TU Graz

SCIENCE
PASSION
TECHNOLOGY

Masking Hybrid Homomorphic Encryption Schemes

Emira Salkić

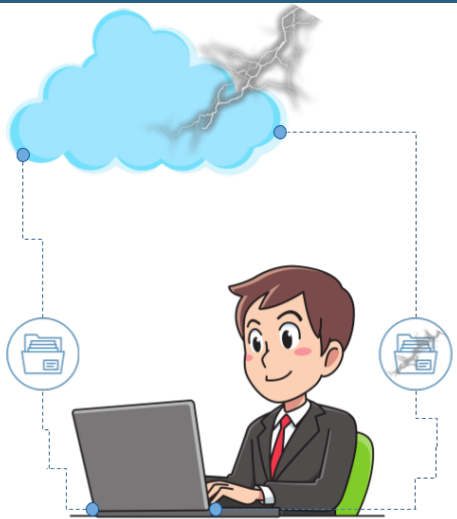
13.02.2025

Introduction

Introduction to Homomorphic Encryption

Introduction to Homomorphic Encryption

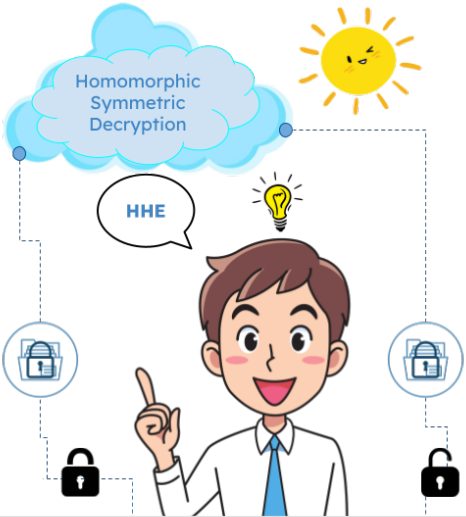
Introduction to Homomorphic Encryption



Homomorphic Encryption

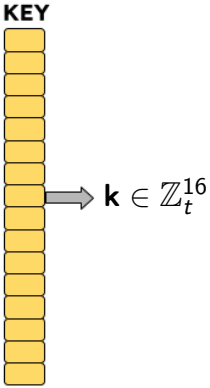
Homomorphic Encryption

Hybrid Homomorphic Encryption



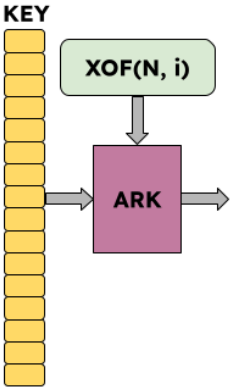
Hybrid Homomorphic Encryption

Hybrid Homomorphic Encryption: HERA



ASIACRYPT 2021: Transciphering Framework for Approximate Homomorphic Encryption, Jihoon Cho, Jincheol Ha, Seongkwang Kim, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon

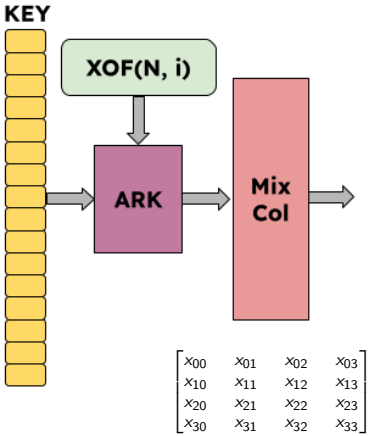
Hybrid Homomorphic Encryption: HERA



$\mathbf{rc} = (\mathbf{rc}_0, \ldots, \mathbf{rc}_r) \in (\mathbb{Z}_t^{16})^{r+1}$

$\mathrm{ARK}[\mathbf{k}, nc, i](\mathbf{x}) = \mathbf{x} + \mathbf{k} \bullet \mathbf{rc}_i$, *where* $\mathbf{x} \in \mathbb{Z}_t^{16}$

