Test Vector Leakage Assessment (TVLA) methodology in practice

G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi and S. Saab

Cryptography Research Inc.

1. Introduction

Many security standards require cryptographic devices and modules to resist side-channel attacks such as Timing Analysis as well as Simple and Differential Power/Electromagnetic Analysis. These requirements have also been included in the draft FIPS 140-3 standard [1]. However, existing security certification standards mandating side-channel resistance, such as Common Criterion, require an evaluation style testing approach to verify compliance. Such evaluation style testing approaches are not suitable for a conformance style testing program such as CMVP, and effective, yet cost-efficient, conformance style testing for side-channel resistance has been seen as a challenge.

At the NIST sponsored NIAT workshop in 2011, a promising new approach for performing conformance style testing for side-channel resistance was presented for the AES and RSA algorithms [2,3]. The approach was based on measuring power consumption from devices while performing cryptographic operations with a *pre-specified* set of input test vectors and then performing a set of *pre-specified* statistical tests on the collected power measurements. These statistical tests yield confidence scores, using which a clear fail/pass criterion can be established. These papers also presented preliminary experimental results on different devices to establish that these tests could detect sensitive information leakage within the power side-channel, within a reasonably short amount of time and without requiring test operators to become skilled in performing the latest side-channel attacks.

In this paper, we describe our experiences in using and enhancing this methodology which we have renamed as “Test Vector Leakage Assessment”, as part of our analysis flow for performing side-channel assessment of tens of cryptographic devices, as well as part of our design flow for creating side-channel resistant implementations, over the past two years.

Our experience has confirmed that this methodology is a reliable, quick and easy test for detecting potential side-channel problems with devices. Often our analysis requires us to perform both a TVLA assessment to detect leakage and then perform a key extraction attack to verify the seriousness of that leakage. It has been our observation that performing TVLA testing for leakage is quicker by one to two orders or magnitude compared to performing key extraction attacks. The TVLA based testing methodology also lends itself to real-time testing: the tests can be performed as the measurements are being collected. Throughout this paper, we will describe several examples where TVLA testing time was substantially quicker than performing key extraction attacks, and also provide a comparison of the

performance of TVLA with the recently published work on testing protected AES implementations on Sasebo-R using the (key dependent) chosen input method [5].

Our experience has also shown that the non-specific tests of leakage presented in [2,3], i.e., those which compare measurements when the device is repeatedly performing operations with a fixed input vs. measurements when the device is performing operations with random inputs, to be the most powerful. These tests cover a wide variety of leakages and can often detect problems with a device using an order of magnitude fewer measurement traces than tests targeting specific classes of leakage. Non-specific tests for AES and RSA have therefore become the first analysis that we perform while data is being collected from a device, so that problems can be detected early in the process. Based on our experience with a variety of platforms, including mobile devices, we have further enhanced and optimized these non-specific tests in two different ways.

The first enhancement dramatically speeds up the data collection and testing process for AES, by one or more orders of magnitude. It has been found to be most useful for mobile platforms and for designer's seeking quick feedback on their DPA-resistant design in software or on an FPGA. The improved approach works as follows: The device or implementation is configured to perform bulk AES encryption. For AES-CBC mode, the input data is adjusted so that the input to the AES operation after the chaining is always the same as the specified test vector. With this approach, a single trace collected from the sampling oscilloscope can yield several hundred/thousand AES operations performed during a single bulk operation. Software is then used to split the single trace into the individual AES encryption segments for performing the TVLA testing. With this approach it is possible to collect many orders of magnitude more traces in the same amount of data collect time, and analysis involving millions of AES operations, e.g., to validate a masked AES implementation on an FPGA, can be carried out within a day. This approach also works quite well for mobile and software platforms where interrupts or other interference can seriously degrade the parsing of the individual AESs encryption segment in the signal. For the non-specific, fixed-vs.-random test, there is no need to precisely map which AES signal corresponds to which random (or fixed block) in the presence of these interrupts and interference! The test can be done using only those AES signal segments which are clearly identified to be from either the fixed set or from the random set. In contrast, the non-specific tests or attacks require substantial signal processing to identify which signal segment corresponds to which encryption operation, and this can easily take an order of magnitude extra effort and time.

The second enhancement to the non-specific, fixed-vs.-random testing is targeted to devices where there is substantial usage of timing noise and amplitude noise countermeasures, and the precise start and end of the AES operations cannot be determined from the signal. In such cases, it is reasonable to expect the vendor to provide a trigger at the initiation of the AES command or when the timing/amplitude noise generation is started during the AES command processing. The fixed-vs.-random test however, requires the test operator to ignore the start and end of AES and focus only on the middle third of the operation, as the fixed input and output processing from the fixed data set can have a different side-channel profile from the processing of random inputs and outputs without necessarily creating a key extraction vulnerability. However, with timing/amplitude noise, the middle third of the AES may not be determinable from the traces or from the trigger signal. To handle these and other cases,

we have modified the fixed-vs.-random test to create semi-fixed set of test vectors for AES where the input and outputs are statistically random, but significant parts of one or more interior rounds of AES is kept fixed. These semi-fixed test vectors can be used in place of the fixed test vector for the non-specific fixed-vs.-random set, without impacting the effectiveness of the test or identifying the middle third of AES.

