SPA on NTRU software implementation

Rabas Tomas with Jiri Bucek and Robert Lorencz

Czech Technical University (CTU) in Prague, Faculty of Information Technology (FIT), Department of Information Security (DIS)

May 24, 2022

NTRU

Post-quantum candidate for NIST standard

Authors: Jeffrey Hoffsteinem, Jill Pipherovou a Josephem H. Silvermanem (1996)

Polynomial ring $\mathcal{R} = \mathbb{Z}[x]/(x^N - 1)$

An element $f \in \mathcal{R}$ **can be written as** $f = \sum_{i=0}^{N-1} f_i x^i$

Coprime integers p and q where $p << q$
(p is 3, q is 2048 or 4096)

f_q is polynom f with reduced coefficients modulo q

Key Generation

f and g are random ternary polynomials with fixed number of coefficients 1 and -1

$h = pf_q^{-1}g \bmod q$

f is the priv. key, h is the pub. key

Encryption

Input: random r**, message** m**, public key** h

$e = rh + m$ **mod** q

Decryption

Input: ciphertext e**, private key** f

$a = fe$ **mod** q

$m = af_p^{-1}$ **mod** p

Remark: f **is chosen so that** f_p^{-1} **equals 1**

Correctness

$$
\begin{aligned}
a &= e \cdot f \bmod q \\
&\overset{e=rh+m}{=} r \cdot h \cdot f + m \cdot f \bmod q \\
&\overset{h=pf_q^{-1}g}{=} r \cdot pf_q^{-1}g \cdot f + m \cdot f \bmod q \\
&= pr \cdot g + m \cdot f \bmod q
\end{aligned}
$$

$$
\begin{aligned}
a \cdot f_p^{-1} &= (pr \cdot g + m \cdot f) \cdot f_p^{-1} \bmod p \\
&= 0 + m \cdot f \cdot f_p^{-1} \bmod p \\
&= m \bmod p
\end{aligned}
$$

Power Analysis

We want to find the secret key f...

... by observing power consumption of the device

We look at the decryption algorithm (where private key is used)

Essentially, it is just a polynomial multiplication modulo q.

Implementation

Suggested in: An, Soojung, Suhri Kim, Sunghyun Jin, Hanbit Kim and Hee-Seok Kim. “Single Trace Side Channel Analysis on NTRU Implementation.” Applied Sciences 8 (2018): 2014

Implementation was proposed as a countermeasure against power analysis
(dispose of leakage based on different instructions)

Our code based on StrangSwan implementation (https://www.strongswan.org/)

The private key is sparse (values 1,-1 or 0)

The implementation use sparse encoding b of the private key (saves the space)

Stores increasing coefficients of 1 and -1

Assume $f = 1 - x^1 + x^2 + x^3 - x^4 + x^7 - x^8$, then we encode it as $b = [0, 2, 3, 7, 1, 4, 8]$.

Require: cipher-text polynomial $e \in R$ and encoding b of private key F

Ensure: $H = Fe \pmod{q}$

1. **for** $i = 0; i < N; i$++
2. $\quad t_i \leftarrow r$, r is a random value
3. **end for**
4. **for** $j = d_f + 1; j < 2d_f + 1; j$++ **do**
5. $\quad k \leftarrow b_j$
6. $\quad y \leftarrow N - k$
7. $\quad$ **for** $i = 0; i < N; i$++, y++ **do**
8. $\quad\quad t_i \leftarrow t_i + e_y$
9. $\quad$ **end for**
10. **end for**
11. **for** $i = 0; i < N; i$++ **do**
12. $\quad t_i \leftarrow -t_i$
13. **end for**
14. **for** $j = 0; j < d_F + 1; j$++ **do**

