An Implementation Vulnerability Analysis - A case study on ChaCha of SPHINCS

Varun Satheesh, Dillibabu Shanmugam

SETS, Chennai

MOTIVATION and CONTRIBUTION

- Working on Post-Quantum Cryptography- particularly on deploying pq algorithms on hardware(FPGA) and analysing Side Channel behavior
- Secure hardware implementation of ChaCha stream cipher on FPGA hardware against Differential Power Analysis. Threshold Implementation as our countermeasure against Side Channel Attacks.

What is SPHINCS?

Post Quantum Digital Signature

SPHINCS(Stateless practical hash-based incredibly nice cryptographic signatures)

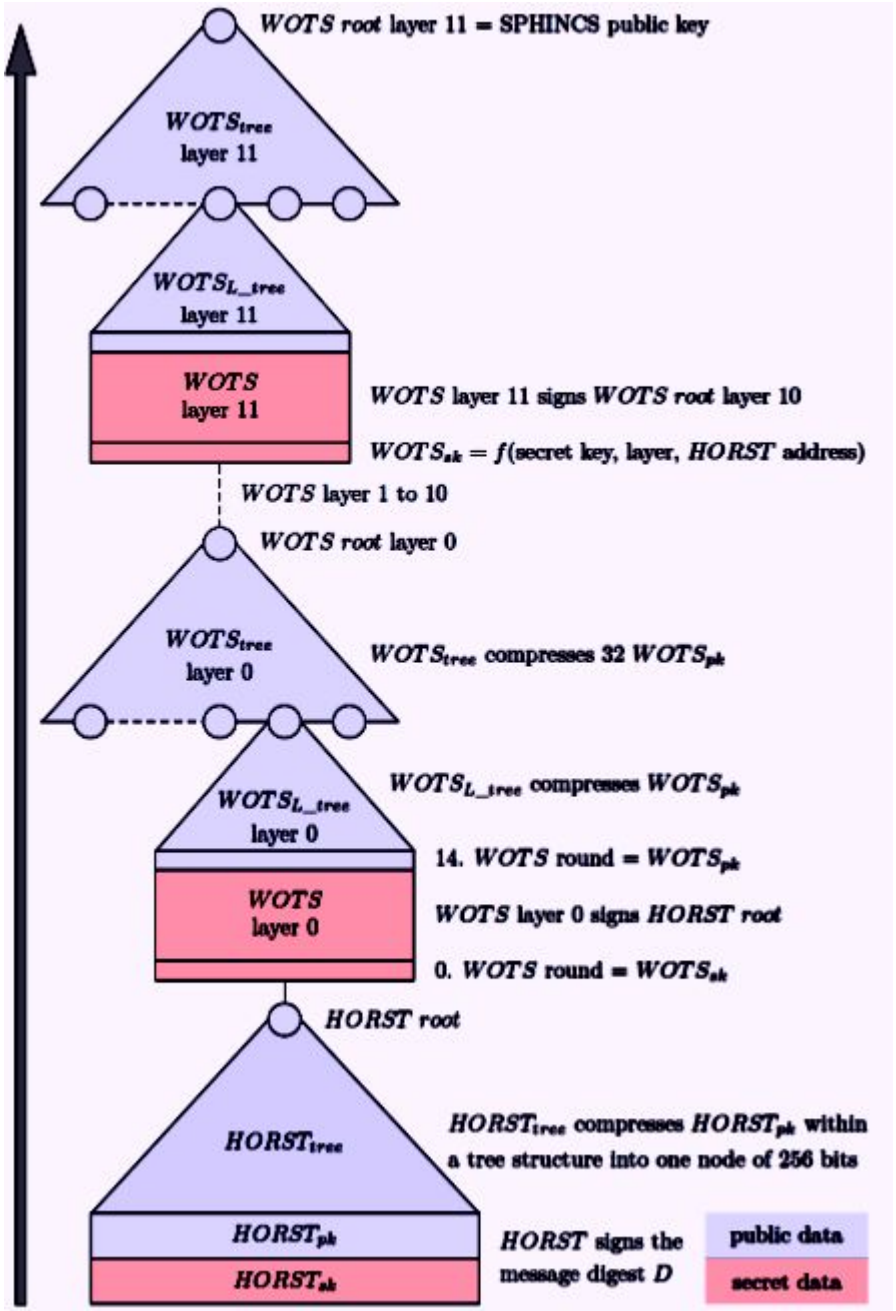
- is a quantum attack resistant digital signature scheme
- NIST PQC Round 3 alternate candidate(SPHINCS+)
- developed by Daniel J Bernstein and team in the year 2015
- hash based scheme that is simple and provably secure

SPHINCS

Designed to be collision-resilient
Designed for 128(95) bits of post-quantum security.
Trees: MSS trees
OTS: WOTS+
FTS: HORST
12 trees of height 5 each

n = 256 bit hashes in WOTS and HORST
m = 512 bit message hash (PRF: BLAKE-512)
PRG: ChaCha12

