy task.

3 Real Case Study: RSA / ECC Key Recovery on Android Smartphone

3.1 SSCA on Android Environment: Key Points

In the following, we discuss the key points to properly mount an SSCA on Android smartphone: first, we describe the measurement requirements to properly access the Android smartphone; second we tackle the most important difficulties which arise when targeting such device; and third we outline the possible software modifications of the Android system that might be performed by an attacker to enhance his malicious analysis.

Access to the Smartphone Basically, the SSCA measurement bench is composed of a PC-Target system (or client-server system) and a measurement device (an oscilloscope and an antenna) to record EM waveforms. Concerning the minimum features needed in the oscilloscope, four points should be taken into consideration: the sampling frequency, the memory depth, the placement of antennas and the trigger which provides a reference timing point necessary to synchronise waveforms. Most of recent smartphones are equipped with high speed processors to offer higher performance. For instance, the default CPU clock frequency of the handset we use is set to 832 MHz. This means we need an oscilloscope which is capable to sample at least 3 Gsample/s (i.e., 3×10^9 samples per seconds) in order to get an acceptable EM measurement for SSCA. Additionally, note that increasing the sampling frequency might create a problem with a memory depth. Besides, the attacker is also conducted to identify which type of antenna should be used and how to place it. In practice, the localisation of the secret leakage can be simply performed by sweeping the antenna all over the backside of the smartphone. The most appropriate location can be precisely recovered by performing an EM cartography. The EM cartography consists of two steps: observing the EM field while sweeping the antenna over the device and then mapping the EM level to each position. The higher the EM level is in the cartography, the more the leakage is at the corresponding position.

Now, we consider the last important factor: the trigger signal which is necessary to identify the start/end of the target process. This also allows aligning acquired EM traces. Regarding SSCA, as only a single waveform is required for the analysis, the trigger is less critical than for vertical SCA that require perfectly aligned waveforms. In our study, we will show how the first block in the proposed composition scheme (*i.e.* the STFT) can be an efficient solution for triggering. In the case of smartphones, the existing problem relies on the way the trigger is generated. Unlike testing boards (*e.g.* the SCA-Standard Evaluation Board (SASEBO) [21]), smartphones do not have I/O ports for sending/receiving control signals. Aboulkassimi et al. [1] proposed to generate a pulse trigger based on read/write queries sent to the SD card system. In our experiments we realised that such method is fast and efficient to localise in time the target cryptographic process. Moreover, we tested another solution which is less invasive and easier to mount in real world attack situations. In fact, it consists in spying in real time the activity of running processes, and activating a serial trigger (through

the USB connector) only when the target application is called by the system. More precisely, when the flag associated with the target application is written in the Android log file system that can be checked using Android ADB tool [2].

Factors That Can Make SSCA Difficult

1. **Just In Time Compiler**
   The Just In Time Compile (JIT) aims to compile on-the-fly the Android bytecode into native machine instructions. This significantly enhances the performance of the Dalvik Virtual Machine that is in charge of executing Android applications. From the side-channel point of view, the activity of the JIT compiler might slightly influence the analysis as reported by Aboulkassimi et al. in [1]. In practice, the impact of the JIT can be seen through the first executions of an iterative process. For instance, in the case of a basic RSA process, the JIT activity may impact the first patterns related to the secret key bits manipulation. As only the first patterns are affected, this does not harm the overall efficiency of the SCA. Besides, even if some few secret bits are guessed incorrectly, some techniques [5] allows recovering the entire secret key.
2. **Garbage Collector**
   The Garbage Collector (GC) can be defined as the process in charge of cleaning the memory (e.g. heap) by deleting unused Java objects. It is basically activated at random times by the processor when running user or Android system applications. This might create a problem for SSCA as it is not trivial to deal with the undesired activity generated by the GC. The GC is normally deactivated during the execution of native elementary processes. For instance, we observed that OpenSSL native cryptographic library is never perturbed by the GC. This is not the case for pure non-native libraries. However, in this case, when the GC is called, it is still easy to characterise its activity and remove it especially when the EM antenna or probe is properly placed. During our experiments, we also noticed that when using hybrid libraries[3] the GC might be called by the system, but will never impact (or interrupt) the native calls.
3. **Multi-Core Processor** In order to provide higher performances for users, handsets with multi-core processors are now the mainstream of smartphones. Unlike the single-core processors, multi-core processors distribute tasks to other existing cores so that they can be efficiently completed. In fact, during the cryptographic operation, the process might be moved from one core to another due to system interruptions. It might be also executed in parallel by several cores for the sake of performance. Therefore, this might rise EM acquisition problems. In practice, to deal with issue, one may acquire the waveforms as following two ways: either try to stay focused on the core