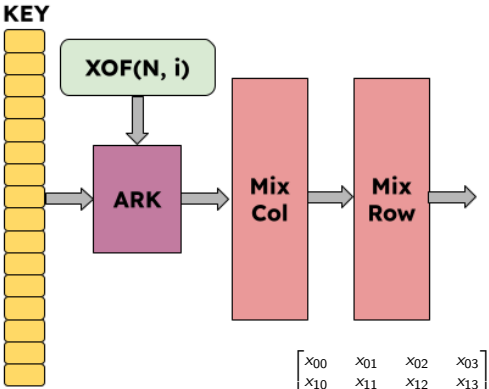
Hybrid Homomorphic Encryption: HERA



$$\begin{bmatrix} y_{0c} \\ y_{1c} \\ y_{2c} \\ y_{3c} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_{0c} \\ x_{1c} \\ x_{2c} \\ x_{3c} \end{bmatrix}$$

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix}$$

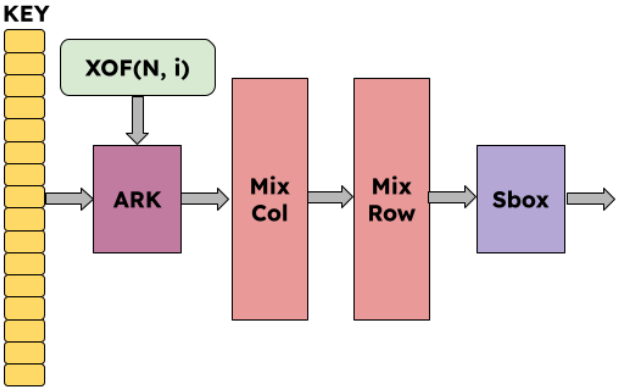
Hybrid Homomorphic Encryption: HERA



$$\begin{bmatrix} y_{c0} \\ y_{c1} \\ y_{c2} \\ y_{c3} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_{c0} \\ x_{c1} \\ x_{c2} \\ x_{c3} \end{bmatrix}$$

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix}$$

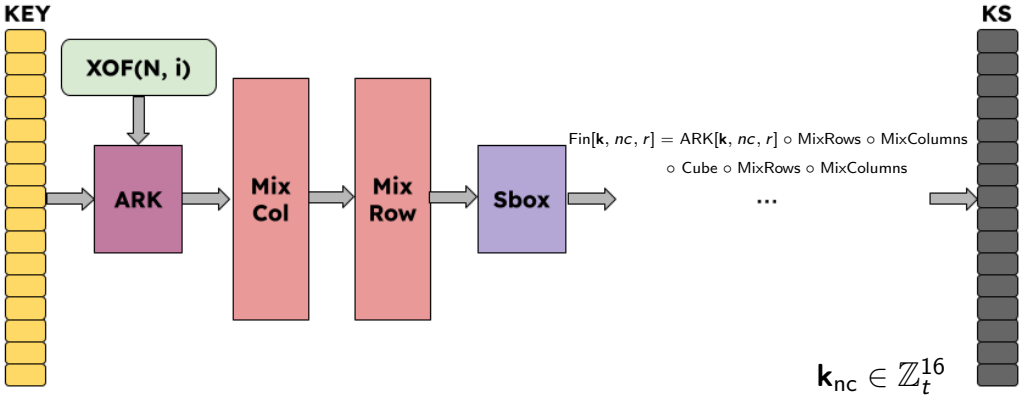
Hybrid Homomorphic Encryption: HERA



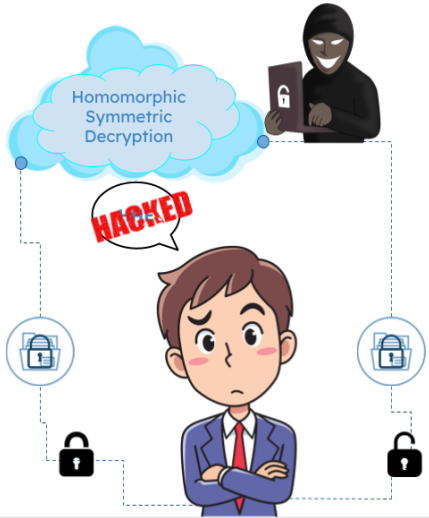
$$\text{Cube}(\mathbf{x}) = (x_0^3, \ldots, x_{15}^3)$$

