Applying TVLA to Public Key Cryptographic Algorithms

Michael Tunstall
Gilbert Goodwill

Rambus

Introduction

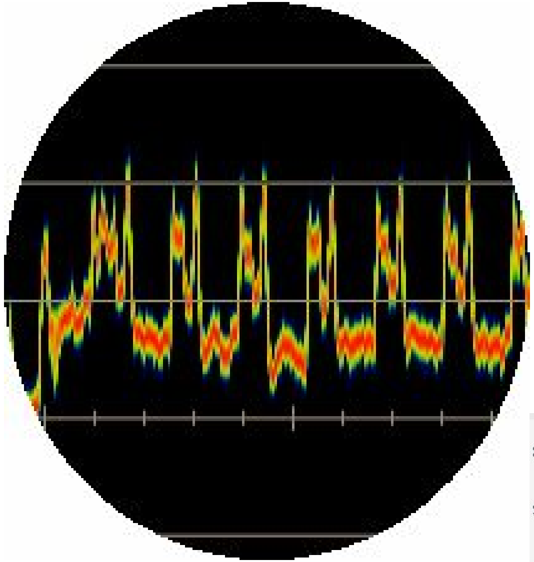
- Test Vector Leakage Assessment (TVLA) was proposed in 2012
- Efficient in evaluating the presence of leakage in block ciphers
- The choice of implementation details make applying the same strategy to Public-key cryptographic algorithms problematic

Statistical leaks: Data dependence in AES

- Using a scope's "infinite persistence" mode to overlay the different traces.

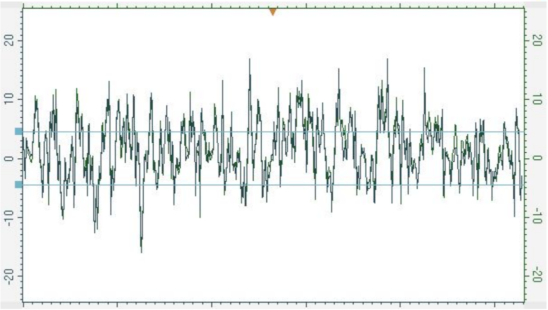
Statistical leaks: Data dependence in AES

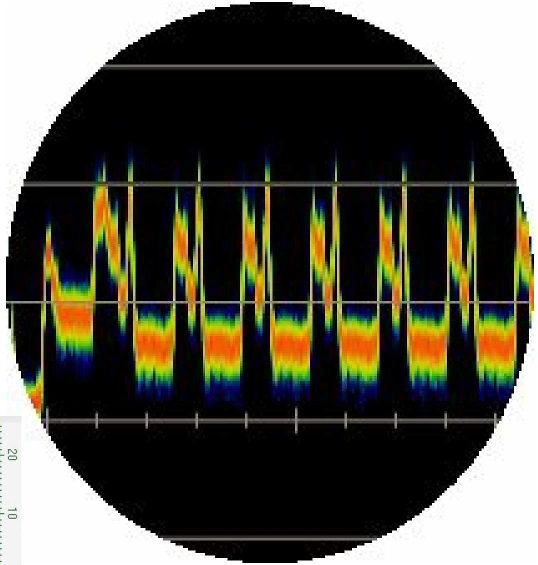
- Using a scope's "infinite persistence" mode to overlay the different traces