[3] A hybrid library, like the JCE default Android library, is basically designed with a non-native language that makes recurrent calls to native primitives, like for instance to the modular arithmetic functions of OpenSSL.

executing the targeted implementation by placing a tiny antenna over the right decoupling capacitor; or capture the whole activity with a single and large antenna placed over the target processor. In our case, this did not impact our analysis as we targeted a mono-core processor based smartphone.

Software Modifications for Better EM Acquisition

1. **Under-clocking the CPU** The higher the CPU clock frequency is, the higher the resolution of the oscilloscope is required. In case that the clock frequency is too high and the oscilloscope is unable to acquire enough samples to cover the necessary leakage, the attacker can decrease the CPU clock frequency for a better acquisition. Basically, lowering clock frequency is not always necessary. In fact, in our experiments we kept the original CPU clock frequency of 832 MHz.
2. **Lowering Radio Emission** When many applications are running on the Android smartphone, in particular that ones which are responsible for ensuring radio and cell communications, the noise generated might have a considerable impact on the effectiveness of SSCA. In such situation, the attacker may reduce the level of noise by turning on the airplane mode available on most smartphones. We note that, in our experiments we show how our SSCA is still efficient even if the airplane mode is turned off.

3.2 Target Cryptographic Android Implementations

There are several cryptographic libraries compatible with the Android environment such as Java Crypto Extension (JCE) [19], OpenSSL [18], Bouncy Castle [6], Crypto++ [9], RELIC [20] and so on. At first we intended to exhaustively evaluate the security of all libraries when RSA and ECC are being executed. Then we realised that most of libraries, particularly the non-native ones, are mainly based on the same low level native primitives to perform arithmetic calculations on big integers, necessary for asymmetric cryptography. This can be verified by checking the source code of libraries or more easily by analysing the real time processor activity through the debugging Trace View Android tool [13]. We note that the comparison of these libraries is out of the scope of this paper.

Now, for the sake of generality and clarity, we will focus only on the most commonly used library that is the default JCE Android library. Actually, JCE provides the essential cryptographic primitives to manage security within an application. Its basic API packages are *java.math*, *javax.crypto* and *java.security*. Note that these APIs call the arithmetic primitives of the *OpenSSL* native library, such as modular squaring and multiplying, that have been already ported to Android system. Moreover, these APIs include a lightweight version of *Bouncy Castle* library that provides ready high level functions to execute, for instance, an elliptic curve point multiplication.

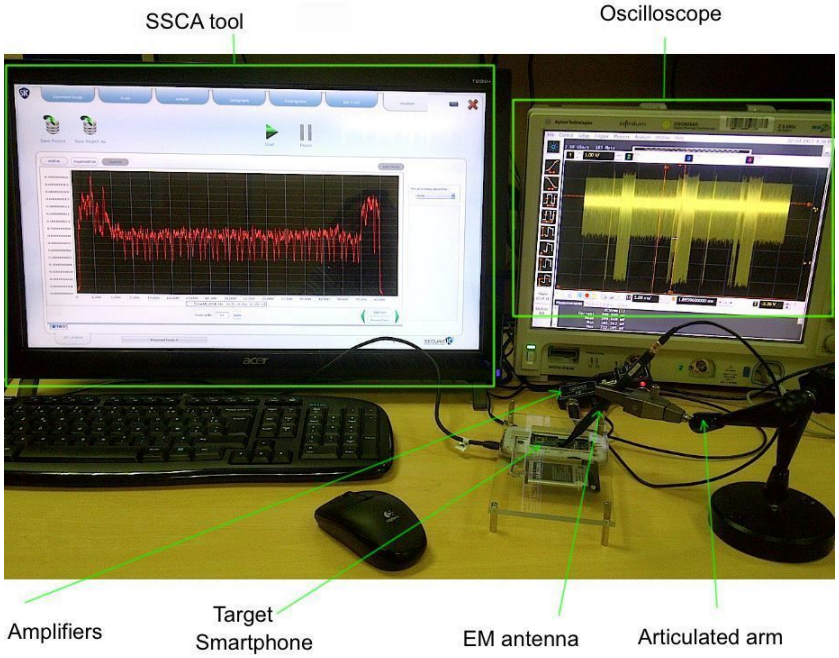

Fig. 2. An illustration of the measurement bench.