Signature: 41000 Bytes
Public key: 1056 Bytes
Secret Key: 1088 Bytes



ARX ciphers

ChaCha is also an ARX cipher, namely composed of Modular **A**ddition, **R**otation and **X**OR operations

Popular ARX based ciphers and functions are:

Block ciphers: Speck, SPARX, FEAL, Threefish
Stream ciphers: Salsa20, ChaCha(R)
Hash functions: BLAKE

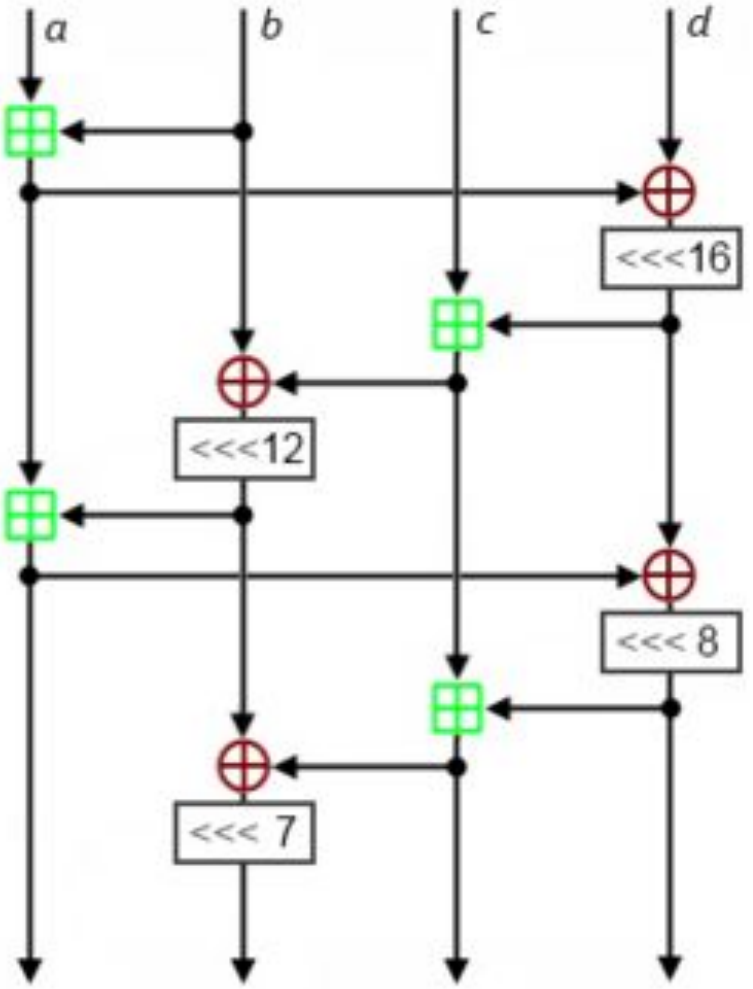
Where is ChaCha used?

ChaCha is a stream cipher developed by Daniel Bernstein.
Pseudo-random number generator in SPHINCS.
Operations take place over 4 elements at a time, hence a ***quarter round***-

constant	constant	constant	constant
key_0	key_1	key_2	key_3
key_4	key_5	key_6	key_7
counter	counter	nonce	nonce

Initial round of ChaCha matrix

ChaCha QR



\+ Modular addition

^ Bitwise XOR

<< Bit shift

What is the fault with ChaCha?

As we saw, ChaCha operates on 4 elements at a time and performs ARX operations on its elements.

Modular addition is a non-linear operation that shows vulnerability to Side Channel Analysis.

Power consumption can be tapped from operations as traces and secret information can be retrieved with statistical analysis.

Existing attack literature on ARX ciphers

- Attack on primitive's structure
  - "DPA on XMSS"
  - "Bricklayer Attack" on ChaCha20 Attack on building blocks
- Attack on cipher building blocks
  - "DPA on Speck, SPARX"

Crypto Modules Evaluation process

"Secure algorithm ≠ secure implementation"

Implement

Efficient realization of crypto primitives on various platforms

Do functional testing

Leakage Assessment

TVLA

Explore Vulnerability

Five steps of CPA

Countermeasure

Provable countermeasures (masking and hiding)

- Threshold Implementation
- Algorithmic countermeasures

Evaluation

Common criteria: use all the strategy

NIST FIPS-140-2/3 CMVP: Test Vector Leakage Assessment

State-of-the-art approach (Deep Learning)

Standards : FIPS140-2/3 and Common Criteria

- Effectiveness of test: Reproducible & reasonable indicators of resistance
- Ease & cost-effective: validation should not require excess amount, time and skill.