$$\mathbf{x} = (x_0, \ldots, x_{15}) \in \mathbb{Z}_t^{16}$$

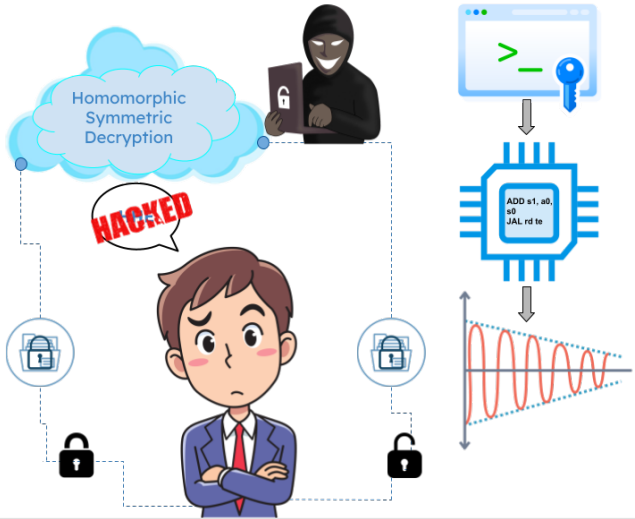
Hybrid Homomorphic Encryption: HERA



Are Implementations of Hybrid Homomorphic Schemes Secure?

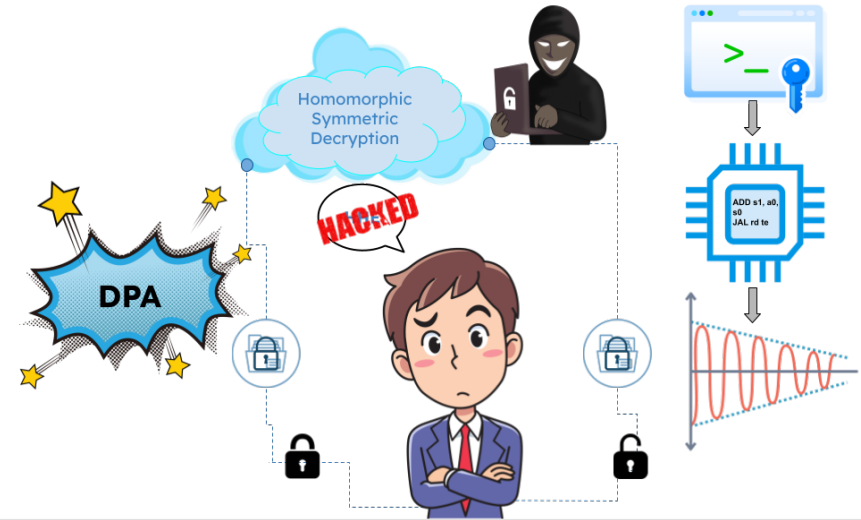


Are Implementations of Hybrid Homomorphic Schemes Secure?



Emira Salkić, ISEC

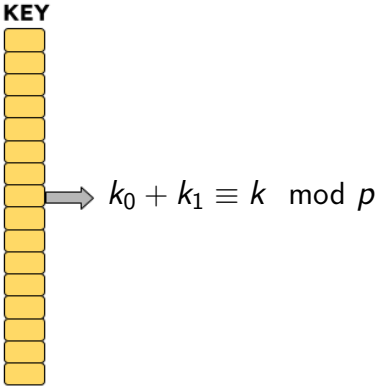
Are Implementations of Hybrid Homomorphic Schemes Secure?