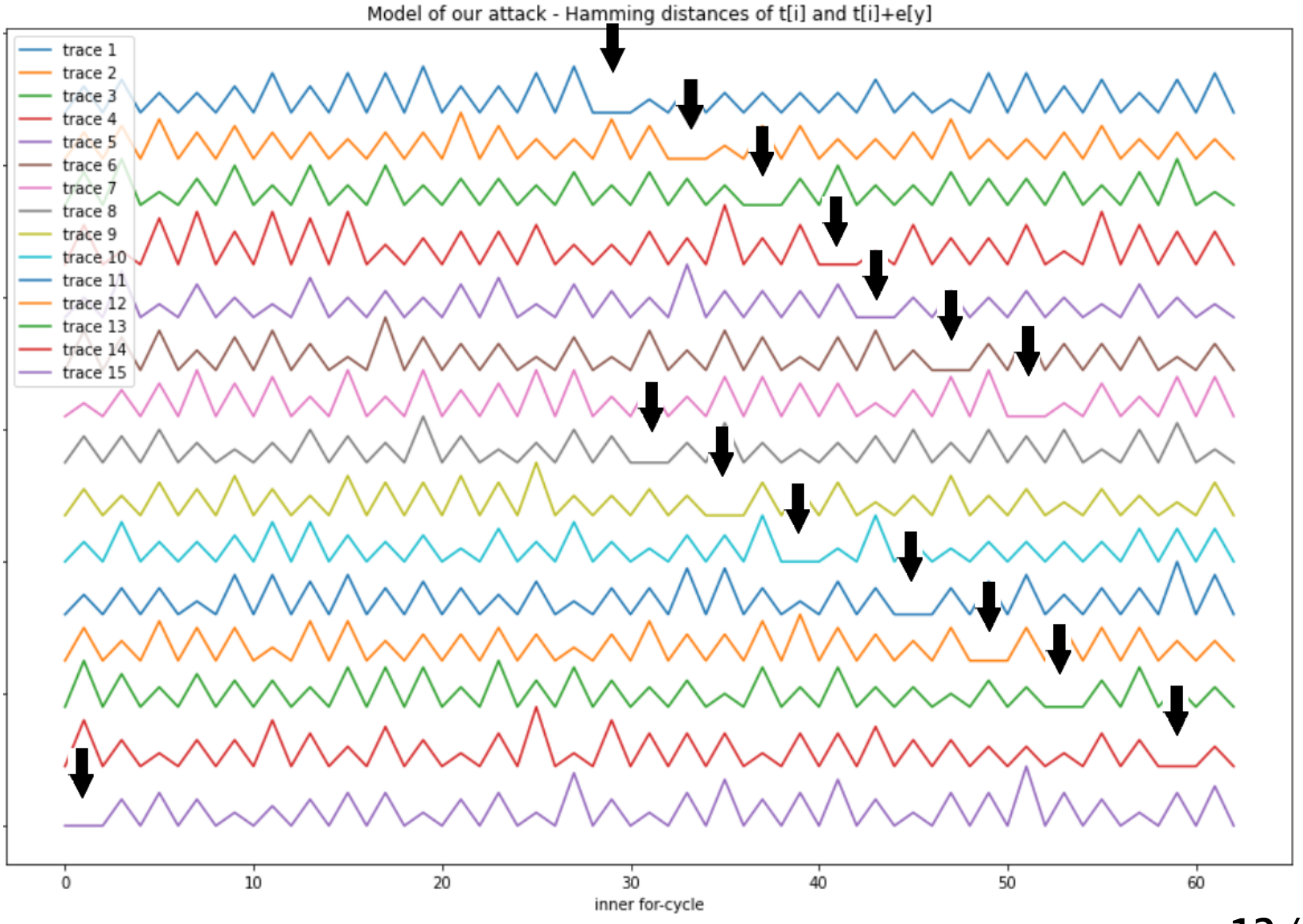
⋮

Visualization

$$
\begin{array}{lll}
(\ e_{N-1}e_{N-2}\ \cdots\ e_1e_0\ )\ e_{N-1}e_{N-2}\ \cdots\ e_1e_0 & | & \text{ciphertext stored in doubled way} \\
\times\ b_{2*df}\ \cdots\ b_{df+1}b_{df}\ \cdots\ b_1b_0 & & \\
\hline
r\ \cdots\ r & | & \text{initialization step of array } t \\
+\ e_{N-b_{df+1}} & | & \text{step } 1.0\ ,\ b_{df+1}\ (=:k) \\
+\ e_{N-b_{df+1}+1} & | & \text{step } 1.1\ ,\ b_{df+1} \\
+\ e_{N-b_{df+1}+2} & | & \text{step } 1.2\ ,\ b_{df+1} \\
\cdots & \vdots & \\
+\ e_{2*N-b_{df+1}-1} & | & \text{step } 1.N\ ,\ b_{df+1} \\
 & & \\
+\ e_{N-b_{df+2}} & | & \text{step } 2.0\ ,\ b_{df+2}\ (=:k) \\
\cdots & \vdots &
\end{array}
$$

Where is the leakage

$$
\begin{array}{rcccccccccccccl}
 & (& e_{N-1} & \cdots & 0 & \cdots & e_0 &) & e_{N-1} & \cdots & 0 & \cdots & e_0 & | & \text{Lets say there is a zero...} \\
 & & & & & \times & & & b_{2*df} & \cdots & b_{df}\, b_{df-1} & \cdots & b_1\, b_0 & & \\
\text{This zero slowly shifts to the left...} & & & & & & & & & & & & & & \\
 & & r & r & & & & & & \cdots & & & r & r & | \ \text{initialization step of array } t \\
 & + & e_{N-k-1} & & & & & & & \cdots & 0 & \cdots & & e_{N-k} & | \ \text{step 1 , } b_{df+1} (=: k) \\
 & + & e_{N-k-1} & & & & \cdots & 0 & \cdots & & & & & e_{N-k} & | \ \text{step 2 , } b_{df+2} (=: k) \\
 & + & e_{N-k-1} & & \cdots & 0 & \cdots & & & & & & & e_{N-k} & | \ \text{step 3 , } b_{df+3} (=: k) \\
\text{This can be seen in power trace...} & & & & & & & & & & & & & & \vdots \ \text{step 4 , } b_{df+4} (=: k)
\end{array}
$$