FIPS140-2/3

- Conformance-style testing: Detects the presence of any leakage, independent of attack methodologies and leakage models
- Test Vector Leakage Assessment(TVLA)

Common Criteria

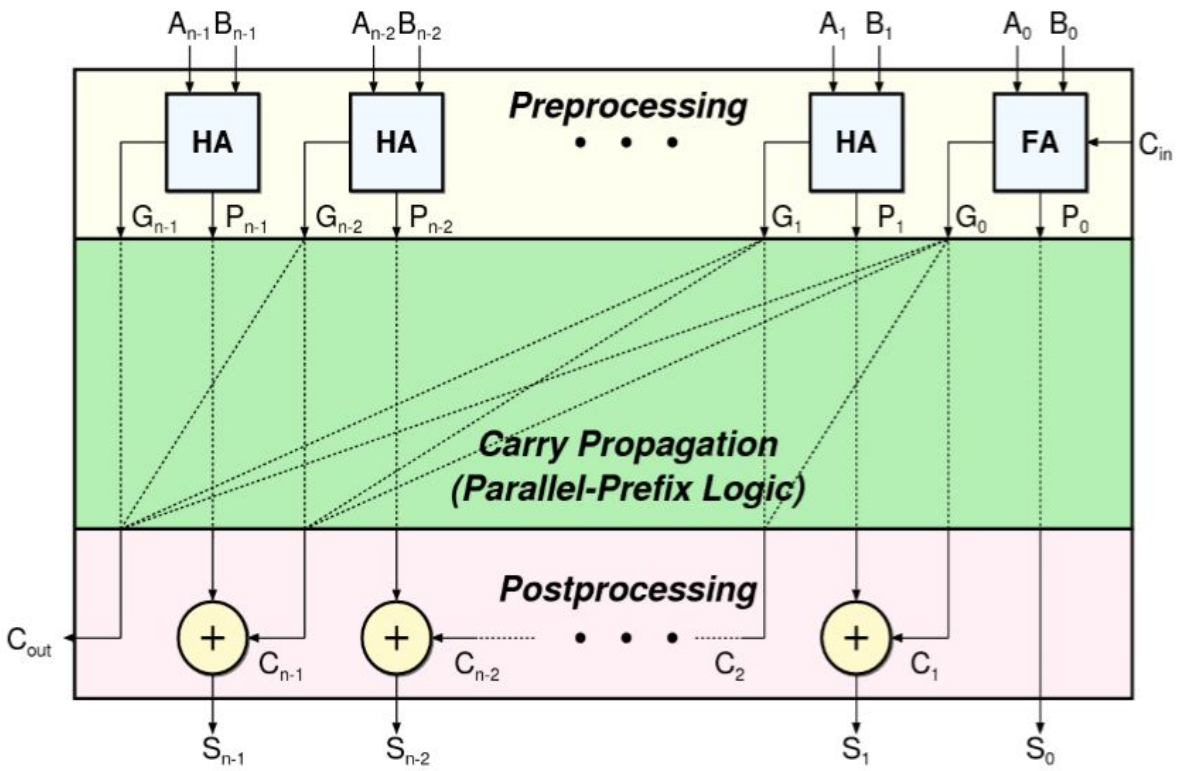
- Evaluation-style testing: Evaluating the system against all attack strategies.
- Testing methodology tedious, costly and limited by the testing expertise available at the hand.

JCMVP (Japan Cryptographic Module Validation Program): for testing and validation of commercial cryptosystems

Fast Adders

- Parallel Prefix Adders pre-computes the carry propagation with the parallel prefix technique.
- Performance depends on the design of the carry chain.
- We adopt PPA as a function in our implementation that carries out binary addition.
- Various adders, e.g. Brent-Kung, Kogge-Stone, Han-Carlson

Parallel Prefix Adders- Fast Adders



1. Pre-Processing stage: generate and propagate signals are calculated.

2. Prefix stage: Group level generate and propagate signals are computed using different prefix trees.

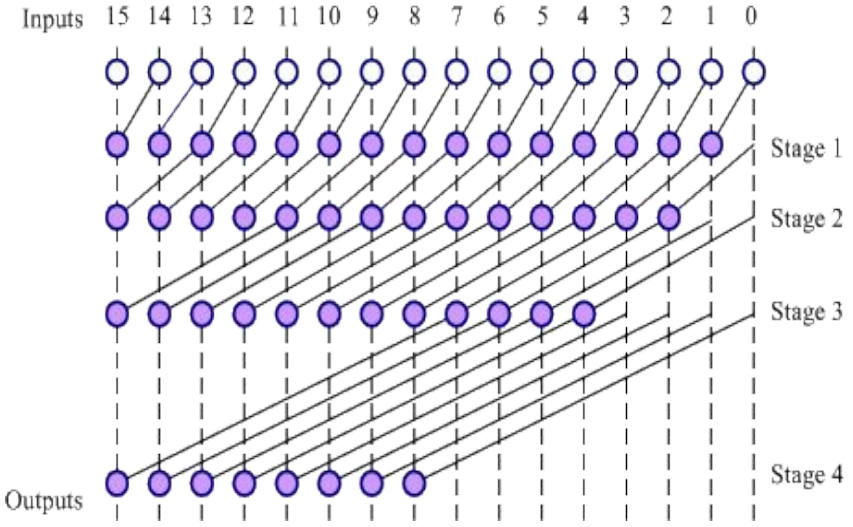
3. Post-processing stage: Sum bits are calculated using simple XOR or conditional sum adders.