The paper is organized as follows: Section 2 provides a background on TVLA with a focus on tests for AES. Section 3, illustrates how TVLA testing can be sped-up using bulk AES encryption modes. Section 4 compares the efficacy of our TVLA methodology with the (known key) chosen input proposal [5] on the protected AES implementations on Sasebo-R [4]. Finally, Section 5 describes the design of test-vectors that can be used to test for non-specific leakages in AES implementations where it is not feasible to demarcate the beginning and end of individual AES operations.

2. Background on TVLA testing for AES

As described in [1], TVLA testing for AES requires the following:

1. AES device with side-channel test fixture.
2. Ability to invoke AES encryption/decryption operation on the device with a chosen key, and chosen plaintext/ciphertext.
3. Trigger signal to demarcate the beginning and end of AES operation
4. Equipment to collect side-channel traces

The tester selects an appropriate value of n, the parameter that decides the number of traces used for the individual tests for leakage. The choice of n depends on the desired level of security, with larger values of n corresponding to higher levels of resistance.

The tester must collect side-channel traces for two data sets DATA-SET-1 and DATA-SET-2 from the AES cryptographic block encryption with a specific, published key and a set of data as follows:

DATA-SET 1:

a. Key K is set to
    i. 0x0123456789abcdef123456789abcdef0 for AES-128
    ii. 0x0123456789abcdef123456789abcdef023456789abcdef01 for AES-192
    iii. 0x0123456789abcdef123456789abcdef023456789abcdef013456789abcdef012 for AES-256
b. Perform 2n encryptions with inputs: $I_0, I_1, \ldots, I_{2n-1}$, here I_0 = 0x00000000000000000000000000000000 (16 0 bytes) and $I_{j+1} = AES(K, I_j)$ for $0 \leq j < 2n$.

DATA-SET 2:

a. Set key K as in data set 1

b. Input data J is set to
    i. 0xda39a3ee5e6b4b0d3255bfef95601890 for AES-128
    ii. 0xda39a3ee5e6b4b0d3255bfef95601888 for AES-192
    iii. 0xda39a3ee5e6b4b0d3255bfef95601895 for AES-256
c. Perform n encryptions with input J

From a practical perspective, the fixed input DATA-SET 2 traces *MUST* be randomly interspersed between the collection of DATA-SET 1 traces. Both DATA-SET 1 and DATA-SET 2 require the entire AES operation to be measured and recorded. All the traces are aligned to the AES start trigger signal (or to an AES start feature) and the following tests are performed.

2.1 Non-specific, Fixed-vs.-Random Test

Two independent t-tests are performed with disjoint sets of traces as follows:

1. Test 1: t-test comparison between first n/2 traces from DATA-SET 1 vs. first n/2 traces from DATA-SET 2.
2. Test 2: t-test comparison between last n/2 traces from DATA-SET 1 vs. last n/2 traces from DATA-SET 2

If both Test 1 and Test 2 show a t-score of < -4.5 or > 4.5 **at that same point in time during the middle third of the AES operation,** the device fails.

2.2 Specific leakage tests

The tester chooses an arbitrary AES middle round M. The following 896 t-tests are conducted on two independent groups of traces.
Group 1: Traces 1 through n from DATA-SET 1
Group 2: Traces n+1 through 2n from DATA-SET 1

List of t-tests:

1. (128 Tests) RIRO_M_bit_0 through RIRO_M_bit_127 : For each trace t, let RIRO_M(t) denote the EXOR of Round M input with Round M output for trace t. For each bit i from 0 to 127, the test RIRO_M_bit_i is a t-test that compares the subset of traces t, where bit i of RIRO_M(t) is 0 vs. traces t where bit i of RIRO_M(t) is 1.
2. (128 Tests) Sout_M_bit_0 through Sout_M_bit_127: For each trace t, let Sout_M(t) = Concatenated output of the 16 Sbox table lookups in Round M for trace t. For each bit i, from 0 to 127, the test Sout_M_bit_i is a t-test that compares the subset of traces t where bit i of Sout_M(t) is 0 vs. traces t where bit i of Sout_M(t) is 1.
3. (128 Tests) Rout_M_bit_0 through Rout_M_bit_127: For each trace t, let Rout_M(t) denote the output of Round M for trace t. For each bit i, from 0 to 127, the test Rout_M_bit_i is a t-test that compares the subset of traces t where bit i of Rout_M(t) is 0 vs. traces t where bit i of Rout_M(t) is 1.
4. (256 Tests) Rout_M_byte_0_is_0 through Rout_M_byte_0_is_255: For each trace t, let Rout_M_byte_0(t) denote the value of the first byte of output of Round M for that trace. For each value b of a byte from 0 to 255, the test Rout_M_byte_0_is_b is a t-test that compares the subset of traces t where Rout_M_byte_0(t) is b vs. traces t where Rout_M_byte_0(t) ≠b.
5. (256 Tests) Rout_M_byte_1_is_0 through Rout_M_byte_1_is_255: For each trace t, let Rout_M_byte_1(t) denote the value of the second byte of output of Round M for that trace. For each value b of a byte from

