Practical Evaluation of Protected RNS Scalar Multiplication

CHES 2019

By

Louiza Papachristodoulou

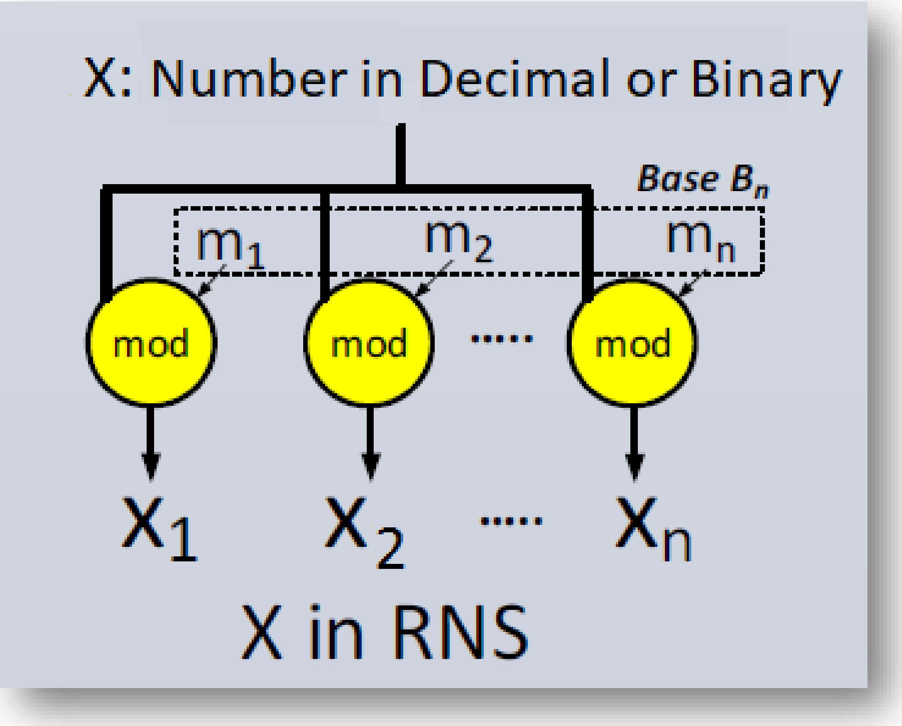
Joint work with

A. Fournaris, K. Papagiannopoulos, L. Batina

Outline

- Residue Number System in Elliptic Curve Cryptography
- Proposed TVLA threshold calculation
- TVLA analysis
- Location and Data Dependent Template Attacks
- Conclusions

Residue Number System

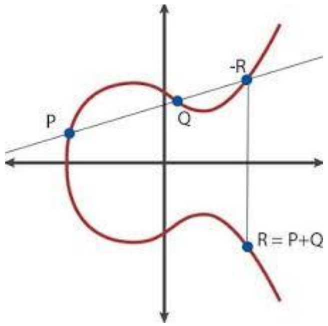


$$X = 50$$

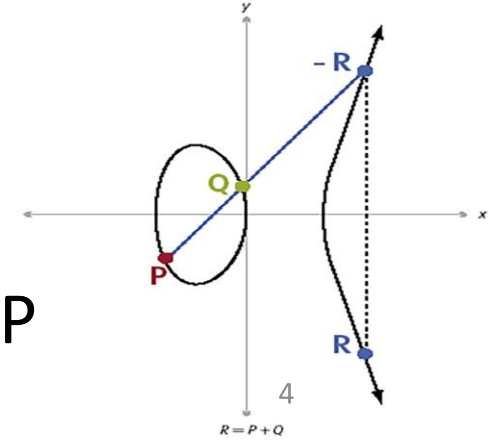
$$(m1, m2, m3) = (3, 7, 11)$$

$$(x1, x2, x3) = (2, 1, 6)$$

RNS in Elliptic Curve Cryptography



- Elliptic curves defined over prime fields GF(p)
- Modular operations turn easily to RNS modular operations over GF(p)
- RNS mod multiplication usually realized through RNS Montgomery multiplication to avoid modular inversion, but includes base extension
- EC scalar multiplication is the critical operation Q = kP



LRA Montgomery Power Ladder

Choose base B_n, B'_n. Transform V, R to RNS format using permutation p_t

- $R_0 = R,\ R_1 = R + V, R_2 = -R$
- Convert R_0 , R_1 , R_2 to Montgomery format
- For i= t-1 to 0
  - $R_2 = 2R_2$ in permutation p_t
  - If k_i = 1