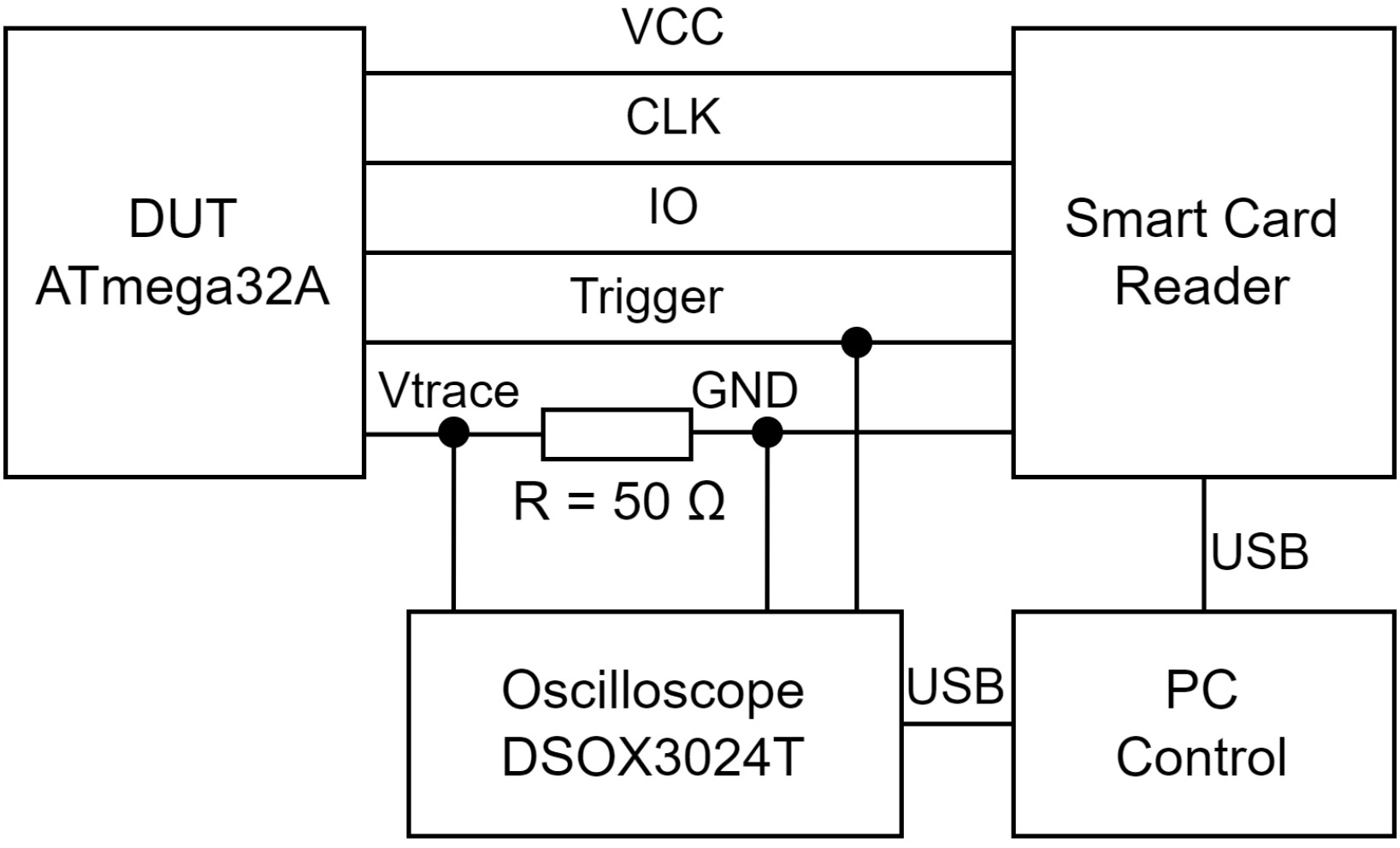

Model of our attack - Hamming distances of t[i] and t[i]+e[y]
trace 1
trace 2
trace 3
trace 4
trace 5
trace 6
trace 7
trace 8
trace 9
trace 10
trace 11
trace 12
trace 13
trace 14
trace 15
0
10
20
30
40
50
60
inner for-cycle