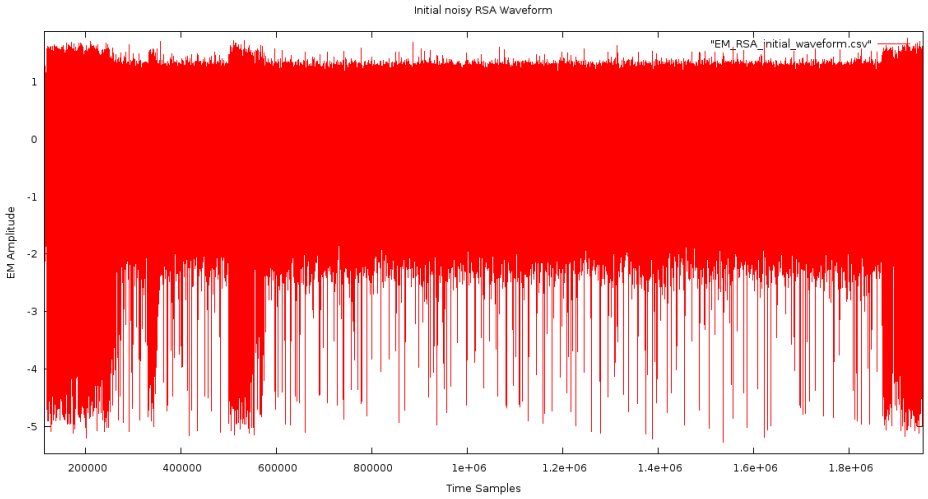

Fig. 3. Initial noisy RSA EM waveform.

3.3 Experiments and Results

SSCA Experimental Environment Our measurement bench is illustrated in Fig. 2. Basically, it is composed of an Agilent Infiniium 9000 oscilloscope

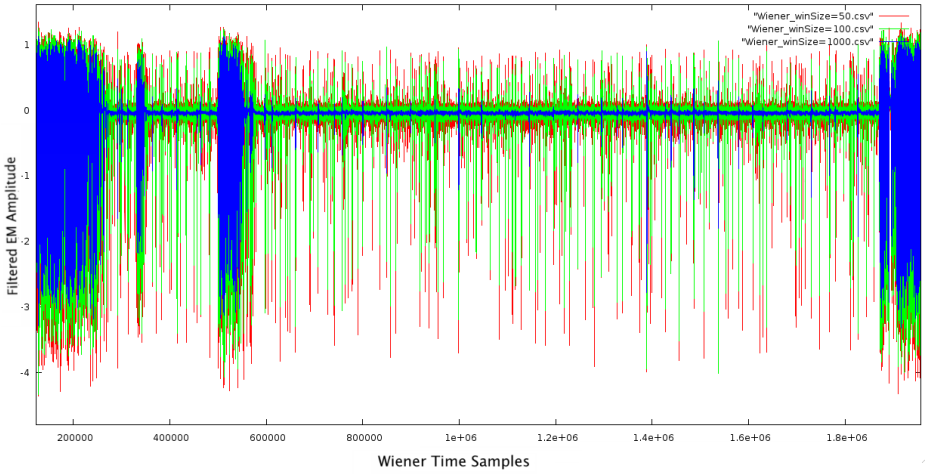

Fig. 4. Failure of Wiener filter to recover the RSA key.