0 to 255, the test Rout_M_byte_1_is_b is a t-test that compares the subset of traces t where Rout_M_byte_1(t) is b vs. traces t where Rout_M_byte_1(t) ≠b.

If any of these 896 tests show a t-score of < -4.5 or > 4.5 **at that same point in time** for both Group 1 and Group 2 data-sets, the device fails.

3. Rapid TVLA testing using bulk AES

As described in Section 2, having a large number (n) of encryption traces is critical for testing to high degree of resistance. While modern cryptographic devices can perform AES encryptions at a very high rate, capturing and storing side-channel traces from individual AES operations using standard, commercially available measurement equipment such as sampling oscilloscopes can be quite slow – most scopes can capture and store no more than a few tens of traces per second. For tests requiring tens of millions of AES traces, for example, for high security applications or when designing DPA resistant AES designs on FPGAs, this limitation can become a serious bottleneck.

For such applications, we recommend the use of bulk AES encryption modes to substantially speed up the process of performing TVLA testing. Most high speed AES engines provide an option of encrypting bulk data using the ECB and/or CBC modes. Using a bulk encryption mode to encrypt a few hundred kilobytes of data, can yield a single side-channel trace consisting of several tens of thousands of AES operations. The overheads associated with capturing and storing the trace using a standard scope are amortized over these tens of thousands of AES operations to yield much better performance. Separating the signal segments corresponding to individual AES operations within a bulk encryption trace can be performed very efficiently using software. For example, if a scope can collect 30 individual AES traces/s, but takes a minute to collect and store a single trace consisting of 10,000 AES operations, collecting traces for 10 million AES operations, one at a time would take over 90 hours, whereas collecting the same number of AES operation traces using a bulk encryption mode will take only 17 hours. A data collection time of 90 hours may be acceptable for a testing lab to evaluate a product, but may be unacceptably long to incorporate into the protected AES design/test cycle.

The following example based on the AIST's AES implementation on the standard SASEBO G-II board [4] illustrates how such a setup works. Using a PC and a Tektronix DPO 7104 scope, one can capture up to 20 power traces of AES operations/s. Figure 1 shows a single trace captured from the scope. Capturing 60, 000 operations which are sufficient to cause TVLA test failures for both the specific and non-specific AES tests takes 50 minutes. (While the non-specific fixed-vs.-random test failure only requires 100 traces, non-specific tests require many more traces). However by modifying the default AES implementation on SASEBO G-II to perform AES-CBC bulk encryption using plaintext stored within the FPGA's block RAM, we were able to collect a power trace for encrypting 160Kb of chosen data in around 15 seconds. Figure 2 shows the captured trace from an AES-CBC operation, with the individual AES operations clearly visible. Software was used to find the time offsets for each of the AES operations. Using the approach the entire data collection time is reduced to 45 seconds, and the entire set of TVLA

tests could be performed within a couple of minutes. In fact, for this example, full key extraction is possible using 60,000 distinct AES operations collected from two sets of AES-CBC operations.