- 1000 AES decryptions with constant key and ciphertext

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

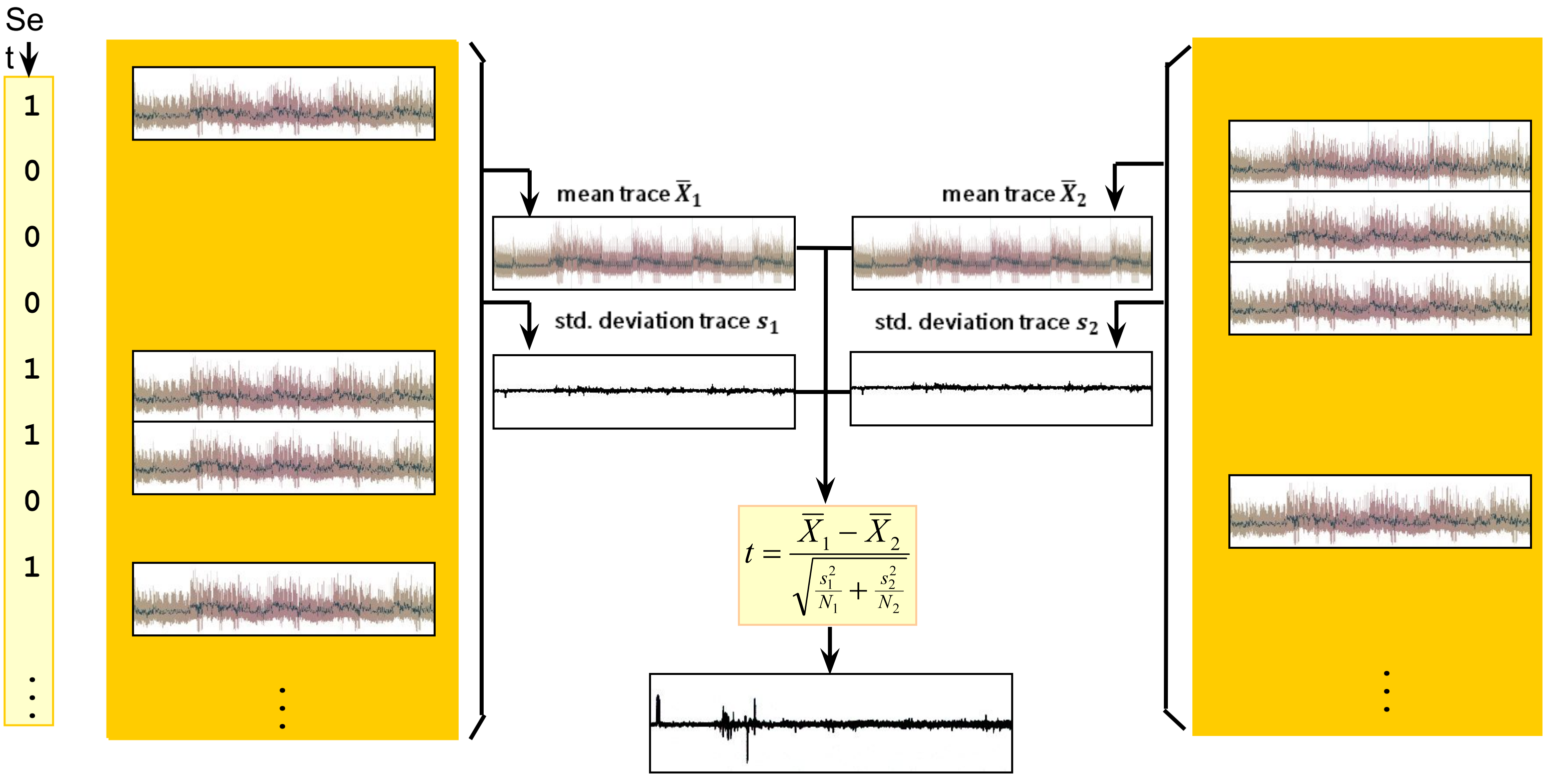




1000 AES decryptions with varying key and ciphertext

Leakage Detection: Test Summary

- Comparing traces from two sets:



TVLA on Public-Key Algorithms

- TVLA typically applied to block ciphers, something like:
  - Fixed secret key vs. random key
  - Fixed message vs. random message
- For public-key algorithms it is not so straightforward
- We propose a process, as follows:
  1. Theoretical Analysis
  2. Timing Analysis
  3. Simple Power Analysis
  4. Leakage Detection
  5. Collision Attacks

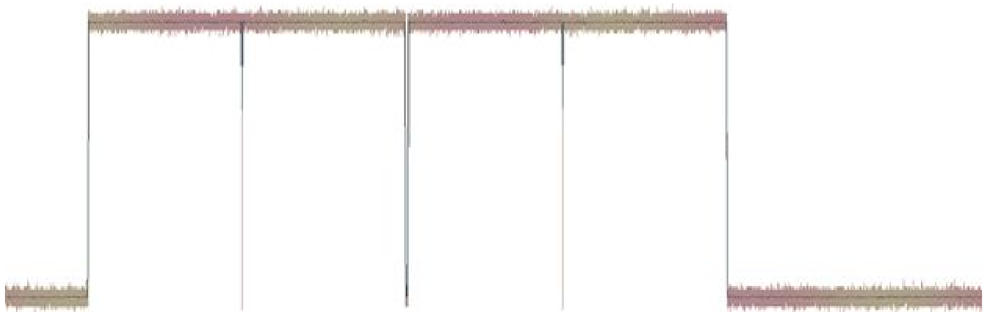
TVLA: Theoretical Analysis

- Information gathering
  - What group exponentiation algorithm is being used?
  - Other potential vulnerabilities?