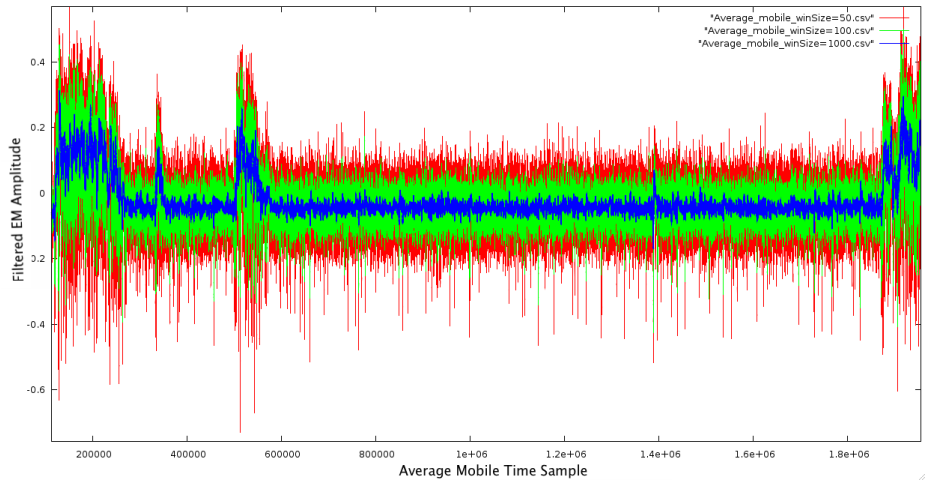

Fig. 5. Failure of average mobile smoother to recover the RSA key.

with a bandwidth of 2.5 GHz, a maximal memory depth of 123 mega samples, a maximal sample rate of 20 giga samples per seconds; antenna of the HZ-15 kit from Rohde & Schwarz and a 30 dB amplifier. The communication between our SSCA tool, referred to as *Smart-SIC Analyzer*[4], and the smartphone is ensured by the TCP/IP protocol. Note that the attack can be performed over different ways of communication (i.e. USB, Wifi, etc).

[4] `http://secure-ic.com/smart-sic-analyzer`

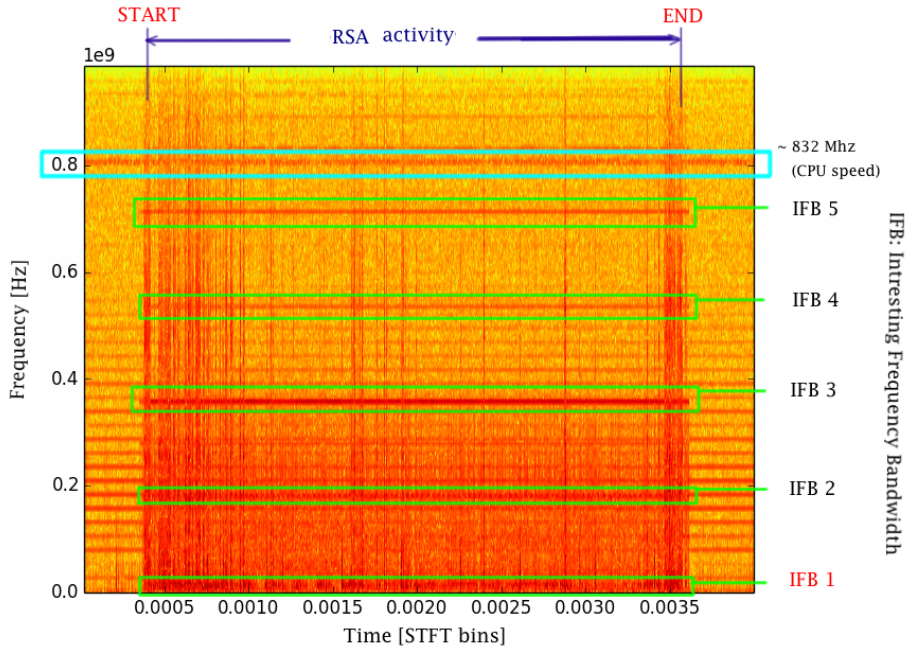

Fig. 6. STFT based 2D-localisation of the RSA activity.

RSA Key Recovery The targeted RSA implementation is the right-to-left binary exponentiation variant: the sequence of operations (*i.e.* multiply and square) are dependent on the private key. Therefore, from the SSCA viewpoint, the idea being that we could identify the multiply-square pattern when the corresponding bit of the private key is '1', and only square pattern when the corresponding bit of the key is '0'. The RSA baseband waveform as it is acquired by the oscilloscope is shown in Fig. 3. We note that for the sake of clarity, in this example we have chosen a small RSA key that is `0xAF0AE3` (hexadecimal form). Clearly, the RSA activity is totally hidden by the noise, which makes classic SSCA inefficient. Indeed, in our experiments, we tested the most commonly used techniques (Wiener filtering and Average mobile smoothing) to extract the useful leakage from such noisy waveform. The results are illustrated by Fig. 4 and Fig. 5. We exhaustively tested many window sizes (*i.e.* the input parameter l). For the sake of convenience only the results for window sizes 50, 100 and 1000 are shown in both figures. Obviously, the localisation of the secret patterns is not possible when applying these techniques directly on the baseband waveform. Indeed, these techniques are usually used to remove only the measurement (or Gaussian) noise which is not sufficient in our case, as we should deal also with the algorithmic noise. In what follows, we will show the efficiency of our proposed