    R_0 = R_0 + R_1 and R_1 = 2R_1 in permutation p_t

    else

    R_1 = R_0 + R_1 and R_0 = 2R_0 in permutation γ_t
- Integrity check: if i,k not modified and $R_0 + V = R_1$ then ret. $R_0 + R_2$ in permutation γ_t

  else ret. random value

Transform $R_0 + R_2$ to binary format

Test Vector Leakage Assessment (TVLA)

- Statistical tests between two trace-sets of acquisition

- Welch's t-test to evaluate if two sets have significant statistical differences

$$s_i = \frac{L_{i,A} - L_{i,B}}{\sqrt{\frac{\sigma_{i,A}^2}{n_A} + \frac{\sigma_{i,B}^2}{n_B}}}$$

- Values above ±4.5, indicates leakage, but TVLA does not exploit it

t-test Threshold Calibration for TVLA

Input nt_A , nt_B : number of traces for groups A,B
n_s : number of samples
σ_A, σ_B : sampled standard deviation
Output Threshold value for Welch's t-distribution th_t

$nt_A = nt_B = 4 * 10^3 - 10 * 10^3$
$n_s = 4 * 10^5 - 8 * 10^5$
$\sigma_A = 9.7, \sigma_B = 6.1$

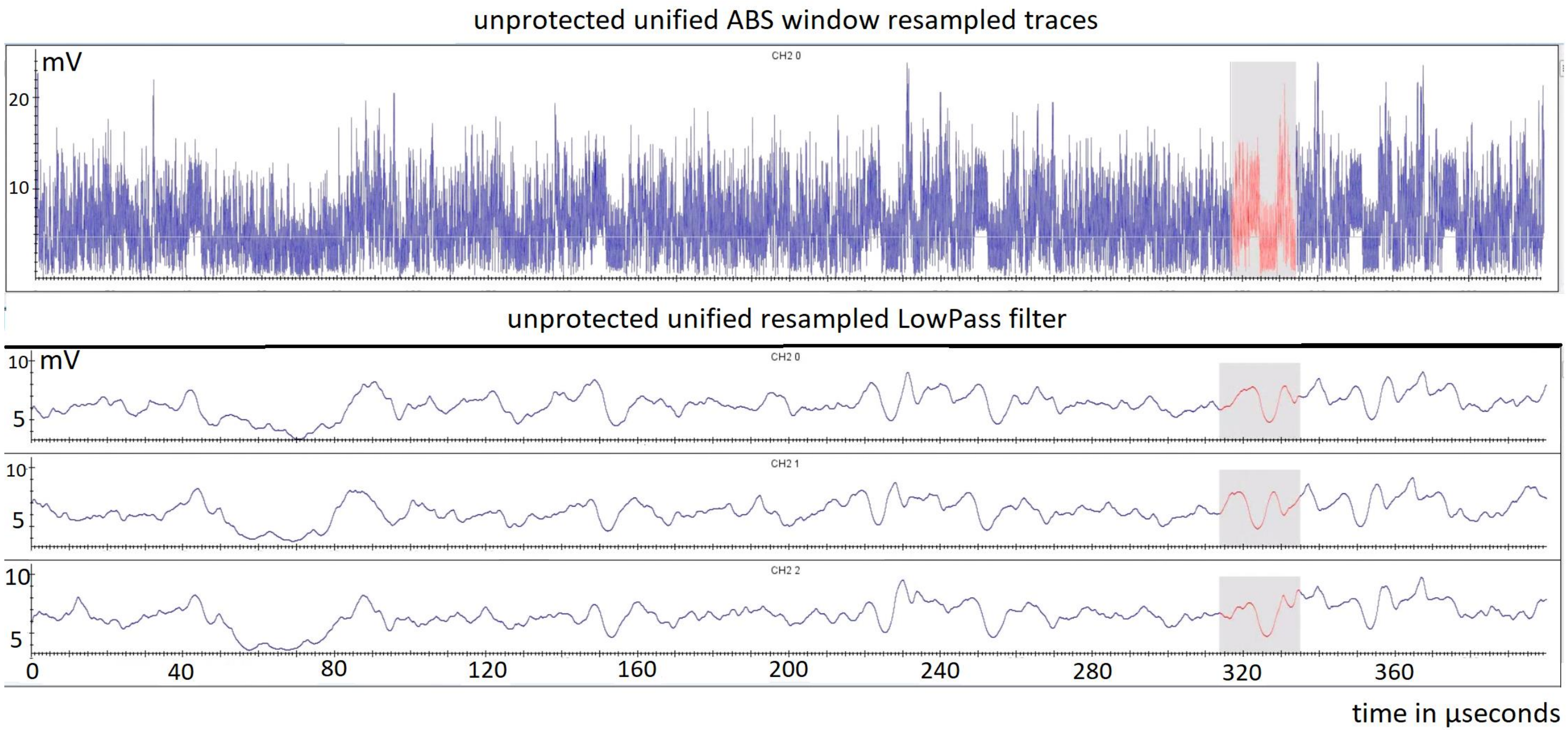
1. Choose level of significance α. Here α=0.00001
2. Family-wise error rate fwer = $(1 - a)^{n_s}$
3. Šidak correction $sidak_a$= 1 - $(1 - a)^{(1/n_s)}$
4. df = $\left(\frac{\sigma_A^2}{nt_A} + \frac{\sigma_B^2}{nt_B} \right)^2 / \left(\frac{\left(\frac{\sigma_A^2}{nt_A}\right)^2}{nt_A - 1} + \frac{\left(\frac{\sigma_B^2}{nt_B}\right)^2}{nt_B - 1} \right)$
5. Threshold th_t= |tinv (1- $sidak_a$ /2, df)|