- Ideally, one would have full implementation details
- If the implementation is not known, some information can still be determined
  - How many bits does the group exponentiation take in one loop?
  - Are there any operations that execute in a variable amount of time?

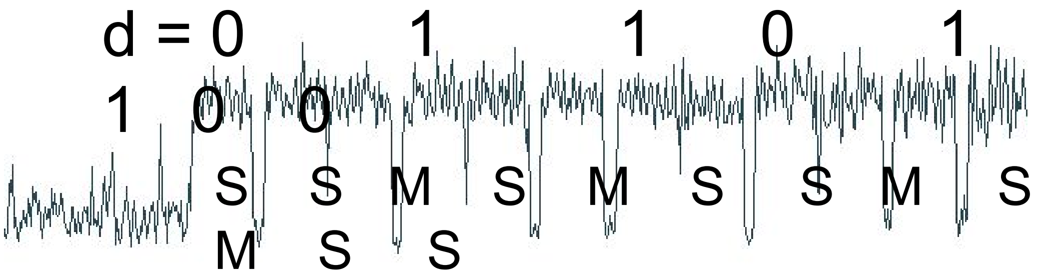
TVLA: Timing Analysis

- Do any operations execute in a variable amount of time?
  - Montgomery multiplication
  - Extended-GCD
- We test to determine if the time taken would indicate leakage for a fixed input compared to a random input.



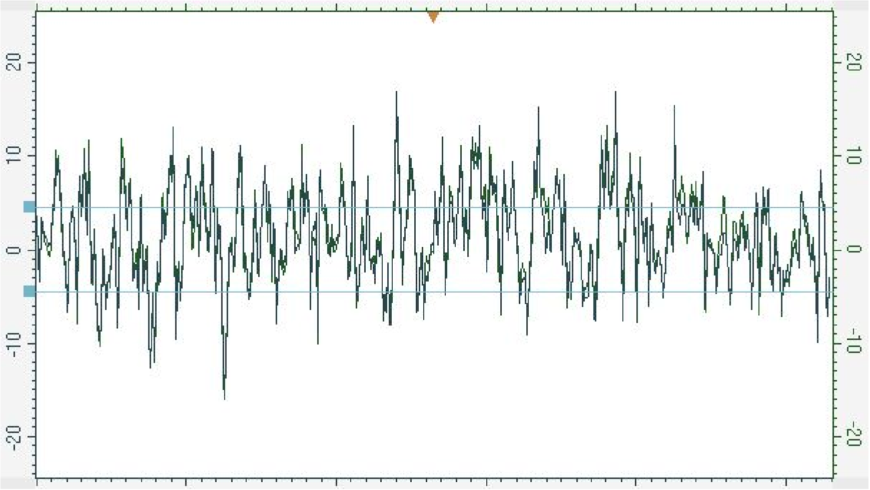
TVLA: Simple Power Analysis

- Can patterns be spotted in some operations, Typically targeting:
  - The group exponentiation algorithm
  - Final subtraction on Montgomery multiplication
- Optional, since leakage detection will reveal this quickly
  - But may save time



TVLA: Leakage Detection

- As with block ciphers we have, something like:
  - Fixed secret key vs. random key
  - Fixed message vs. random message
- Variation depending on the algorithm
- No statistic greater that 4.5
  - $P(\text{false positive}) = 1 \times 10^{-5}$
  - $P(\text{false negative})$ is undefined



TVLA: Collision Analysis

- Also known as:
  - The BigMac Attack
  - Horizontal Side-Channel Analysis
  - Collision-Correlation Analysis
  - Correlation-Collision Analysis
  - The Riscure Attack
- Class of attacks looking for intermediate values that are the same at two points in an algorithm
  - Identical operand(s) for operations
- Only concerned with attacks applied to one trace

Algorithm 1: Joye's Add-Only Scalar Multiplication

Input: $\boldsymbol{P}$ a point on elliptic curve $\mathcal{E}$, an n-bit scalar $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$

Output: $\boldsymbol{Q} = k\,\boldsymbol{P}$

$\boldsymbol{R}_0 \leftarrow \boldsymbol{\mathcal{O}}$; $\boldsymbol{R}_1 \leftarrow \boldsymbol{P}$; $\boldsymbol{R}_2 \leftarrow \boldsymbol{P}$;

for $i \leftarrow 0$ **to** $n-1$ **do**

  $\boldsymbol{R}_{1-k_i} \leftarrow \boldsymbol{R}_{1-k_i} + \boldsymbol{R}_2$;