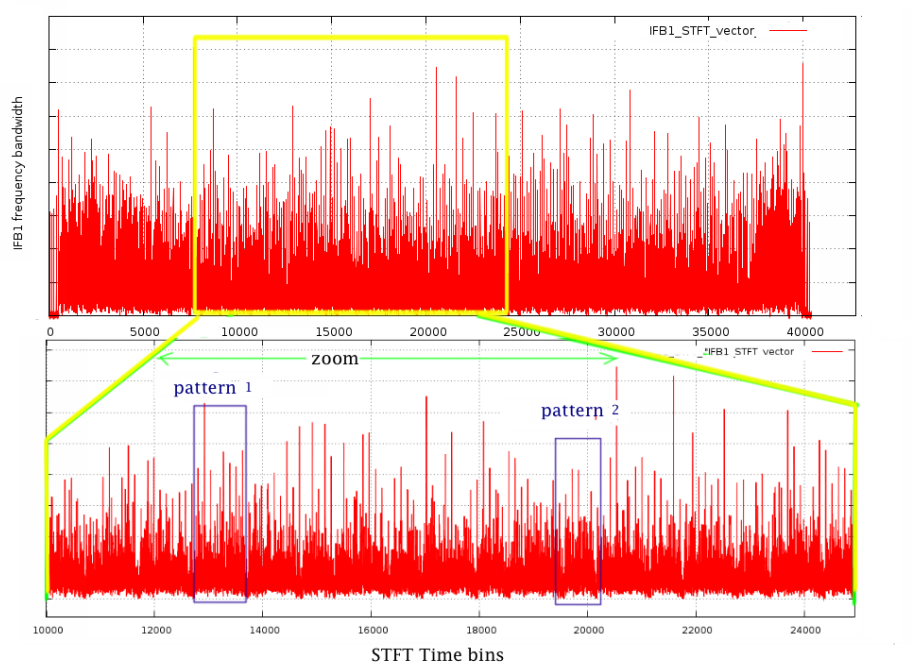

Fig. 7. IFB1 vector when extracted from the STFT matrix.

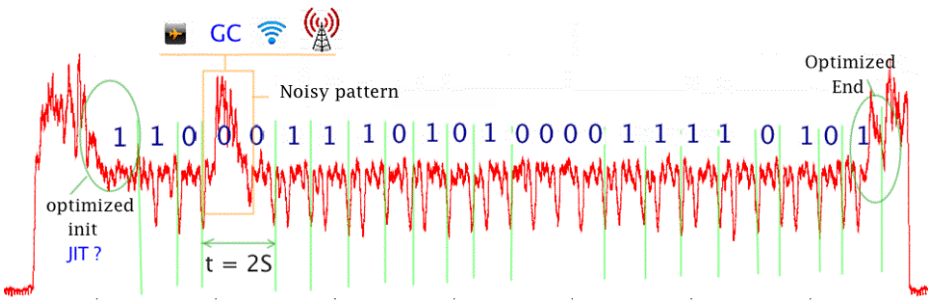

Fig. 8. Key recovery of RSA right-to-left binary exponentiation.

method when applied on the same EM waveform. The Fig. 6 is an illustration of the STFT when applied on the RSA baseband waveform. Three important points can be directly deduced from this representation:

- The time-localisation of the RSA activity (start/end time identification).
- The recovery of the CPU activity around 832 MHz.