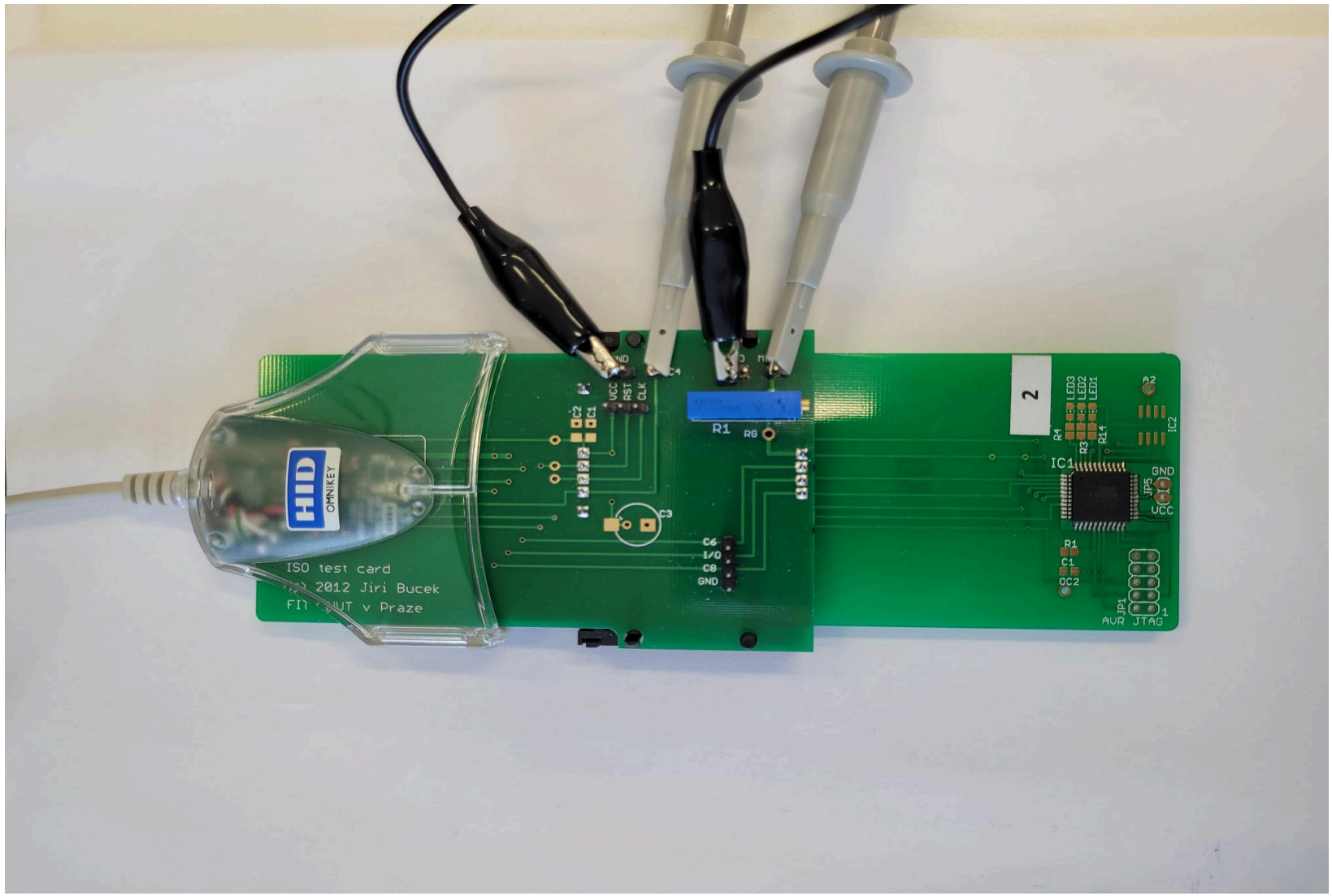
Target

8-bit microcontroller of the Microchip AVR family, namely ATmega32A

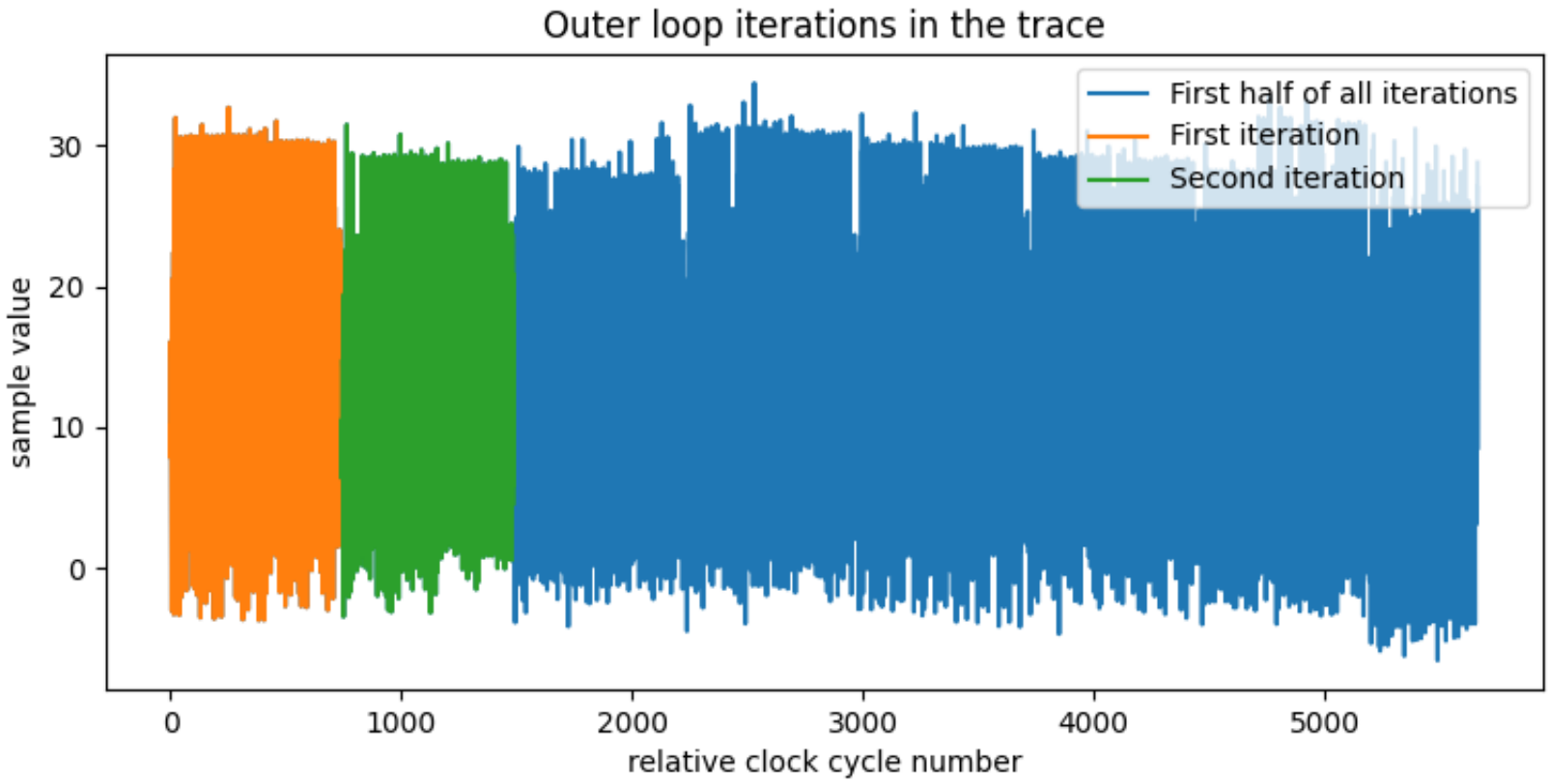
50Ω series resistor using a standard passive oscilloscope probe

Keysight DSOX3024T oscilloscope connected to the controlling PC via USB


VCC
CLK
IO
Trigger
Vtrace
GND
R = 50 Ω
DUT
ATmega32A
Smart Card
Reader
USB
Oscilloscope
DSOX3024T
USB
PC
Control


HID
OMNIKEY
ISO test card
2012 Jiri Bucek
Praze
VCC
RST
CLK
R1
RG
C1
C2
C3
C4
C6
I/O
C8
GND
IC1
IC2
LED3
LED2
LED1
R4
R3
R14
GND
VCC
JP5
JP1
AVR JTAG
2

Experimental results



How to get rid of noise...