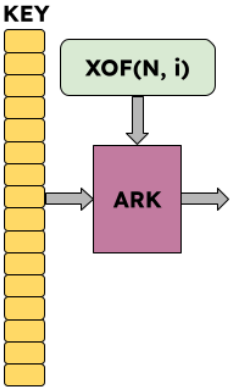
Emira Salkić, ISEC

First-order arithmetic masking

First-order arithemtic masking of HERA



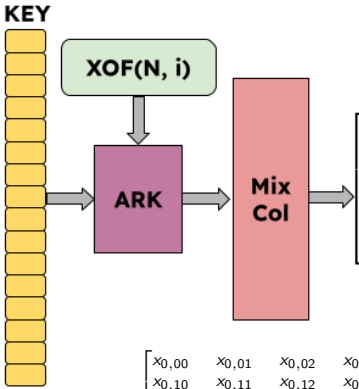
Hybrid Homomorphic Encryption: HERA



$\mathbf{rc} = (\mathbf{rc}_0, \ldots, \mathbf{rc}_r) \in (\mathbb{Z}_t^{16})^{r+1}$

$$\mathrm{ARK}[\mathbf{k}, nc, i] = \begin{cases} x_0 + (k_0 \bullet rc_i) \\ x_1 + (k_1 \bullet rc_i) \end{cases}$$

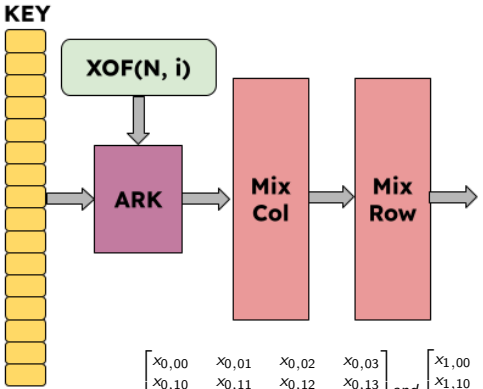
Hybrid Homomorphic Encryption: HERA



$$\begin{bmatrix} y_{m,0c} \\ y_{m,1c} \\ y_{m,2c} \\ y_{m,3c} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_{m,0c} \\ x_{m,1c} \\ x_{m,2c} \\ x_{m,3c} \end{bmatrix}, \; m \in \{1, 0\}$$

$$\begin{bmatrix} x_{0,00} & x_{0,01} & x_{0,02} & x_{0,03} \\ x_{0,10} & x_{0,11} & x_{0,12} & x_{0,13} \\ x_{0,20} & x_{0,21} & x_{0,22} & x_{0,23} \\ x_{0,30} & x_{0,31} & x_{0,32} & x_{0,33} \end{bmatrix} \text{ and } \begin{bmatrix} x_{1,00} & x_{1,01} & x_{1,02} & x_{1,03} \\ x_{1,10} & x_{1,11} & x_{1,12} & x_{1,13} \\ x_{1,20} & x_{1,21} & x_{1,22} & x_{1,23} \\ x_{1,30} & x_{1,31} & x_{1,32} & x_{1,33} \end{bmatrix}$$

