Secure Cloud Computing:
ORAM and Homomorphic Encryption

Riccardo Gennaro
Student ID: 3534219
Group 5

Péter Svelecz
Student ID: 3542629
Group 5

1. Path ORAM

1.1 Simulation results for Path ORAM

Following the requirements for this assignment, we tested our implementation for a number of blocks $N = 2^{15}$. The test consisted of two runs with a warmup of $3 * 10^6$ write accesses and an actual simulation of $3 * 10^6$ read accesses. The first simulation was carried out with a number of blocks per bucket $Z = 2$ while the second with $Z = 4$.

Following, you can find the results for $Z = 4$ formatted as per instructions.

```
For Z=4
    -1,3000000
    0,3000000
    1,2083373
    2,792131
    3,306820
    4,122796
    5,51130
    6,22367
    7,10150
    8,4804
    9,2381
    10,1240
    11,654
    12,359
    13,179
    14,70
    15,27
    16,6
    17,3
    18,1
```

Output can be found in files `simulation1.txt` and `simulation2.txt` for $Z = 2$ and $Z = 4$ respectively.

Given the lenght of the simulation for $Z = 2$, its output is not reported in this document.

1.2 Probability of stash overflow

For both simulations, we map the probabilty of stash overflow given a stash lenght contraint. Following the results for $Z = 2$ and $Z = 4$ respectively.

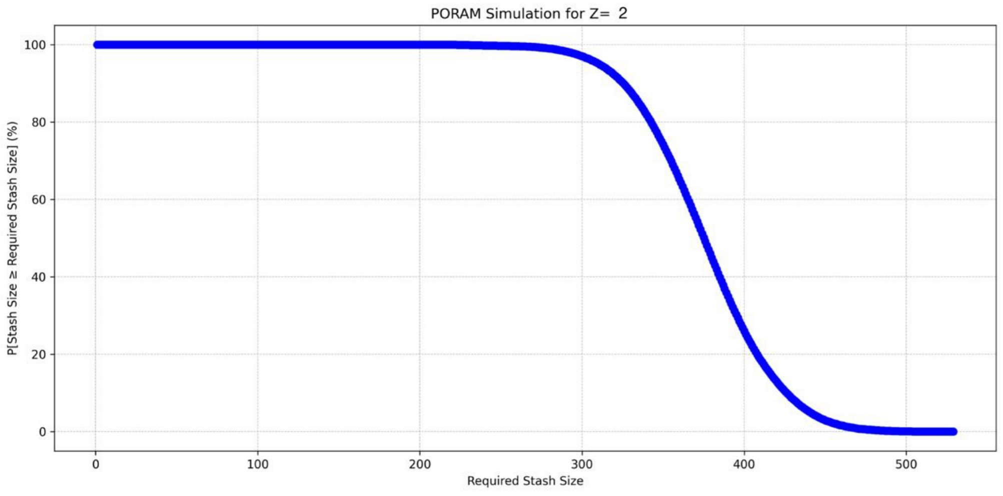

Figure 1: Probabilty of stash overflow for $Z = 2$.

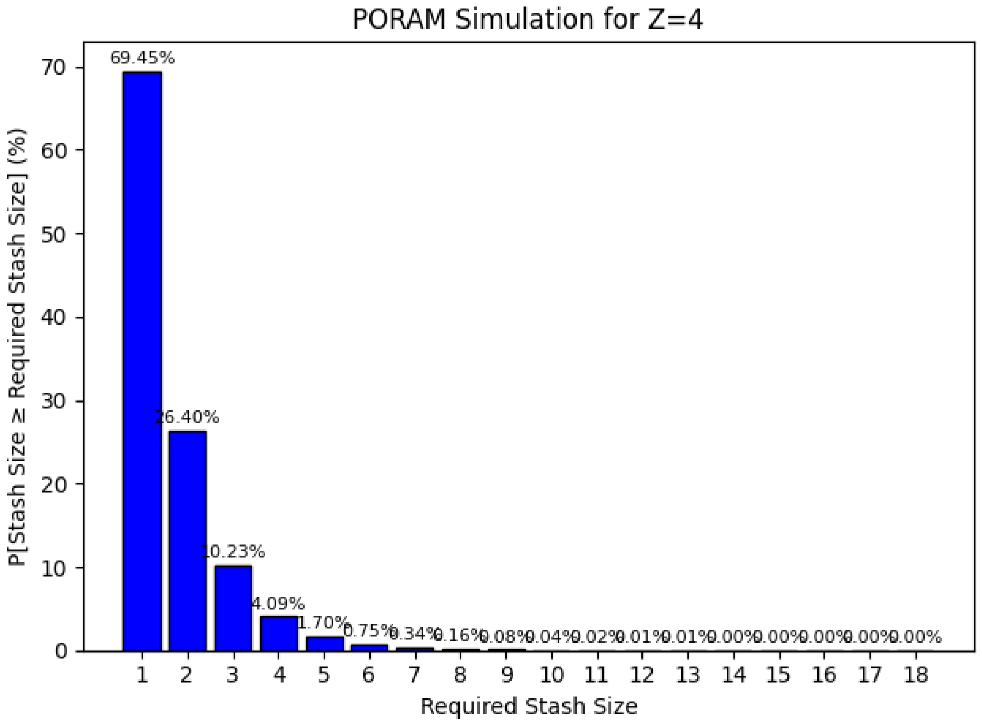

Figure 2: Probabilty of stash overflow for $Z = 4$.

2. Homomorphic Encryption over the Integers

2.1 Encrypting a bit vector

We encrypted the bit vector provided in the assignment and added the resulting ciphertexts to the `swhe-task1.json` file. The ensure the correct way of encryption we made sure to use the quotient and modulo functions as defined in the scheme and described in the assignment text. Below you can find the resulting ciphertexts, each belonging to a bit message.

2.2 Number of supported operations

We tested the possible number of operations (homomorphic XOR and AND) for each ciphertext, using the provided parameters for the scheme. The reason behind the significant difference between the number of supported XOR and AND operations lies in the way these operations are implemented. Since XOR is implemented as integer addition, the generated noise during this operation accumulates way more slowly than iin the case of the AND operation which is implemented via integer multiplication (in both cases a $modx_0$ operation is also performed to decrease the generated noise). Because of this we were expecting similar results but it was interesting to see the actual numbers, as the difference is multiple orders of magnitude.

Following you can find the results for the number of supported operations.

```
Ciphertext 1 (Noise Bitlength: 1) supports:
  XOR operations: 40701
  AND operations: 5
Ciphertext 2 (Noise Bitlength: 3) supports:
  XOR operations: 105669
  AND operations: 3
Ciphertext 3 (Noise Bitlength: 5) supports:
  XOR operations: 12728
  AND operations: 1
Ciphertext 4 (Noise Bitlength: 7) supports:
  XOR operations: 62678
  AND operations: 2
Ciphertext 5 (Noise Bitlength: 9) supports:
  XOR operations: 75964
  AND operations: 1
```