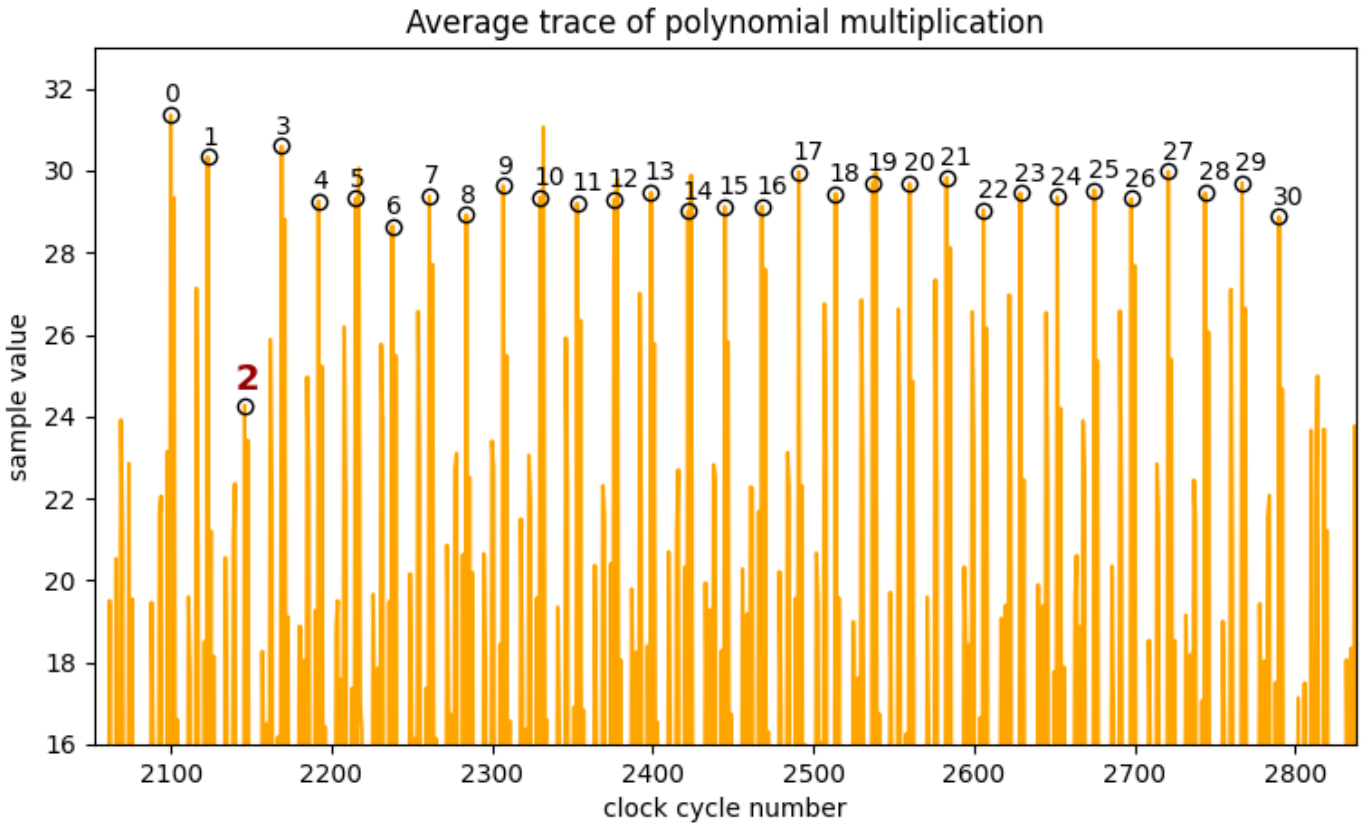
We use more traces and compute mean from them

We choose convenient ciphertext as 0 following with high hamming weight values (255)