$th_t = \pm 6.3$

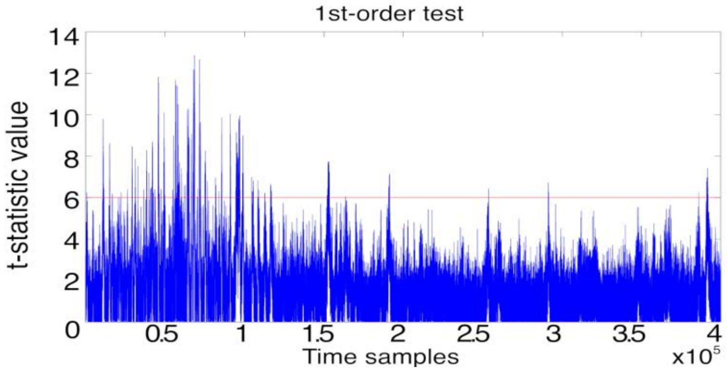
RNS implementation on BeagleBone

- C Software implementation on ARM Cortex A8
- RNS Montgomery multiplication
- Dedicated and Unified Group Law
- 5 different variations: unprotected, randomized scalar, random input point, random base permutations (LRA), random order of operations

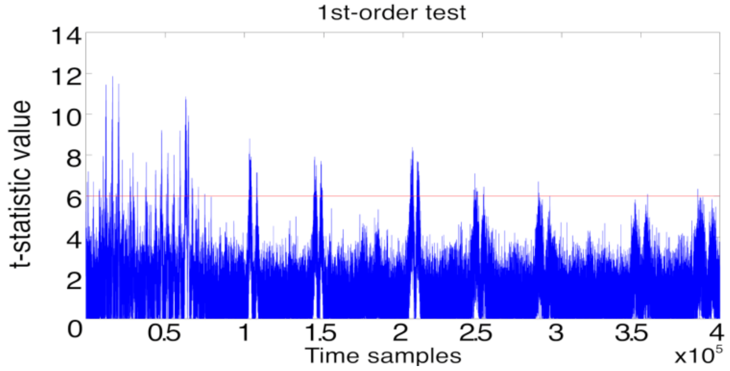
Processing of Traces – Low Pass Filter



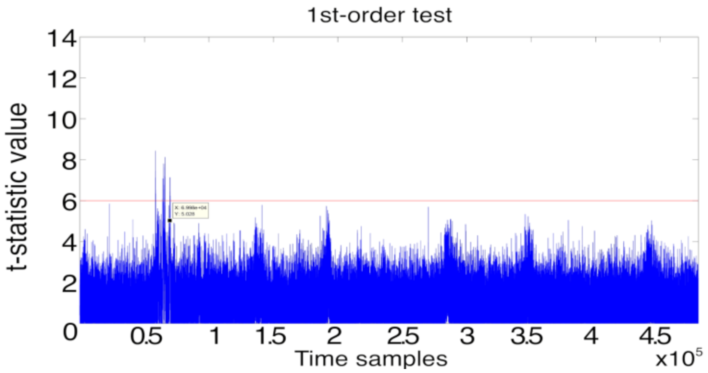
t-test random vs fixed scalar on twisted Edwards curve (a=1, d=2, p= $2^{192} - 2^{64} - 1$)