Fast Adders

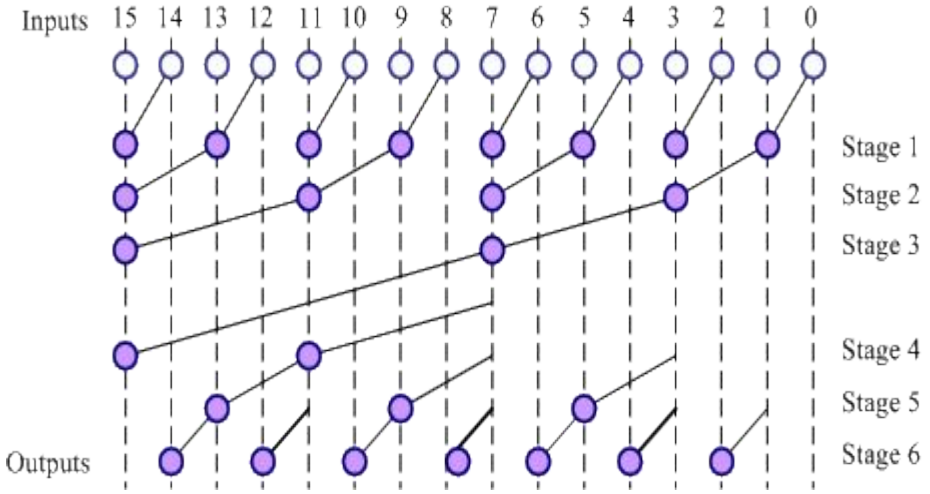
Parallel Prefix Adders : Brent-Kung, Kogge Stone, and Han-Carlson

Adders	Area	Delay	Fan-out	Wiring Complexity	Application Devices	Implementation style	Profiles FPGA utilization
BKA	Less	High	log n	Less	Constrained	Serial	P1(748GE)
KSA	High(	Medium	2	High	Conventional	Round	P2(1106GE)
HCA	Medium	Less	2	Medium	Accelerator	Unrolled	P3(11423GE)

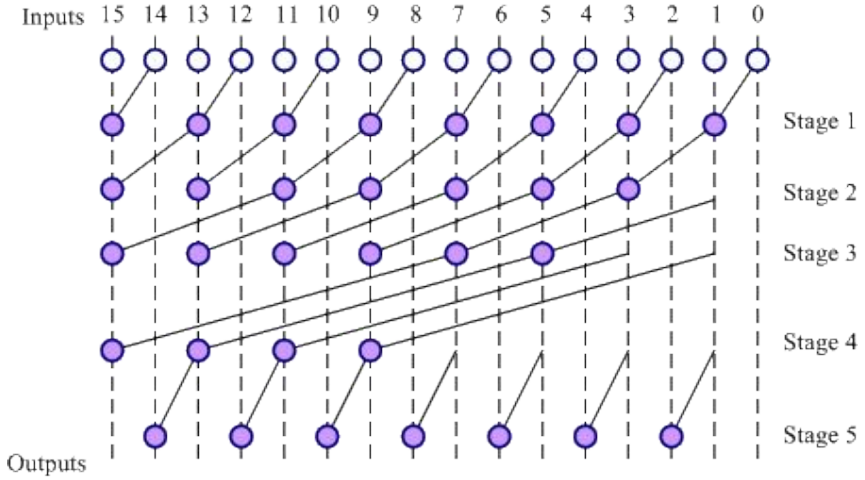
Various PPAs



Kogge-Stone Adder



Brent-Kung Adder



Han-Carlson Adder

What have we done?

- Protection at Three levels:
  - At building blocks: Identify efficient components for hardware realization
  - At countermeasure level: Identify secure and efficient countermeasure like , TI
  - At Algorithm structural level: Different implementation styles(serial, round partially unrolled)

- Created Implementation Styles
- Used fast adders in place of third party adder functionality(Xilinx Vivado EDA tool)
- Adopted Threshold Implementation on top of the fast adders
- Provably secure implementation against Side Channel Attacks

Implementation Styles

- Profile 1 : ChaCha using BKA with serial implementation style
- Profile 2: ChaCha using KSA with round based implementation style
- Profile 3: ChaCha using HCA with partially unrolled implementation style

Threshold Implementation

- Provably secure countermeasure technique
- Based on the secret sharing principal, even (n-1) shares knowledge will preserve the secrecy of the key.
- Two share TI is adopted. Sensitive intermediate functions are converted into shared functions by applying additive boolean masking.