  $\boldsymbol{R}_2 \leftarrow \boldsymbol{R}_0 + \boldsymbol{R}_1$;

end

return $\boldsymbol{R}_0$

- For example, we note that R_0 in round i …

Algorithm 1: Joye's Add-Only Scalar Multiplication

Input: P a point on elliptic curve $\mathcal{E}$, an n-bit scalar $k = (k_{n-1}, k_{n-2}, \ldots, k_0)_2$

Output: $Q = kP$

$R_0 \leftarrow \mathcal{O}$; $R_1 \leftarrow P$; $R_2 \leftarrow P$;

for $i \leftarrow 0$ **to** $n-1$ **do**

  $R_{1-k_i} \leftarrow R_{1-k_i} + R_2$;

  $R_2 \leftarrow R_0 + R_1$;

end

return R_0

- We note that R_0 in round i, will be the same as the first operand of the first operation in round $i+1$.

TVLA: Collision Analysis

- Conducting an attack on a, potentially unknown, implementation will be overly complicated

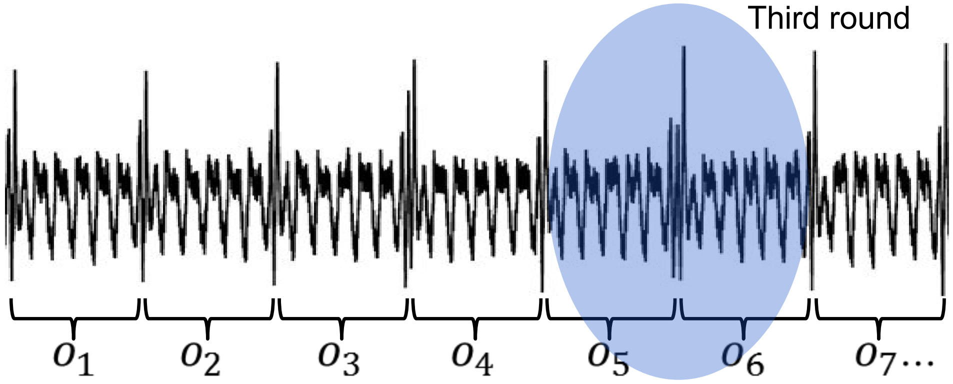
- We can adapt leakage detection to compare every operation in one round with every operation in the following round

- We take a set of traces and extract 1×10^3 traces where two consecutive bits are 00
  - Arbitrarily, we shall consider the third and fourth round
  - Assume two operations per round

TVLA: Collision Analysis

- Break each trace into subtraces corresponding to individual operations

$$O = \{o_1, o_2, o_3, \dots, o_n\}$$



- Generate a mean subtrace $\bar{o}$
- Subtract pointwise from each subtrace

$$\hat{O} = \{o_1 - \bar{o}, o_2 - \bar{o}, o_3 - \bar{o}, \dots, o_n - \bar{o}\} = \{\hat{o}_1, \hat{o}_2, \hat{o}_3, \dots, \hat{o}_n\}$$

TVLA: Collision Analysis

- We compute difference trace

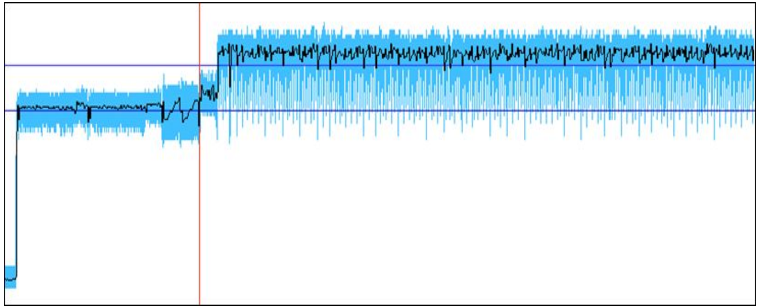
$$\Delta = \{\hat{o}_5 - \hat{o}_7, \hat{o}_5 - \hat{o}_8, \hat{o}_6 - \hat{o}_7, \hat{o}_6 - \hat{o}_8\}$$