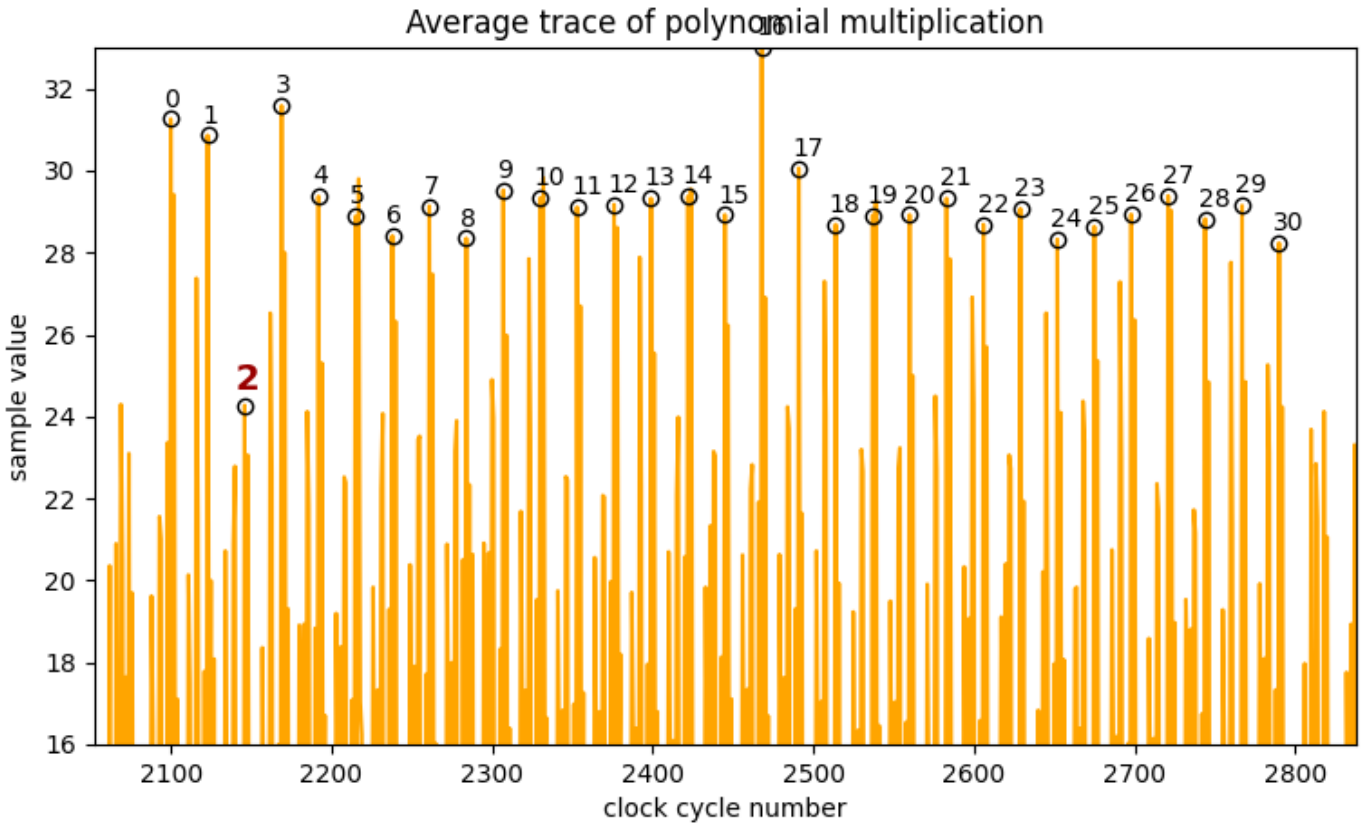
Then we get nice results :)