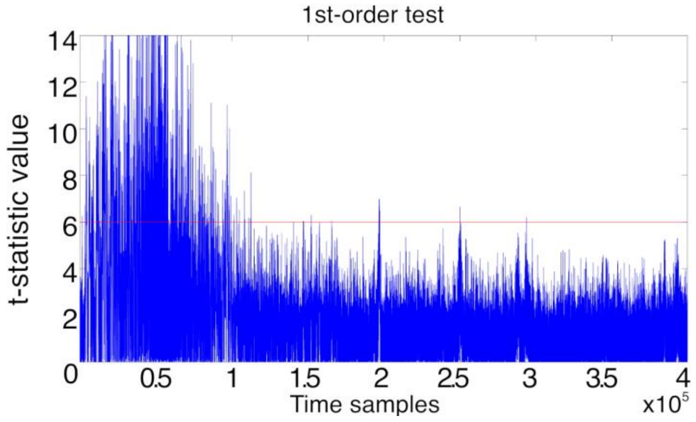
Unprotected scalar mul



Randomized scalar

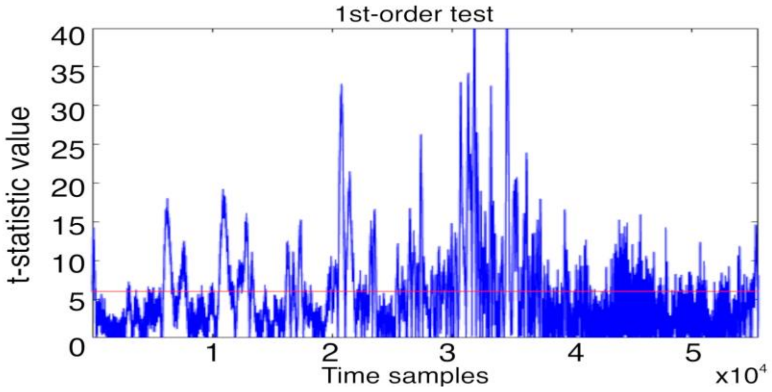


LRA

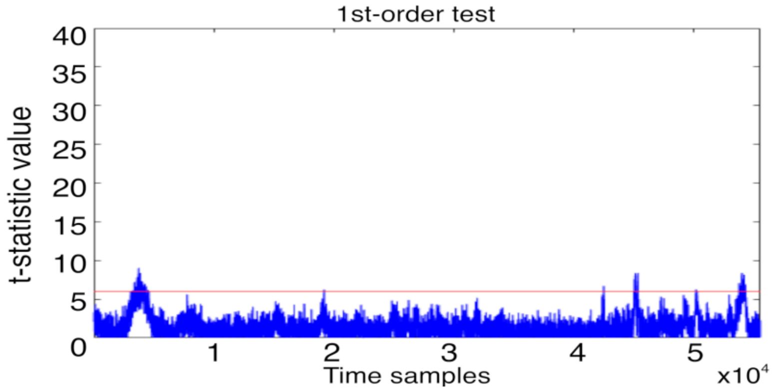


LRA_rdm_point()

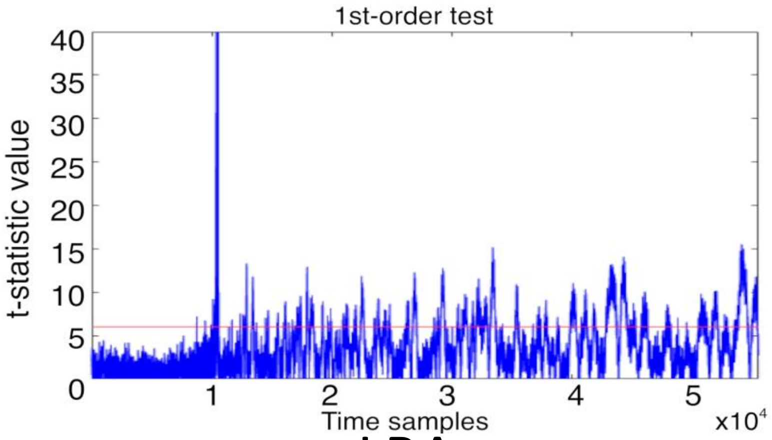
t-test random vs fixed point on secure Edwards curve (a=107, d=47, h=4, p= $2^{192}-2^{64}-1$)