Threshold Implementation - Properties

- Correctness - sum of shared input and output value must be the same as unshared input and output value
- Non-completeness - each shared function must be independent of one input value
- Uniformity - probability of shared output occurrence must be uniform and equally distributed

2 share TI on Adder

Z = a . b + c

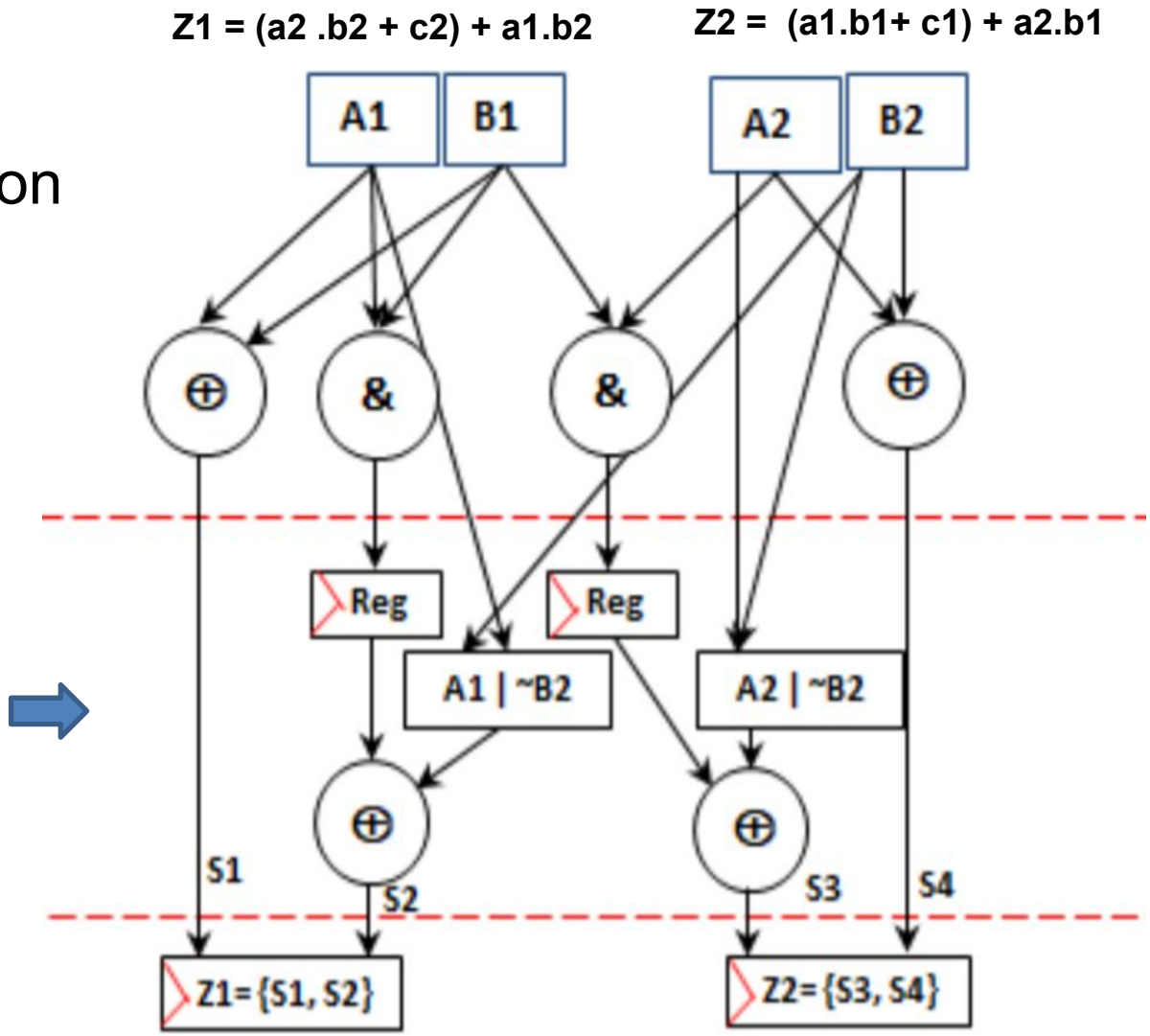
Threshold Implementation
3 share to optimized 2 share
TI properties are satisfied