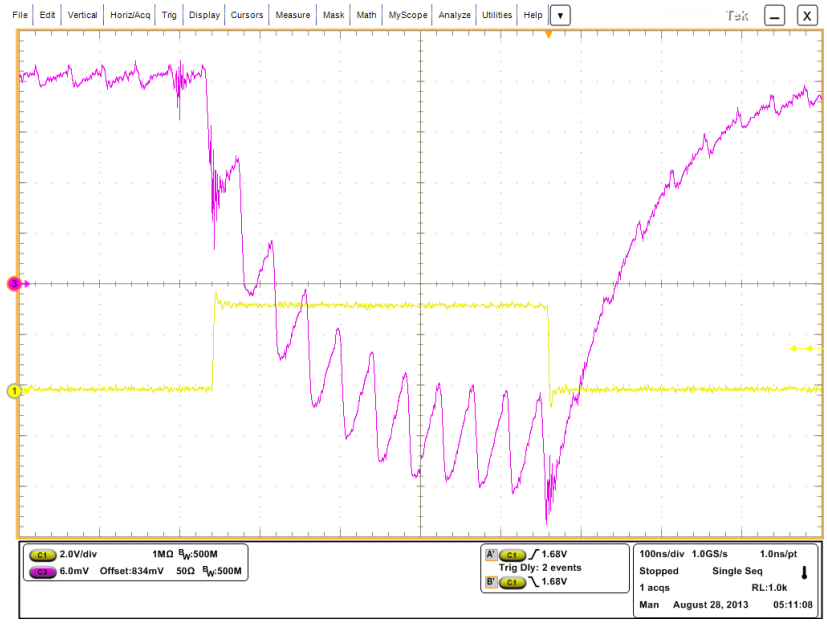

Figure 1: Raw scope capture showing power trace for a single AES operation in purple and the AES start/end trigger signal in yellow. The AES trace shows 11 sharp dips corresponding to the 11-clock edges it takes to perform an AES-128 encryption.

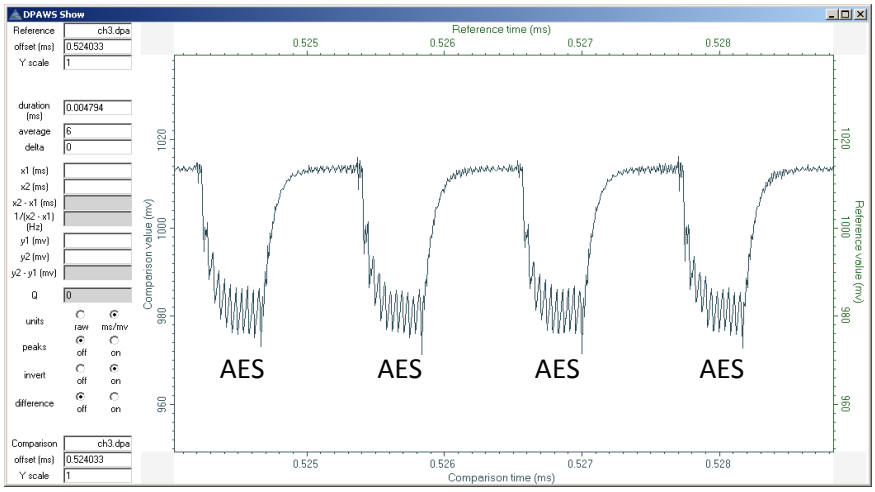

Figure 2: Section of side-channel trace captured during a bulk encryption operation using AES CBC mode. The section shows 4 consecutive AES operations that are part of the CBC-mode bulk encryption.

As noted in Section 2, when collecting individual traces for DATA-SET 1 and DATA-SET 2, the fixed encryptions specified from DATA-SET 2 need to be randomly interspersed with the sequential encryptions from DATA-SET 1. For convenience, both data sets use the same key, so if the device provides a separate interface to set the key, it can be set once for both data sets. When collecting individual AES operations, the script used to drive the data collection can easily specify the input to be

either the next in the sequence from DATA-SET 1 or DATA-SET 2, based on input from an RNG. To collect data using bulk AES in ECB mode, a similar script can provide the blocks of input plaintext that need to be encrypted, which will consist of successive blocks from DATA-SET 1 interspersed with the fixed block from DATA-SET 2. The situation is more complex when only AES-CBC mode is available for bulk encryption. In this case, the script can start with setting the IV to be 0, and then for each AES input block, selected from DATA-SET 1 or DATA-SET 2 using the same procedure as before, the plaintext block has to be calculated by computing the EXOR of the input block with the AES encryption output from the previous block.

As mentioned earlier, the non-specific, fixed-vs.-random encryption based t-test can identify leakages in AES implementations using orders of magnitude fewer traces than the specific leakage tests or key extraction attacks. Therefore for getting quick assessments, we are recommending that labs perform this test, and proceed to the other tests only if needed. For quick assessments, we can relax the tests further – while the fixed AES input test vector has been chosen to have special properties, there is nothing special about the input vectors from DATA-SET 1; any set of randomly chosen inputs would be equally good. This leads to optimizations that can greatly simplifies the task of testing AES implementations on mobile and embedded systems where lack of good triggers can make collecting AES operations individually quite time consuming and isolating side-channel signals corresponding to individual AES encryptions can be difficult due to interference from interrupts and other unrelated activities within the DUT.

In this case, we recommend running an application on the device that performs short sequence of AES bulk operations, where each bulk operation is either processing random data or fixed data. The AES bulk operations are separated with loops where no AES operations are being performed, which show up as gaps in the power trace. Figure 3 shows a side-channel trace collected using this technique. Figure 4 shows the shape of the individual AES operation within the Figure 3 trace.