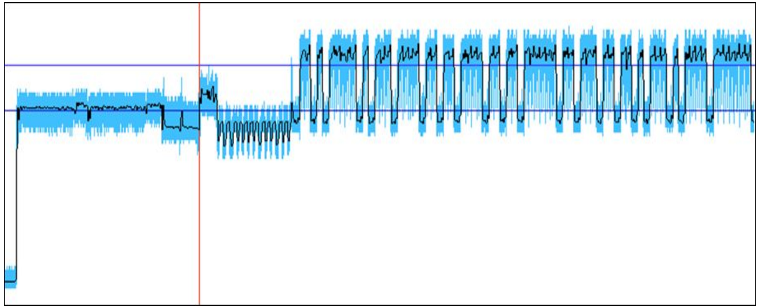
- That is, all the possible combinations of operations when comparing the third and fourth rounds
- Gives a set of 1×10^3 difference traces for 00 case

- Repeat with randomly selected traces to produce 1×10^3 difference traces
- Gives a fixed and a random case for leakage detection
- Conduct attacks for bits 3 and 4 set to $\{00,01,10,11\}$

Applying TVLA to RSA

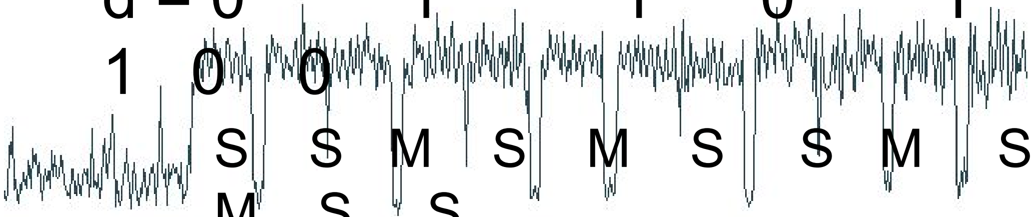
- Theoretical analysis
  - Are there any operations that take a variable amount of time?
    - E.g. modular inversion
  - What information do we have on the exponentiation algorithm used?
  - Are there any special values that cause leakage?
    - E.g. 1, 2,n-1 etc.





Applying TVLA to RSA

- Timing Analysis
  - As described previously on any identified operations
- Simple Power Analysis
  - Can any information be derived from inspecting traces
  - Optional, but potentially saves evaluation time

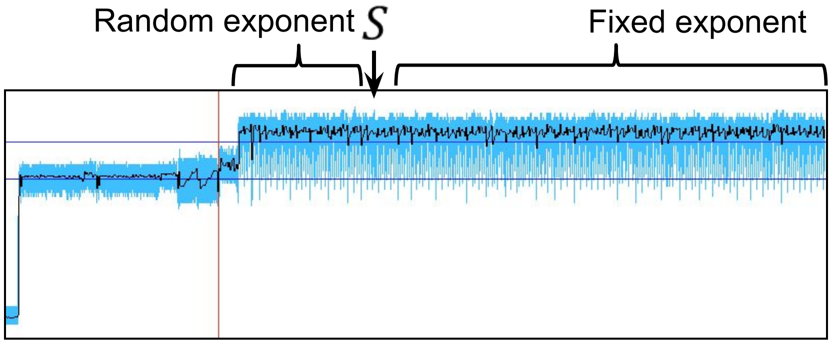


Applying TVLA to RSA

- Leakage Detection
  - Straightforward approach for testing private key, i.e. compare effect of a fixed private key with a random private key
    - Assumes that the keys are blinded when they are loaded
  - Input message not so straightforward,
    - Pick a point in an exponentiation and choose an arbitrary fixed state S
    - Generate random exponent bits and compute the input that would lead to S, with the rest of the exponent fixed
    - Compare to a random input, gives a leakage detection test from S onwards
    - Equivalent to semi-fixed vs. random strategy used for block ciphers