3 Share implementation

$Z_1 = a_2b_2 + a_2b_3 + a_3b_2 + c_2$

$Z_2 = a_3b_3 + a_3b_1 + a_1b_3 + c_3$

$Z_3 = a_1b_1 + a_1b_2 + a_2b_1 + c_1$

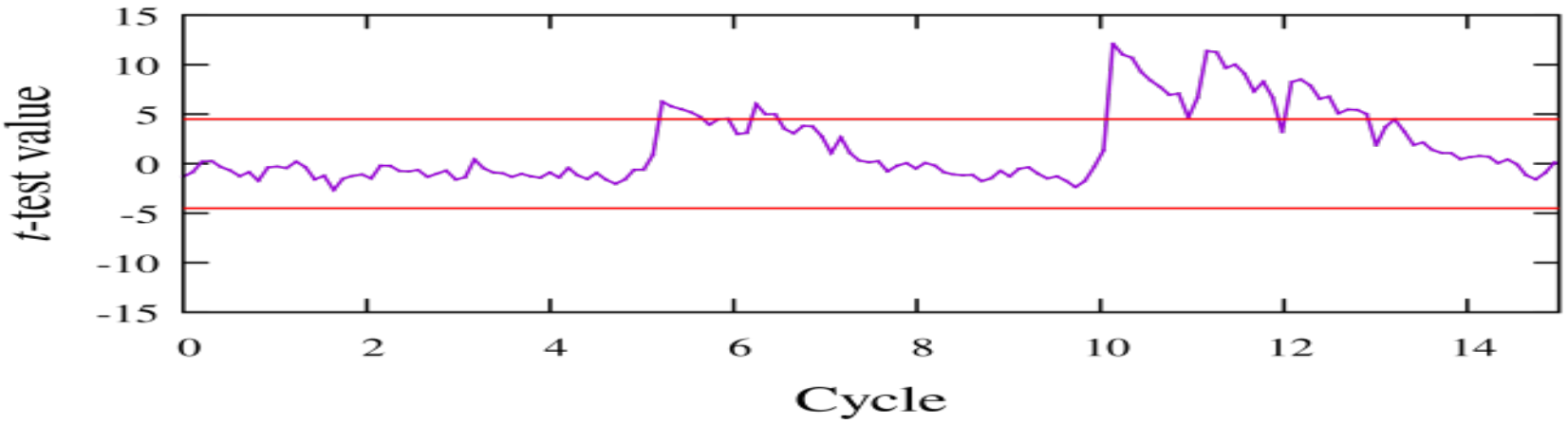


Test Vector Leakage Assessment(TVLA)

TVLA uses well known Welch's t-test. It was proposed as a PASS/FAIL test. If t-value crosses the predefined threshold (proposed as +4.5 to -4.5), then it will leak potential information; otherwise not.

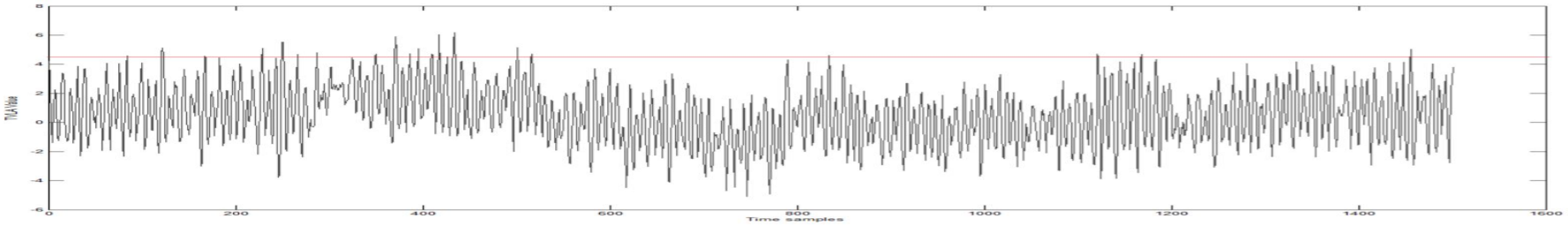
	Data Set 1(Fixed): 5000 Traces	Data Set 2(Random): 5000 traces
Plaintext	Fixed	Random
Key	Fixed	Fixed

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

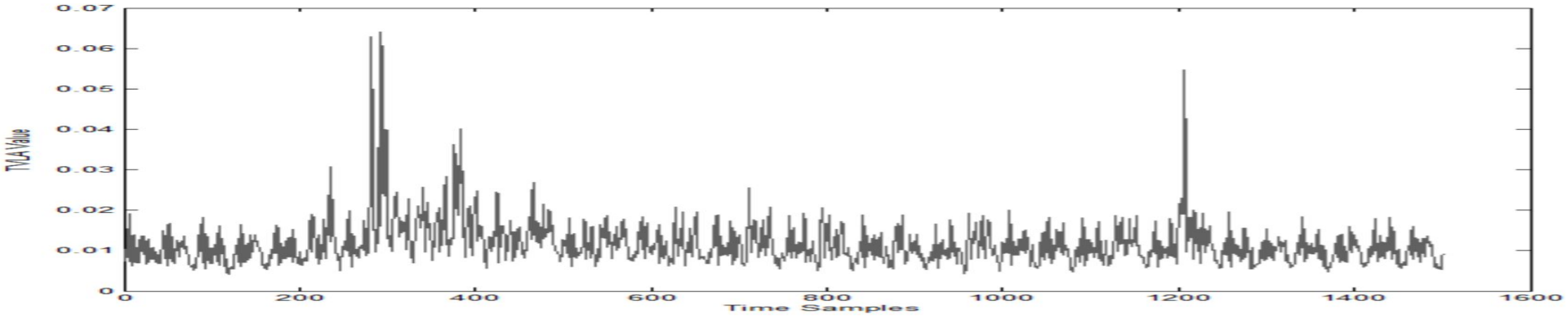


- 1 million power traces are captured for each of the profiles.
- Proven by TVLA testing that there is no information leakage through power.