Once such a trace is collected, whatever AES operations that can be extracted from the fixed or random bulk-AES sections can be used to perform the Non-specific fixed-vs.-random t-test of Section 2.1. Note that this step does not require matching each AES signal segment with its corresponding input. Therefore, sections of the trace which are corrupted by interrupts can be simply ignored. Extracting segments containing individual AES operations from the trace in Figure 3, can be done by matching with a template created using the single AES signal in Figure 4, and keeping only those segments that show a high degree of similarity to the template. Both the collection and the extraction of uncorrupted individual AES signals from the trace are fairly easy operations that can be performed by a testing lab using commonly available tools, making this test and easy and powerful way to test AES implementations on mobile and embedded systems.

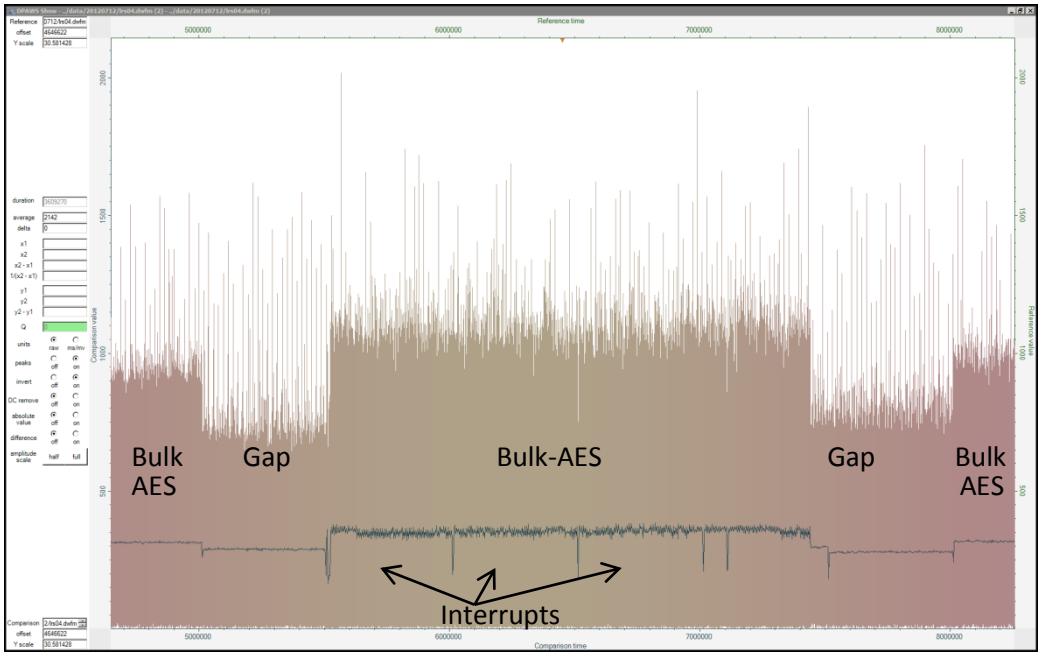

Figure 3: Side-channel trace from a mobile device application performing a sequence of short bulk AES encryptions with random or fixed data, with idle loops in between. Trace shows processor interrupts during the AES operation.

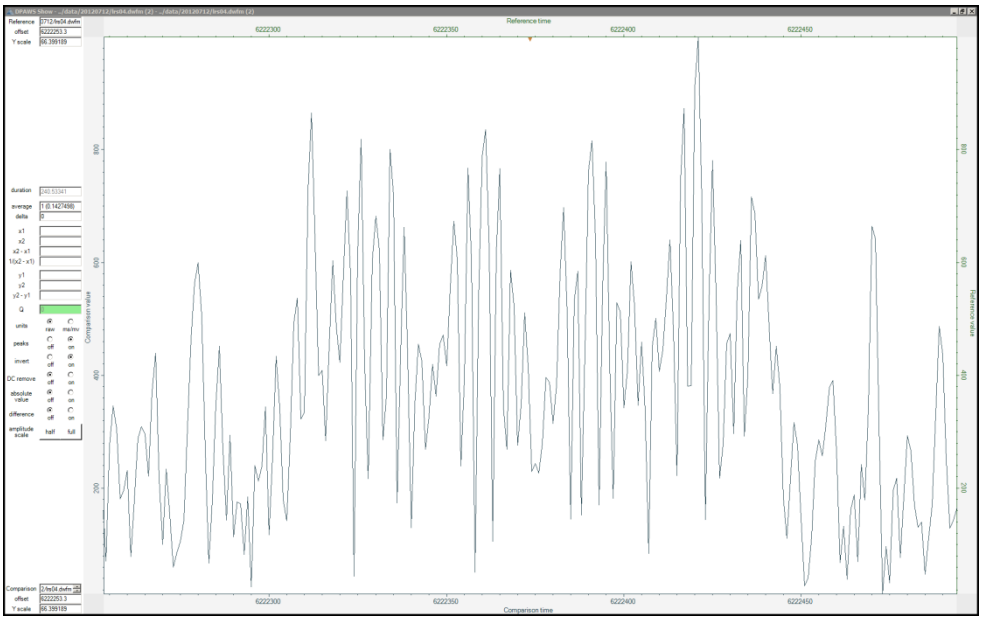

Figure 4: Section of side-channel trace from Figure 3, showing the shape of a single AES operation.