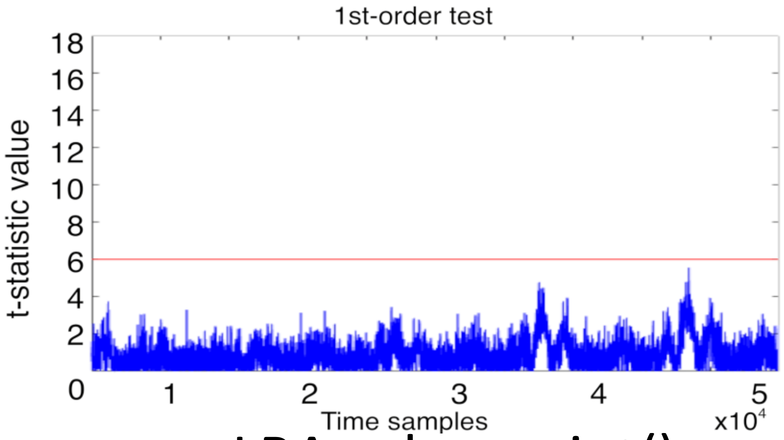
Unprotected scalar mul



Randomized scalar



LRA



LRA_rdm_point()

Data Dependent Template Attacks

- The value of a secret variable can be monitored
- Trigger around the key-dependent assignment (if-statement)

  If $k_i = 1$: $R_0 = R_0 + R_1$ and $R_1 = 2R_1$

  Else: $R_1 = R_0 + R_1$ and $R_0 = 2R_0$
- After alignment, 20k traces. Used half for templates, half for classification
- Success rate 90-91% for the unprotected case, 82-97% for LRA countermeasure activated
- Scalar randomization (65-72%) and LRA randomized RNS operations (55-58%) are good countermeasures

Location Dependent Template Attacks

- Templates created for storage structure that handles the key-dependent instruction (doubling)

  If k_i = 1: $R_0 = R_0 + R_1$ and $R_1 = 2R_1$

  Else: $R_1 = R_0 + R_1$ and $R_0 = 2R_0$

- Template classification: 95-99.9%

- LRA with randomized operations: 70-83%

Location Dependent Leakage

- Registers are not really single registers, RNS values are stored in 50-bit chunks - result of doubling is stored in different memory locations
- Location dependent leakage was not an expected result
- The normal distributions for k_i = 0 and k_i = 1 for every variation of the implementation are very different ($N(-24.3, 9, 7)$, $N(19.6, 6.1)$)
- Leaky platform - capacitors next to each other
- Scalar randomization not an efficient countermeasure
- LRA with randomized operations makes template attacks harder

Evaluation Table

Algorithm	Welch t-test r-vs-f scalar	Welch t-test r-vs-f point	TA Data	TA Location	PO
unprotected	✗	✗	✗	✗	0%
rdm_point	✗	✗	✗	✗	52%
LRA	✗	✗	✗	✗	50%
protected_LRA	✗	✓	✗	✗	110%
unprotect_rdm_scal	✓	N/A	✓	✗	19%
rdm_point_rdm_scal	✓	N/A	✓	✗	54%
LRA_rdm_scalar	✓	N/A	✓	✗	51%
protected_rdm_scal	✓	N/A	✓	✓	110%
unprotect_unified	✗	✗	✗	✗	19%
rdm_point_unified	✗	✗	✗	✗	99%
LRA_unified	✗	✗	✗	✗	72%
protected_unified	✓	✓	✗	✗	144%
LRA_nc_rdm_operat	✗	✓	✓	✓	76%
LRA_nc_rdm_operat _rdm_scalar	✓	N/A	✓	✓	76%

✓ Pass t-test/secure against templates

✗ Fail t-test/not secure against templates

Conclusions

- TVLA bounds not rigid; compute according to distribution of traces, number of samples, number of traces
- Randomization of scalar, input point, regularity of MPL are good countermeasures but not enough to avoid leakage
- Different RNS representations do not lower the template success rates
- Randomization of RNS operations protects against templates and less expensive compared to randomization of input point
- Classification using ML algorithms
- Evaluation on an FPGA would give further insights in the security of RNS

THANK YOU FOR YOUR ATTENTION !

louiza@cryptologio.org