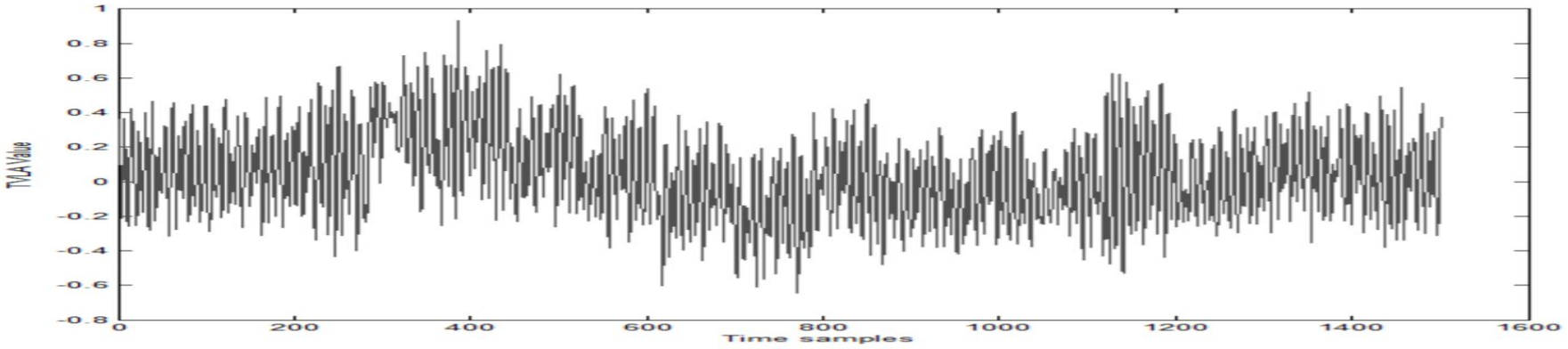
TVLA



TVLA ChaCha BKA - Naive

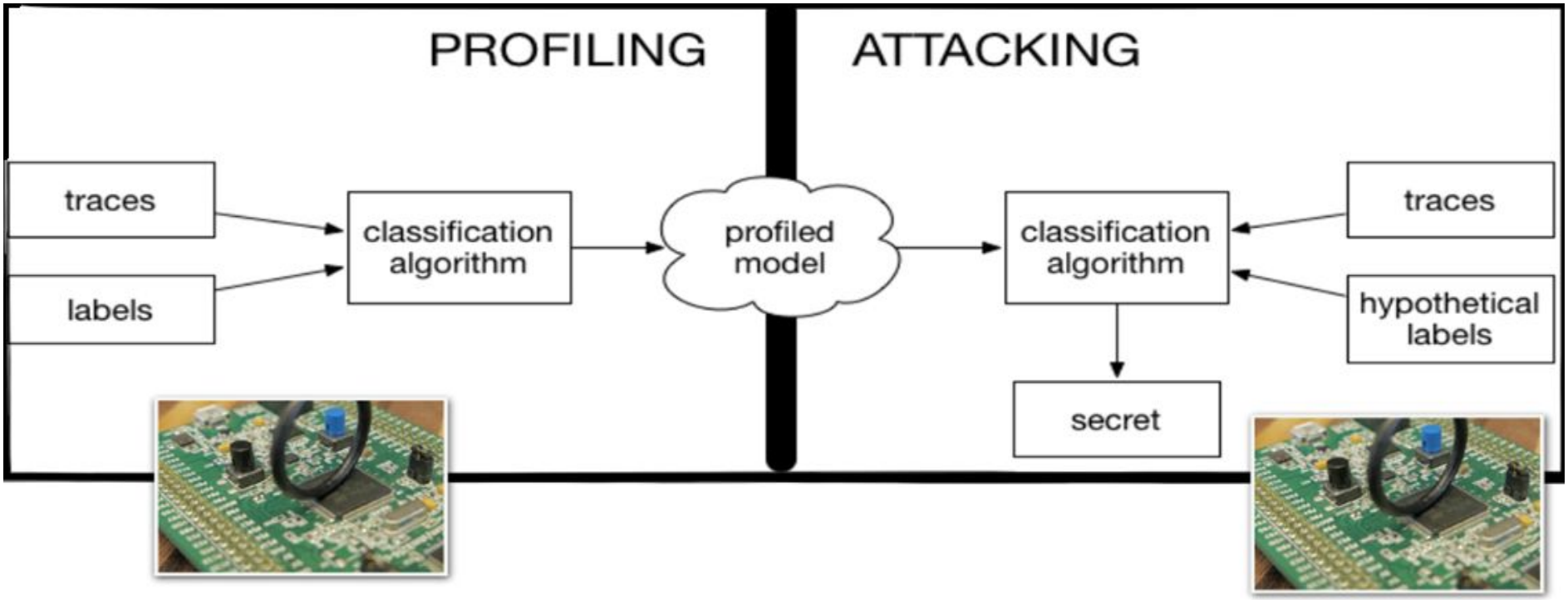


TVLA_Protected_BKA



TVLA_Protected_HCA

Deep Learning for Side Channel Analysis



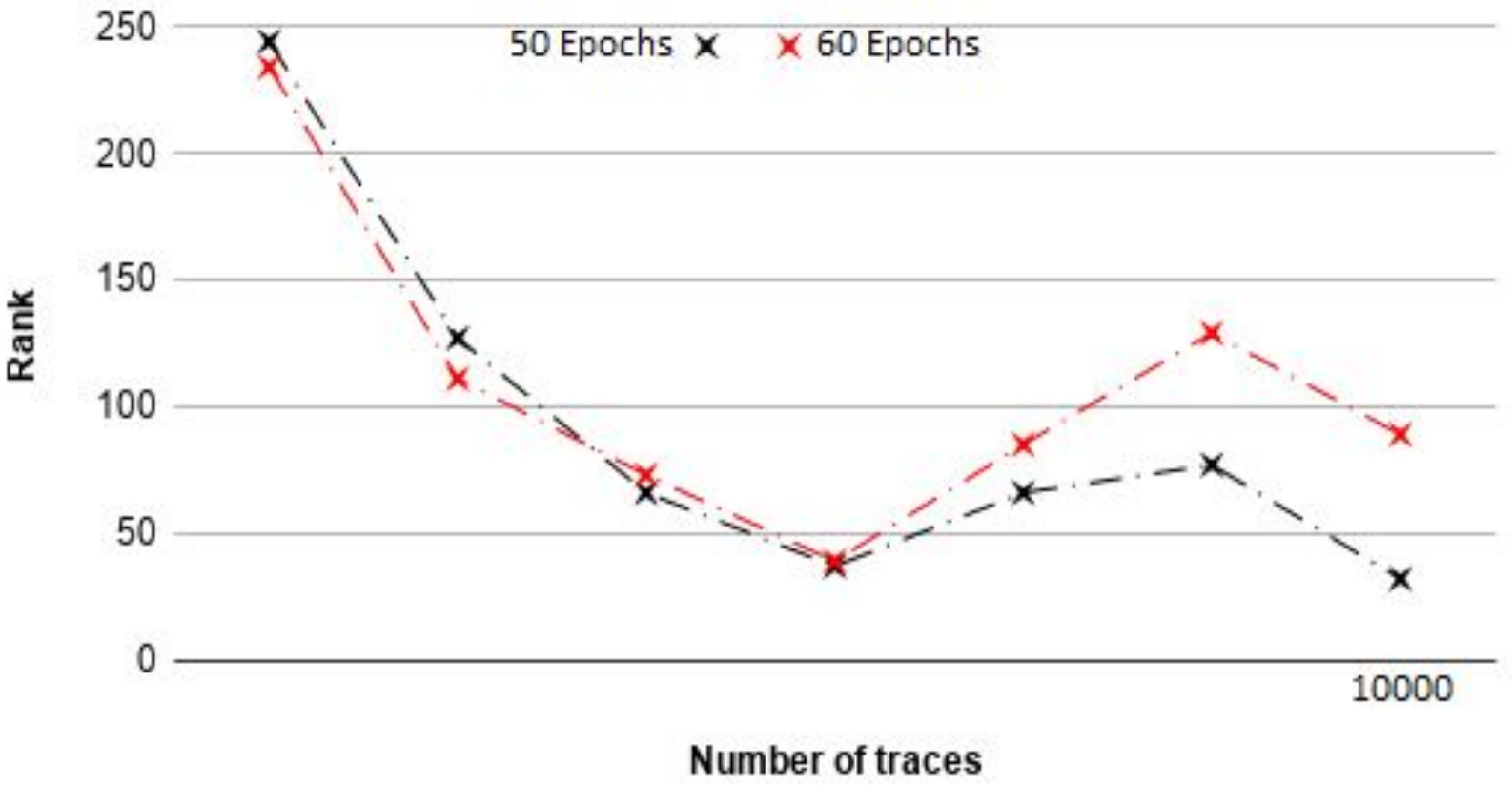
- The CNN models can be trained from one device and the testing on similar device using power traces
- The rank function can be evaluated to estimate the number of traces required to retrieve the key byte (where the rank converges to zero).
- It is possible to use one CNN model trained(more generic) on one key byte to recover the entire key bytes of a cipher.
- The trained CNN model can be used to evaluate the security level of the (un)protected realization of a cipher.

Evaluation of Countermeasure -DL

- A custom CNN architecture is developed.
- Network has 6 convolutional/pooling layers, and two fully connected layers, Kernel size/Filter : 11
- Activation function is a Leaky ReLU
- Pooling – Average Pooling
- No . of neurons in the fully connected layer : 4096
- Activation function for the output layer : SoftMax
- Loss function – Categorical Cross Entropy
- Learning rate - 0.00001, Number of epochs – 50, 60
  Batch size – 200

- Learning phase - 100000 traces, testing phase - 10000 traces.
- Secure upto 10000 traces. Maybe vulnerable above that limit.

Evaluation using Deep Learning



To Sum-up

- Generic Framework for crypto primitives evaluation.
- Our countermeasure of SPHINCS is better than a commercial product(SecurosysHSM) solution.

Thank you for your listening

For contact:
varun24ap@gmail.com
dillibabu@setsindia.net