Figure 5 shows the fixed-vs.-random t-test already fails with only 100 AES operations from this device, whereas Figure 6 shows much more pronounced t-test failure when using 1000 AES operations. In both cases the trace on the top shows the average AES trace and the trace on the bottom shows the t-test trace for the fixed vs. random test. The failure lines for t-score of +4.5 and -4.5 are also plotted on the lower trace.

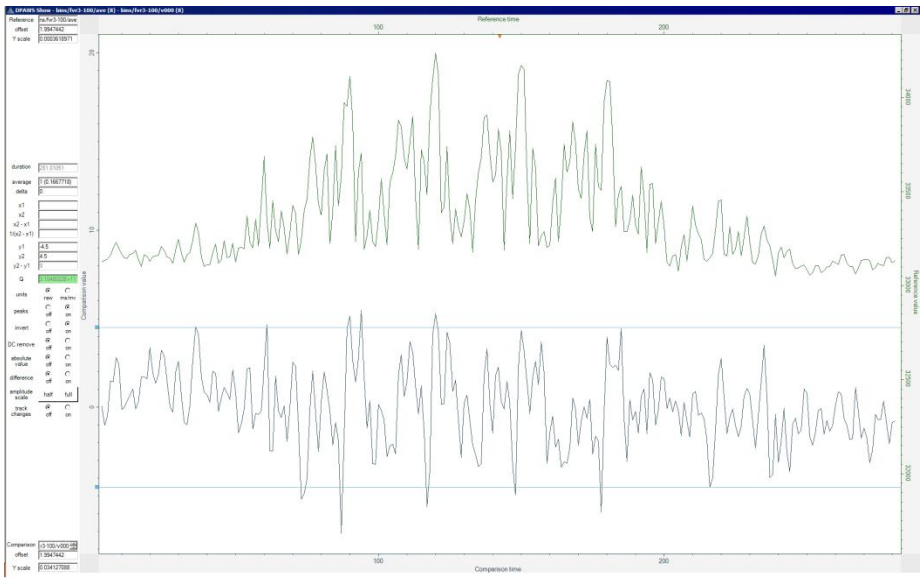

Figure 5: Average AES trace on top and t-test results on bottom together with -4.5 and +4.5 failing thresholds for the fixed-vs.-random test based on 100 AES operations. The lower t-test trace exceeds the +4.5 and -4.5 thresholds during the middle third of the AES operation and thus the device fails.

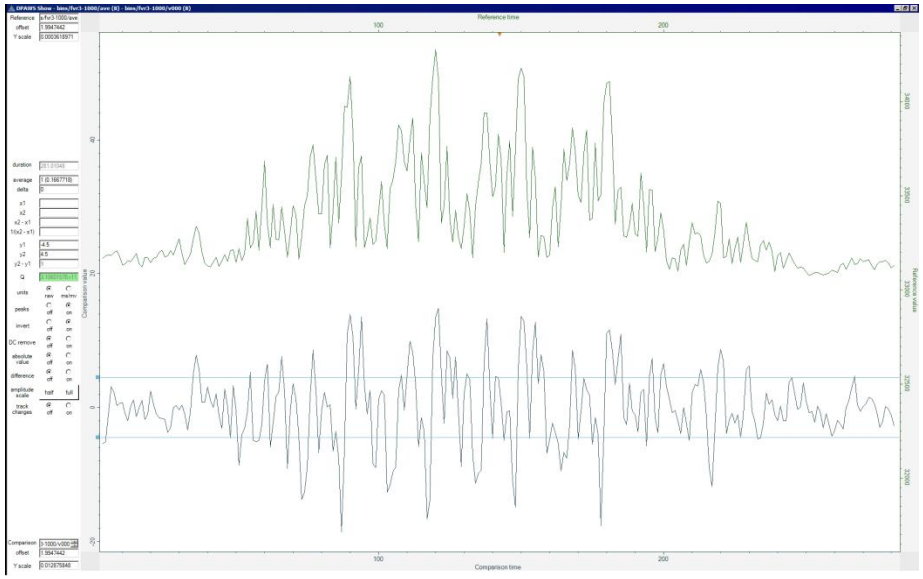

Figure 6: Average AES trace on top and t-test results on bottom together with -4.5 and +4.5 failing thresholds for the fixed vs. random test based on 1000 AES operations. The lower t-test trace significantly exceeds the +4.5 and -4.5 thresholds during the middle third of the AES operation and thus the device fails.