Applying TVLA to RSA

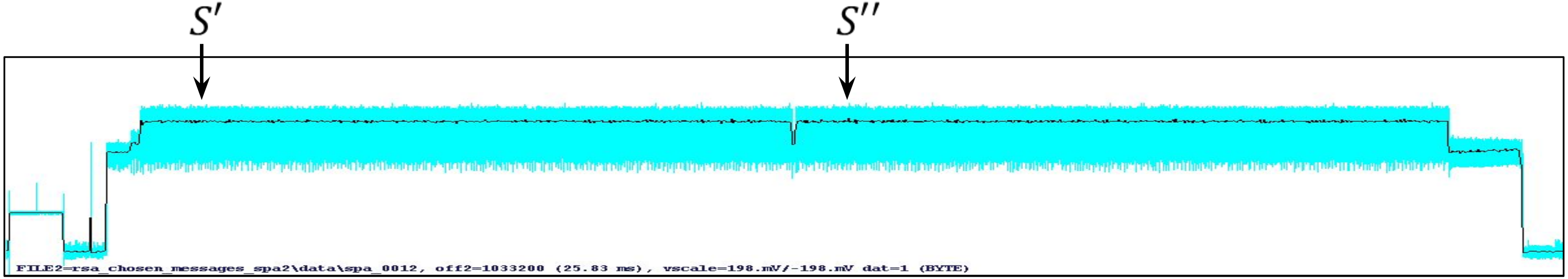
- Leakage Detection



- Blinding should mean S is not visible
- Requires the private key to be changed
- May cause problems with countermeasures to fault attacks

Applying TVLA to RSA

- Leakage Detection



- The same process can be applied to RSA computed using the CRT
- Choose S' and S'' and use the CRT to derive the private key and input
- Requires the private key to be changed
- May cause problems with countermeasures to fault attacks

Applying TVLA to Elliptic Curve-based Algorithms

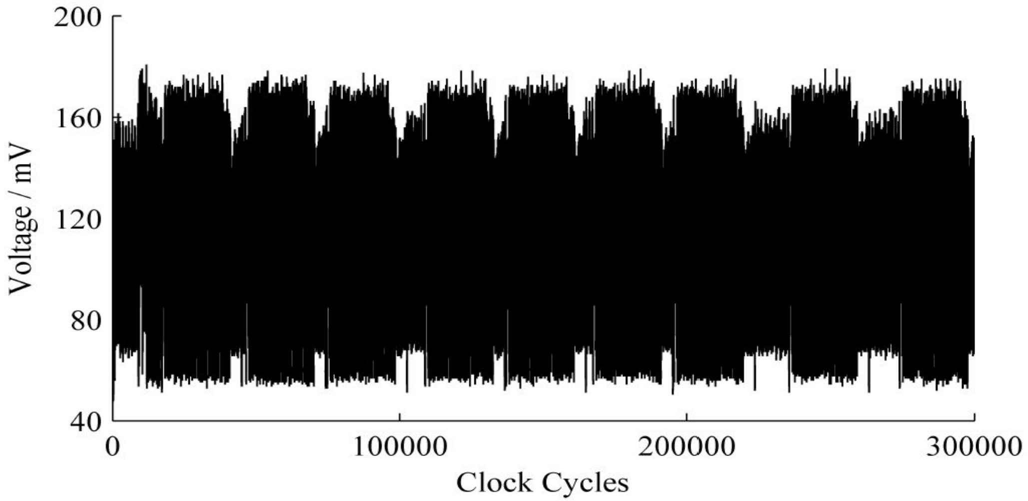
- Theoretical analysis
  - Are there any operations that take a variable amount of time?
    - E.g. modular inversion
  - What information do we have on the group exponentiation algorithm used?
- Timing Analysis
  - As described previously on any identified operations
- Simple Power Analysis
  - Can any information be derived from inspecting traces
  - Optional, but potentially saves time

Applying TVLA to Elliptic Curve-based Algorithms

- Leakage Detection
  - As for RSA, private keys can be tested by comparing the side channel during the treatment of a fixed key compared to a random key.
    - Targeted operations different for ECDH and ECDSA
  - Inputs to compare are algorithm dependent
    - ECDH we choose a point and use it in the same way that we use a chosen state for RSA to generate a public key
    - ECDSA we choose a state for the combination of the x-coordinate output in the signature with the hashed message to be signed

Applying TVLA to Elliptic Curve-based Algorithms

- Collision analysis
  - Operations are not all the same, i.e., additions and doubling operations
  - Compress subtraces by extracting field multiplications from subtraces



Applying TVLA to Elliptic Curve-based Algorithms

- Collision analysis
  - Comparing doubling operations or additions in consecutive rounds is straightforward
  - Comparing a doubling operation with an addition we need to compare each field multiplication in one operation with each field multiplication in the other

  - A matrix of small difference traces to generate difference traces for testing
  - Otherwise, the procedure is the same as the general case

Conclusion

- TVLA can be applied to public-key cryptographic algorithms
- More complex because of the number of implementation choices
- Theoretical analysis can have a large effect on subsequent tests
  - Difficult to define a standard battery of tests that will account for all cases