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

10 traces average

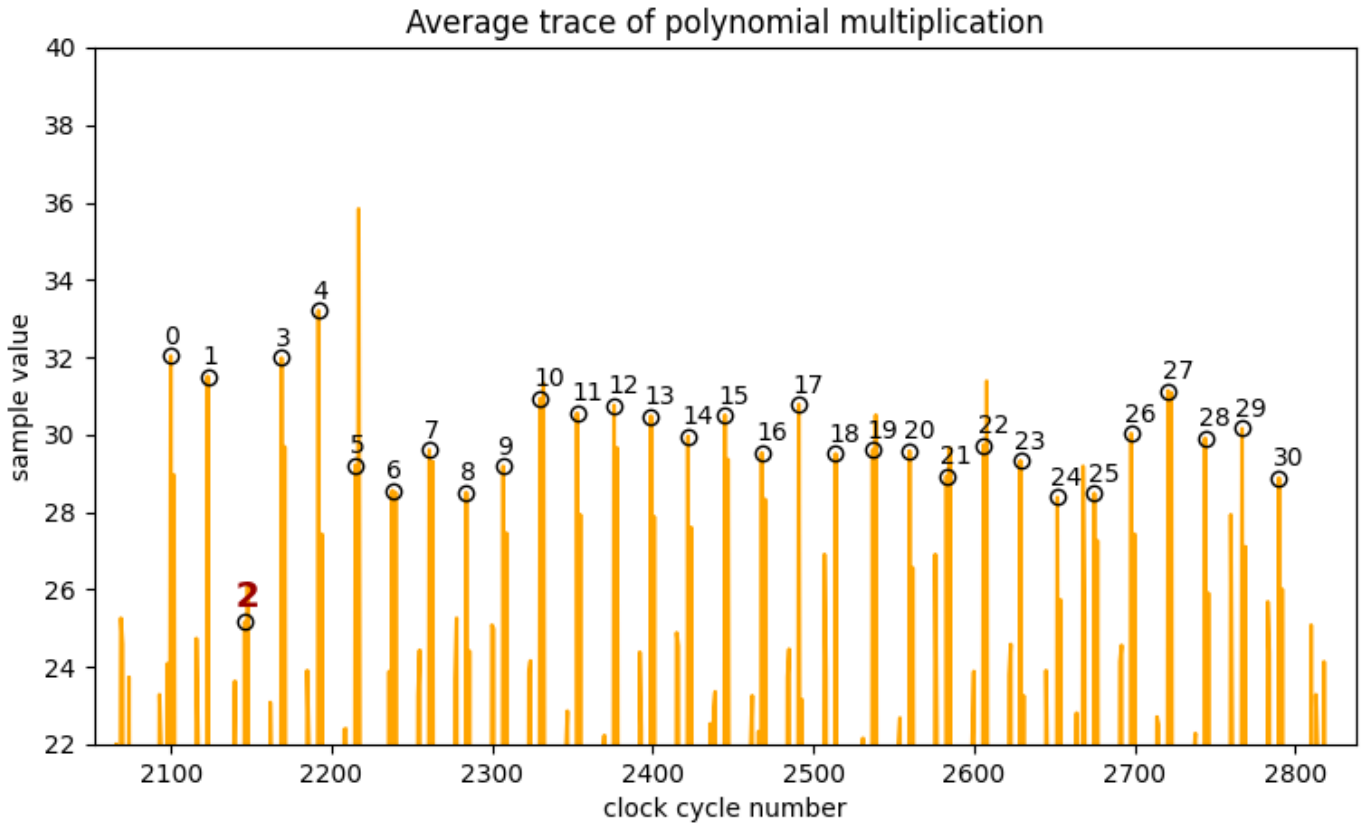


4 traces average

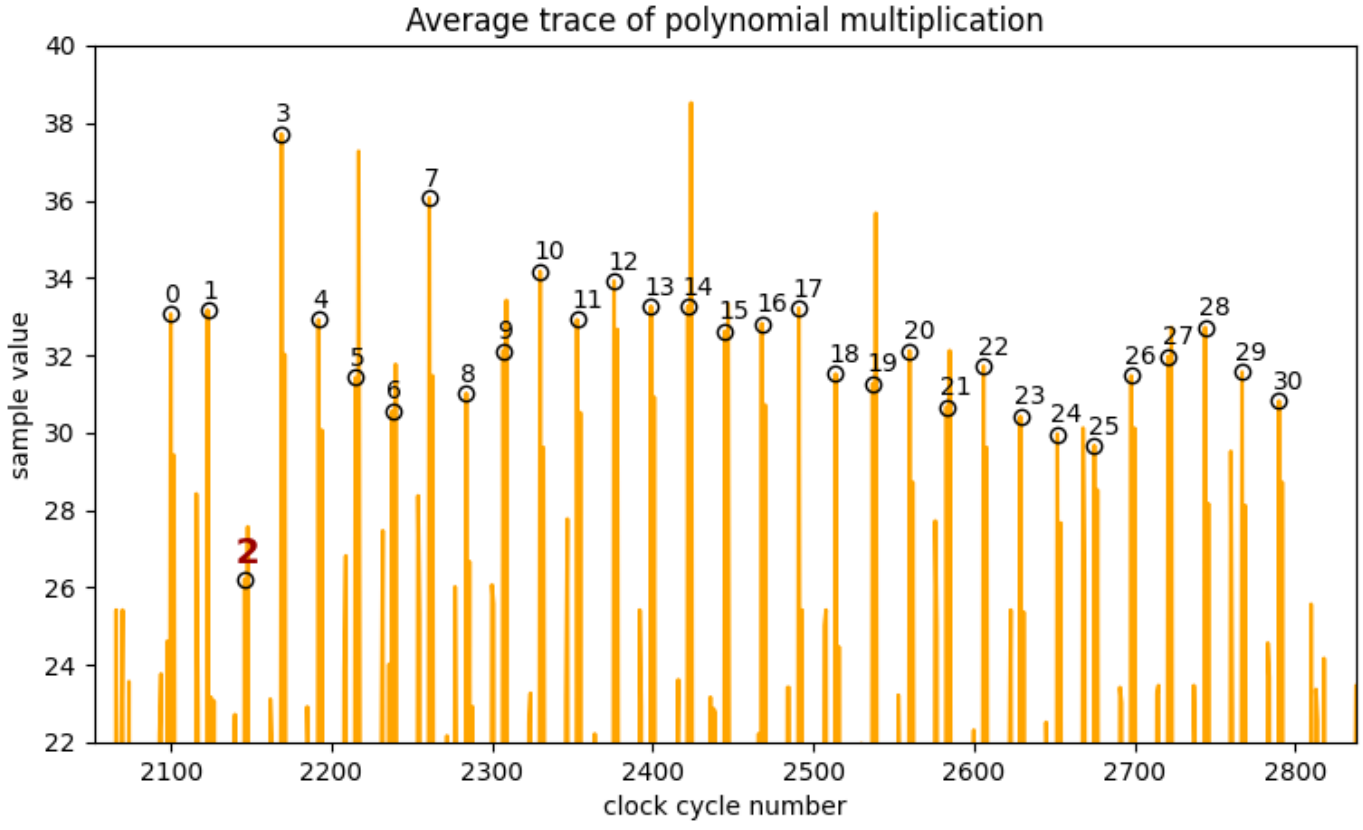


CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

2 traces average



1 trace


Average trace of polynomial multiplication
sample value
clock cycle number
40
38
36
34
32
30
28
26
24
22
2100
2200
2300
2400
2500
2600
2700
2800
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Conclusion

We discovered leakage depending on input data in the protected implementation

Successfully verified by experiment for chosen ciphertext on AVR 8-bit microcontroller as the target

Future work?