While the fixed vs. random tests can easily identify problems in AES implementations in embedded systems, using specific tests or performing key extraction attacks can be fairly complex and require significant more effort. To illustrate this we show how the AES implementation for the mobile device used for experiments in Figures Figure 3Figure 4Figure 5, Figure 6 could be evaluated using the specific t-tests or key extraction attacks. Both specific t-tests and key extraction attacks require matching each AES operation trace with its input. Given the problem with interrupts, the following strategy was

adopted: A special application was developed that performed a sequence of AES block encryptions according to the test-vectors specified in DATA-SET 1 in Section 2 (although following the specified sequence is not essential), but with each AES input repeated 5 times, and with an additional gap between different inputs. Figure 7 shows the section of the side-channel trace with the 5 AES operations for a particular input and the gaps before and after.

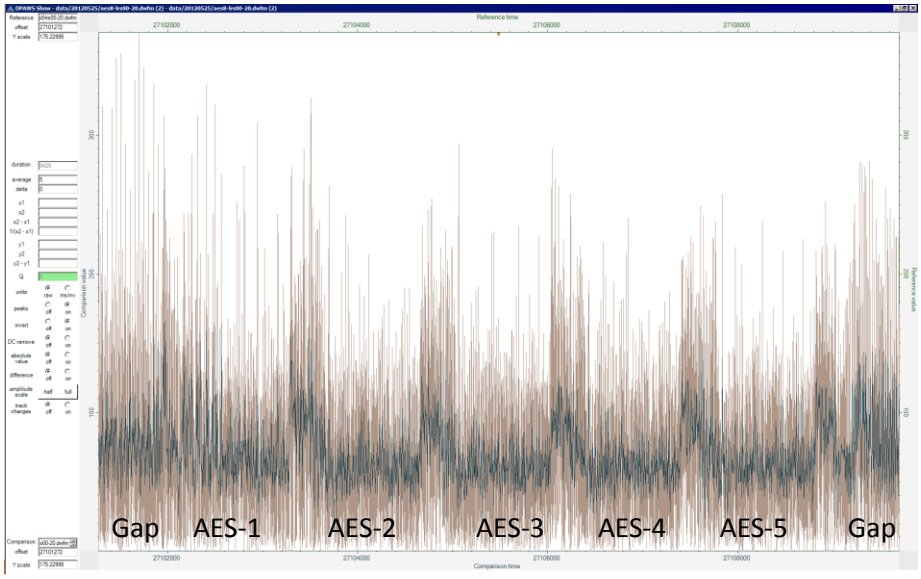

Figure 7: Part of side-channel trace showing a group of 5 AES operations with same input.

For this device, using 5 repeated AES operations with the same input and gaps between each set of 5, with significant signal processing effort it becomes possible to recover from most interrupts; at least one of the 5 AES operations would not get corrupted by the interrupt and the timing between AES operations within a group and between groups, could be used to match a recovered AES operation with its input. Using this approach, it became possible to observe leakages from the specific leakage tests as well as perform key recovery attacks using approximately 50,000 AES operations with different inputs. The process to perform key recovery attacks and specific leakage tests, from end-to-end (including effort to develop the right signal processing steps), took several weeks, whereas the end-to-end effort to set up and undertake the fixed vs. random testing was just a couple of days.

4. Side-channel testing: Comparing TVLA with Chosen Input Method

In [5], the following approach was proposed for efficient testing of AES implementations: Knowing the AES key, the tester computes special inputs to AES such that the hamming weight of the second last round state or the hamming distance between the second last round state and the last round state is maximized. Then the tester performs a correlation power analysis (CPA) attack targeting the last round of AES using the hamming weight/hamming distance model and checks whether the AES key can be extracted using a given number of traces. This technique was shown to substantially reduce the number

of traces needed to perform CPA, compared to attacks which used random inputs, at the cost of an extra (possibly offline) step of computing special inputs based on knowledge of the key. The authors of [5] demonstrated the efficacy of their technique vs. CPA using random inputs on several protected AES implementations on the on the 65-nm Crypto LSI on Sasebo-R [4]. Our testing on the same platform shows that the TVLA fixed vs. random testing is 5-30 times more efficient that even the chosen input method. The following table shows the results of our TVLA tests compared them with the published results of [5]. The second column shows the range of the number of traces needed to achieve a TVLA test failure over 5 independent TVLA tests and the third column shows the number of traces needed for by the chosen input method to fail the same implementation Since these are the same implementations on a publicly available platform, these results can be directly compared and independently verified.