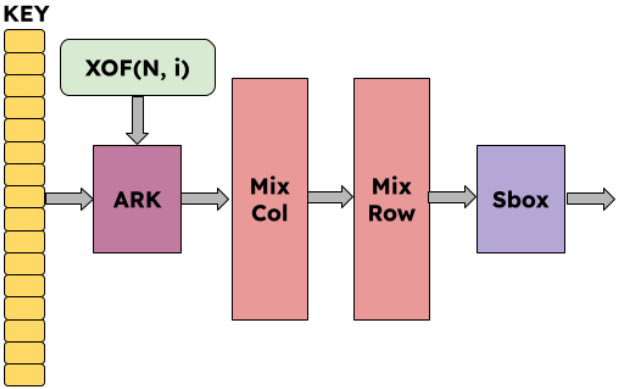
Hybrid Homomorphic Encryption: HERA



$$\begin{bmatrix} y_{m,c0} \\ y_{m,c1} \\ y_{m,c2} \\ y_{m,c3} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_{m,c0} \\ x_{m,c1} \\ x_{m,c2} \\ x_{m,c3} \end{bmatrix}$$

$$\begin{bmatrix} x_{0,00} & x_{0,01} & x_{0,02} & x_{0,03} \\ x_{0,10} & x_{0,11} & x_{0,12} & x_{0,13} \\ x_{0,20} & x_{0,21} & x_{0,22} & x_{0,23} \\ x_{0,30} & x_{0,31} & x_{0,32} & x_{0,33} \end{bmatrix} \textit{ and } \begin{bmatrix} x_{1,00} & x_{1,01} & x_{1,02} & x_{1,03} \\ x_{1,10} & x_{1,11} & x_{1,12} & x_{1,13} \\ x_{1,20} & x_{1,21} & x_{1,22} & x_{1,23} \\ x_{1,30} & x_{1,31} & x_{1,32} & x_{1,33} \end{bmatrix}$$

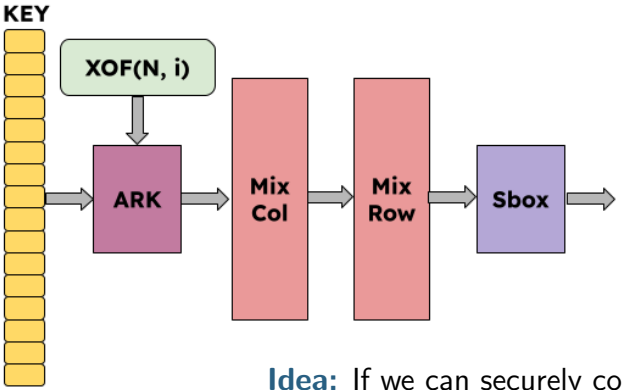
Hybrid Homomorphic Encryption: HERA



$$\text{Cube}(\mathbf{x}) = x^3$$

$$x_0 + x_1 \equiv x \mod p$$

Hybrid Homomorphic Encryption: HERA

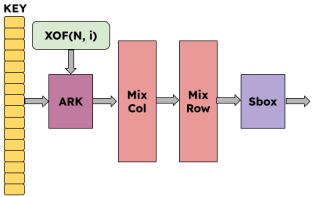


$$\text{Cube}(\mathbf{x}) = x^3$$

$$x_0 + x_1 \equiv x \mod p$$

Idea: If we can securely compute x^2,
then extending this approach to x^3 is straightforward.

Hybrid Homomorphic Encryption: HERA



$$\text{Cube}(\mathbf{x}) = x^3$$

$$x_0 + x_1 \equiv x \mod p$$

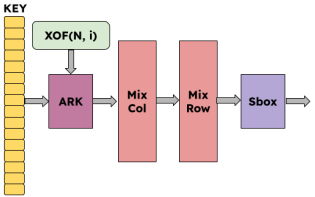
$$\text{Cube}(x) = \begin{cases} r = \text{rand}() \mod p \\ x_0' = (x_0 + r) \mod p \\ x_1' = (x_1 + p - r) \mod p \end{cases}$$

Note:

$$x_0' + x_1' \equiv x \mod p$$

Emira Salkić, ISEC

Hybrid Homomorphic Encryption: HERA



$$\text{Cube}(\mathbf{x}) = x^3$$

$$x_0 + x_1 \equiv x \mod p$$

$$\text{Cube}(x) = \begin{cases} r = \text{rand}() \mod p \\ x_0' = (x_0 + r) \mod p \\ x_1' = (x_1 + p - r) \mod p \\ \mathbf{f}(x_0, x_1, x_0', x_1') \\ \mathbf{f}(x_0'', x_1'', x_0', x_1') \end{cases}$$