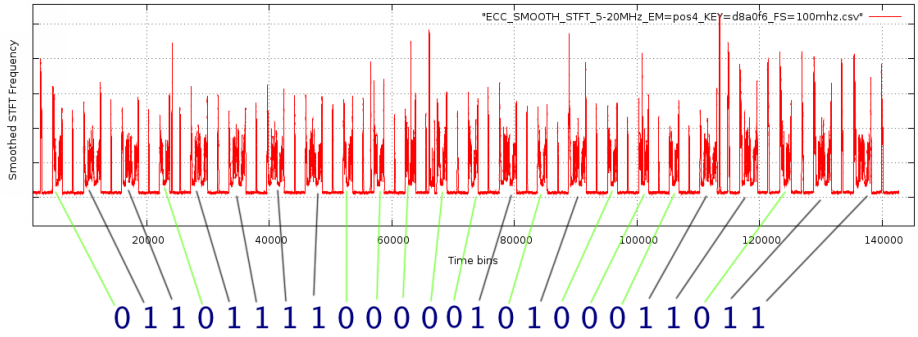

Fig. 9. An illustration of smoothed ECC activity.

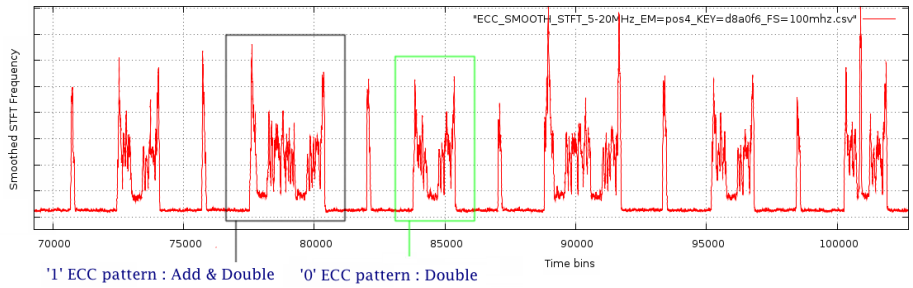

Fig. 10. Zoom on smoothed ECC activity.

– The frequency-localisation of five interesting activities (i.e. the most leaking sources) located into different frequency bandwidths, termed by Interesting Frequency Bandwidth (IFB) in the figure. Numerically, IFBs are float vectors that can be extracted from the STFT matrix to be analysed.

After having visually analysed the five IFB vectors, we noticed that IFB1 vector ([10 MHz–40 MHz]) is likely to reveal the secret RSA patterns. The extracted IFB1 vector is illustrated by Fig. 7. Interestingly, it is possible to identify a very noisy sequence of patterns located in the same time interval of the RSA process. More precisely, two forms of patterns can be revealed and that are likely to be related to the activity of Square and Multiply RSA operations. Now when applying a basic smoother, like the mobile average smoothing filter, the secret RSA patterns are clearly identified as shown in Fig. 8. Besides, we noticed the presence of some noisy patterns, which are likely to be caused by the JIT or the cellular communications.

ECC Key Recovery Our target here is a basic ECC implementation, more precisely its elliptic curve point multiplication operation. The challenge is to

characterise the ECC activity knowing that, iteratively, when the manipulated secret bit is '0' then only one elementary operation, that is *Point doubling* is performed; otherwise an additional operation, that is *Point adding*, is involved. Therefore, from the SSCA viewpoint one expect the identification of two different patterns. Similarly to RSA, we realised that, the most interesting STFT frequency bandwidth is represented by lower frequencies (IFB1 vector) located between 10 MHz and 20 MHz. The secret bits (`0xD8A0F6`) are entirely revealed when an average mobile smoothing filter is performed on the analysed IFB1 vector (Fig. 9). The Fig. 10 is a zoom on the resulting waveform. Obviously, the two ECC secret patterns can be easily differentiated based on their shape and their execution time.

4 Conclusion and Perspectives

In this paper, we have proposed an efficient pre-processing composition to mount a powerful SSCA on RSA and ECC. We applied the proposed attack to recover the secret keys on an Android smartphone clocked at high frequency (832 MHz). We remind the reader that, in practice, the higher the frequency is, the harder to attack as more noise is generated and more sophisticated equipments are needed. Our scheme succeeded in recovering the secret keys from a single waveform. Therefore, we conclude that our technique is particularly efficient to perform the pattern discrimination as it deals with both types of noise (measurement and algorithmic noise) and both domain representations (time and frequency). The proposed attack is applied directly on baseband traces. Hence, we expect to further enhance our analysis with a Software-Defined Radio (SDR) demodulator. Future work will be applying our attack to the devices with the CPUs of higher frequency and multi cores.