Protected AES Implementation	**Range of #traces for TVLA failure over 5 independent experiments**	**#traces for Chosen input method [5]**
Masked AND (MAO)	200-1000	11,000
MDPL	200-600	6,000
Threshold	400-500	12,000
WDDL	100-400	6000
Pseudo RSL (1)	100	6000
Pseudo RSL (2)	100	3000

5. Testing AES without synchronization: updated TVLA

As illustrated, the Fixed vs. Random testing described in Section 2.1, 3 and 4 is a powerful technique for weeding out insecure AES implementations. However the technique requires the tester to identify the middle third of the AES operation, as a t-test failure during the beginning or end of AES may be from leakage of the input or output, which may not be exploitable. The test also makes an assumption that the device is not performing an unrelated operation on the input (such as moving or overwriting input data within the system) during AES processing. These assumptions may not hold for some systems. For example, in a system where there is substantial usage of timing noise and amplitude noise countermeasures, an attacker would not be able to determine the precise start and end of the AES operation and it would not be fair to provide the tester with that extra information. In such cases, it would be more reasonable to provide a trigger at the initiation of the AES command or when the timing/amplitude noise generation is started during the AES command processing, as both these events are observable by the attacker. In that case, the tester cannot determine if a failing fixed vs. random t-test is due to leakage of inputs (which may be acceptable) or leakage of internal AES state (which would be unacceptable). In devices that move AES plaintext data while performing AES, a significant t-score, even within the middle of AES could be due to side-channel differences in moving fixed vs. random data which may be an acceptable leak.

To address such problems that we have occasionally encountered during our device testing, we propose an alternative to the fixed-vs-random test that retains most of its advantages, but without these drawbacks. We refer to this alternative as the semi-fixed vs. random test. In this alternative, a

different key is used for generating DATA-SET 1 and DATA-SET 2. Also, DATA-SET 2 is no longer the encryption of a fixed input, but the encryption of a pre-specified sequence of different inputs. While the DATA-SET 2 inputs (and outputs) are statistically indistinguishable from random AES inputs (outputs), these inputs have been carefully chosen so as to cause significant parts of the internal AES state for two successive intermediate rounds to be the same for all these inputs. As an example, for AES-256, one can create a set of inputs for DATA-SET 2, such that for all these inputs, the round 7 and round 8 AES state has at 13 bytes with the same fixed value. We refer to DATA-SET 2 as the "semi-fixed" data set. The same test as prescribed in Section 2.1, can then performed using these new definitions of DATA-SET 1 and DATA-SET 2, without any limitation with respect to the start or end of AES. Any t-score peak exceeding +/- 4.5 in both tests anytime during the AES operation is grounds for failing the device, as it has identified a leakage of sensitive information from an internal state of AES.

5. 1 Creating DATA-SET 2

The following technique can be used to create DATA-SET 2: The goal is to ensure that most of the bytes of an interior round state (e.g., round 5 for AES-128, round 6 for AES-192 and round 7 for AES-192) and the subsequent round state are close to fixed. A simple way to generate such a data set is by choosing an arbitrary key and setting most bytes of the interior round state to fixed values. The input data set can then be generated by varying the few remaining bytes (e.g., 3 bytes) of the interior round state, and inverting the AES cipher to compute the corresponding input that results in that interior round state. By clever choice of which bytes of the interior round state to vary, the impact of the varying bytes on the next round state can also be limited. For example, by choosing a subset of the bytes 0, 5, 10, 15 in the AES specification (the bytes in the diagonal) to vary, only the bytes 0, 1, 2, 3 in the next round state will vary and all the remaining 12 bytes of the state will remain fixed. In general, varying a byte in a round state affects all the 4 bytes in the AES column to which that byte get mapped to after the shift-rows operation.

For our tests, we use a slightly different semi-fixed data set construction technique to draw out even more leakage. It is our experience that a 0 byte in an AES round state (that becomes a 0 input to an S-box operation), or a low hamming distance between successive round states draw out more leakages. By incorporating these additional design criteria, we created a semi-fixed data set and key, where an internal round state has several 0 bytes, and the hamming distance between that round state and the next also have several 0 bytes. Upon request, we can make that data set and key available.

6. References

1. FIPS 140-3 PUB Development, http://csrc.nist.gov/groups/ST/FIPS140_3/
2. Goodwill G., Jun B., Jaffe J. and Rohatgi P: "A testing methodology for side-channel resistance validation", NIST Non-Invasive Attack Testing Workshop, September 2011. Available at http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf
3. Jaffe J., Rohatgi P. and Witteman M: "Efficient side-channel testing for public key algorithms: RSA case study" NIST Non-Invasive Attack Testing Workshop, September 2011. Available at http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/09_Jaffe.pdf

4. Standard Cryptographic LSI Specification - Countermeasures against Side Channel Attacks (65nm) , Version 0.9, http://www.risec.aist.go.jp/project/sasebo/download/CryptoLSI3_Spec_Ver0.9_English.pdf
5. T. Kishikawa, S. Saito, Y. Tsuchiya, T. Toyama, T. Matsumoto (Yokohama Nat'l Univ.), “An Efficient Method of Strictly Evaluating Side-Channel Security“, IEICE Tech. Rep., vol. 112, no. 211, ISEC2012-56, pp. 67-74, Sept. 2012,