Use 32-bit microcontroller

Use real value parameters (not smaller)

Combine with classical cryptoanalysis to achieve good results even for random ciphertext and just one trace

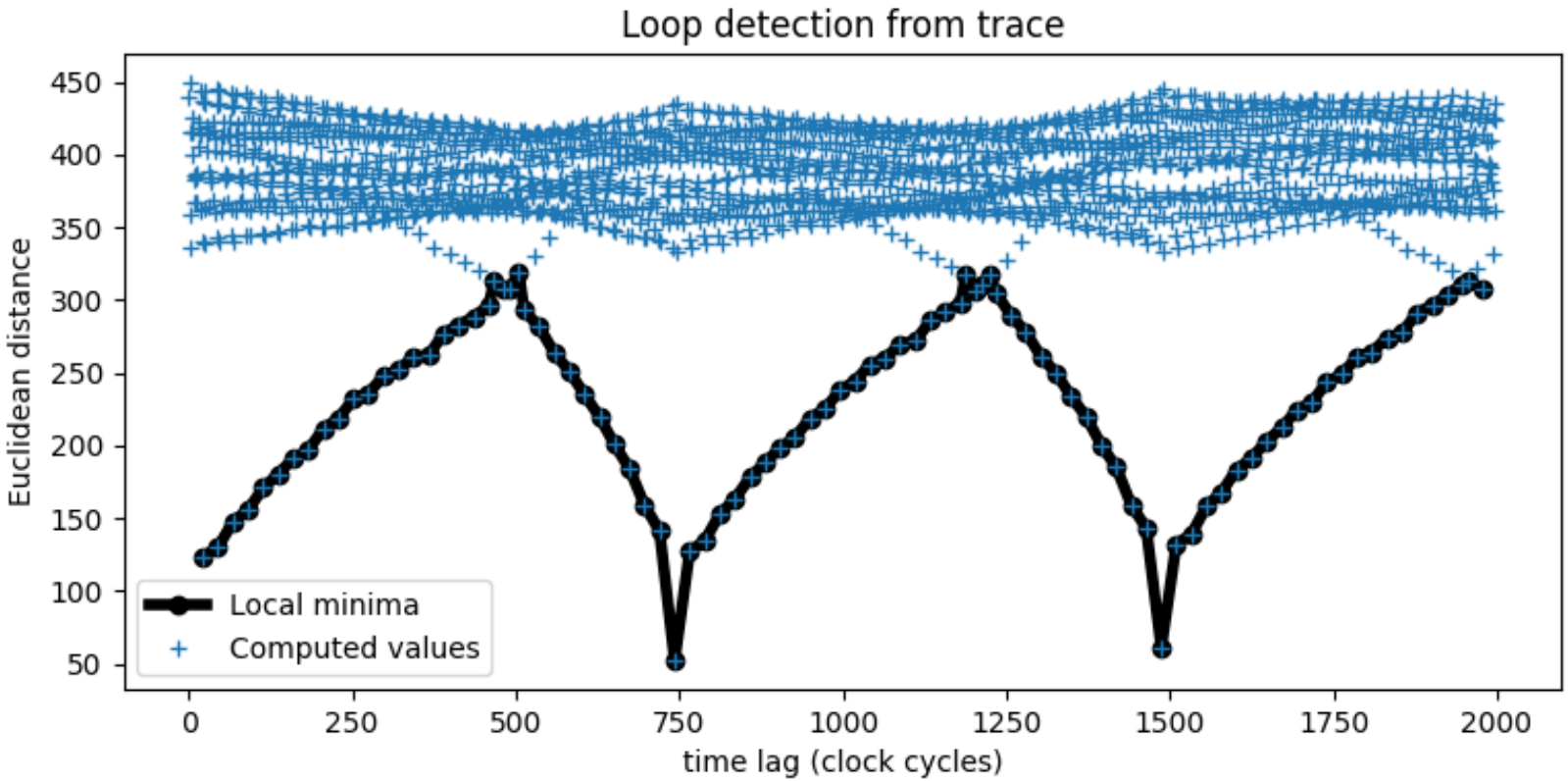
Thanks for attention

This work was supported by the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16 019/0000765 "Research Center for Informatics"


CTU
CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Appendix



Bugs in original implementation:

Parity of N

Subtraction of r in the end

Indeces out of range
-> double array e

Algorithm 4 Countermeasure of NTRU Open Source Proeject

Require: cipher-text polynomial $e \in R$ and coefficient location indices of private key b
Ensure: $H = F \cdot e \pmod{q}$

for $i = 0; i < N; i$++ **do**
  $t_i \leftarrow r$ ▷ r is a random value
end for
for $j = d_f + 1; j < 2d_f + 1; j$++ **do**
  $k \leftarrow b_j$
  $x \leftarrow \frac{N-1}{2} - k, y \leftarrow N - k$
  $t_{\frac{N-1}{2}} \leftarrow t_{\frac{N-1}{2}} + e_x$
  $x \leftarrow x + 1$
  for $i = 0; i < \frac{N-1}{2}; i$++, x++,y++ **do**
    $t_{\frac{N}{2}+i+1} \leftarrow t_{\frac{N}{2}+i+1} + e_x$
    $t_i \leftarrow t_i + e_y$
  end for
end for
for $i = 0; i < N; i$++ **do**
  $t_i \leftarrow -t_i$
end for
for $j = 0; j < d_F + 1; j$++ **do**
  $k \leftarrow b_j$
  $x \leftarrow \frac{N-1}{2} - k, y \leftarrow N - k$
  $t_{\frac{N-1}{2}} \leftarrow t_{\frac{N-1}{2}} + e_x$
  $x \leftarrow x + 1$
  for $i = 0; i < \frac{N-1}{2}; i$++, x++,y++ **do**
    $t_{\frac{N}{2}+i+1} \leftarrow t_{\frac{N}{2}+i+1} + e_x$
    $t_i \leftarrow t_i + e_y$
  end for
end for
for $i = 0; i < N; i$++ **do**
  $H_i \leftarrow t_i - r \pmod{q}$
end for
return H