References

1. D. Aboulkassimi, M. Agoyan, L. Freund, J. Fournier, B. Robisson, and A. Tria. ElectroMagnetic analysis (EMA) of software AES on Java mobile phones. In *WIFS*, pages 1–6. IEEE, 2011.
2. Android Debug Bridge. available at `http://developer.android.com/tools/help/adb.html`.
3. A. Bauer, É. Jaulmes, E. Prouff, and J. Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. In E. Dawson editor, *CT-RSA*, volume 7779 of *LNCS*, pages 1–17, Springer, 2013.
4. A. Bauer, E. Prouff, É. Jaulmes and J. Wild. Horizontal Collision Correlation Attack on Elliptic Curves. In T. Lange, K. Lauter, and P. Lisoněk editors, *SAC*, volume 8282 of *LNCS*, Springer, 2014.
5. D. Boneh, G. Durfee, and Y. Frankel. An Attack on RSA Given a Small Fraction of the Private Key Bits. In K. Ohta and D. Pei editors, *ASIACRYPT*, volume 1514 of *LNCS*, pages 25–34, Springer, 1998.
6. Bouncy Castle project. Bouncy Castle Crypto APIs. available at `http://www.bouncycastle.org/documentation.html`.

7. X. Charvet and H. Pelletier. Improving the DPA Attack using Wavelet Transform. *Physical Security Testing Workshop* available at `http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper14.pdf`
8. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. *IACR Cryptology ePrint Archive*, Report 2010/394, 2010. `http://eprint.iacr.org/2010/394`.
9. Crypto++ Library. available at `http://www.cryptopp.com/`.
10. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.
11. C. H. Gebotys, S. Ho and C. C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In J. R. Rao and B. Sunar, editors, *CHES*, volume 3659 of *LNCS*, pages 250–264. Springer, 2005.
12. D. Genkin, A. Shamir, and E. Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. *Cryptology ePrint Archive*, Report 2013/857, 2013.
13. Google Inc. Profiling with `Traceview` and `dmtracedump`. available at `http://developer.android.com/tools/debugging/debugging-tracing.html`.
14. J. Heyszl, A. Ibing, S. Mangard, F. D. Santis, and G. Sigl. Clustering Algorithms for Non-Profiled Single-Execution Attacks on Exponentiations. *IACR Cryptology ePrint Archive*, Report 2013/438, 2013. `http://eprint.iacr.org/2013/438`.
15. G. Kenworthy and P. Rohatgi. Mobile Device Security: The case for side channel resistance. available at `http://mostconf.org/2012/papers/21.pdf`.
16. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
17. T. H. Le, J. Clédiere, C. Serviere, and J. L. Lacoume, Noise Reduction in Side Channel Attack Using Fourth-Order Cumulant. *IEEE Transactions on Information Forensics and Security*, (4):710–720.
18. OpenSSL Project. OpenSSL library documentation. available at `http://www.openssl.org/related/binaries.html`.
19. Oracle Corporation. JAVA JCE documentation. available at `http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html`.
20. RELIC library (UNICAMP). available at `https://code.google.com/p/relic-toolkit/`.
21. Research Center for Information Security (RCIS). Side-channel Attack Standard Evaluation Board (SASEBO). available at `http://www.rcis.aist.go.jp/special/SASEBO/index-en.html`.
22. Y. Souissi, A. E. Aabid, N. Debande, S. Guilley, and J.-L. Danger. Novel Applications of Wavelet Transforms based Side-Channel Analysis. *Non-Invasive Attack Testing Workshop*, available at `http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/01_Souissi.pdf`, 2011.
23. Y. Souissi, S. Guilley, J.-L. Danger, S. Mekki, and G. Duc. Improvement of power analysis attacks using Kalman filter. In *ICASSP*, pages 1778–1781. IEEE, 2010.
24. M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pages 1–16, Berkeley, CA, USA, 2009. USENIX Association.
25. C. D. Walter. Sliding Windows Succumbs to Big Mac Attack. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *LNCS*, pages 286–299. Springer, 2001.
26. C. Zenger, C. Paar, K. Lemke-Rust, T. Kasper and D. Oswald. SEMA of RSA on a Smartphone. B.Sc. (from 01/03/2011 to 17/10/2011) report. available at `http://www.yumpu.com/en/document/view/19636241/sema-of-rsa-on-a-smartphone`.