Note:

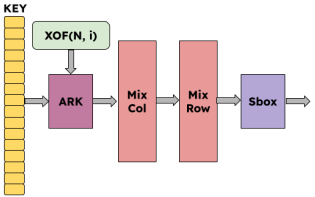
$x_0' + x_1' \equiv x \mod p$

Note:

f(...) executes secure multiplication

Emira Salkić, ISEC

Hybrid Homomorphic Encryption: HERA



$$\text{Cube}(\mathbf{x}) = x^3$$

$$x_0 + x_1 \equiv x \mod p$$

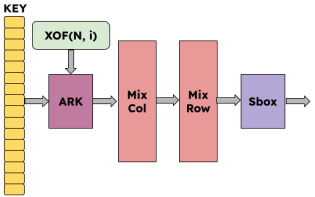
Note:

First call to f(...): $(x_0 + x_1)^2 \equiv x^2 \mod p$

Second call to f(...): $(x_0 + x_1)^3 \equiv x^3 \mod p$

$$\text{Cube}(x) = \begin{cases} r = \text{rand}() \mod p \\ x_0' = (x_0 + r) \mod p \\ x_1' = (x_1 + p - r) \mod p \\ \mathbf{f}(x_0, x_1, x_0', x_1') \qquad \Rightarrow (x_0'' + x_1'')^2 \equiv x^2 \mod p \\ \mathbf{f}(x_0'', x_1'', x_0', x_1') \end{cases}$$

Hybrid Homomorphic Encryption: HERA



$$\text{Cube}(\mathbf{x}) = x^3$$
$$x_0 + x_1 \equiv x \mod p$$

$$\text{Cube}(x) = \begin{cases} r = \text{rand}() \mod p \\ x_0' = (x_0 + r) \mod p \\ x_1' = (x_1 + p - r) \mod p \\ \mathbf{f}(x_0, x_1, x_0', x_1') \\ \mathbf{f}(x_0'', x_1'', x_0', x_1') \end{cases} \quad \mathbf{f}(\ldots) = \begin{cases} c_0 = \text{rand}() \mod p \\ c_1 = (c_0 + (x_0 \cdot x_1') + (x_1 \cdot x_0')) \mod \\ x_0'' = ((x_0 \cdot x_0') + p - c_0) \mod p \\ x_1'' = ((x_1 \cdot x_1') + c_1) \mod p \end{cases}$$

Results

Results

Metric	Not Masked	Masked
Encryption Time (μs)	84	123
Decryption Time (μs)	66	106
CPUs Utilized	0.617	0.785
Task Clock (ms)	3.24	3.45
Instructions Executed	6,044,834	6,114,087

Table: Comparison of Masked vs. Non-Masked Implementations

Conclusion

Conclusion and Future Scope

- First-order masking protects against first-order attacks.
- Implementation is easily extendable to higher-order masking to protect against higher-order attacks.
- Taking hybrid homomorphic encryption one step closer to real-world secure cloud computing.

TU Graz

SCIENCE
PASSION
TECHNOLOGY

Masking Hybrid Homomorphic Encryption Schemes

Emira Salkić

13.02.2025

Extending HERA to Higher-Order Security

Key Idea: Increase the number of shares to improve security.

Current Two-Share Representation

$$x_0 + x_1 \equiv x \mod p$$

Generalized N-Share Representation

$$x_0 + x_1 + x_2 + \ldots + x_{N-1} \equiv x \mod p$$

Note:

- Each share is randomized but maintains modular sum.
- Security improves, but computational cost increases.

Formal proof: AddRoundKey

Algorithm 4 HERA AddRoundKey Function (Masked)

Require: Masked state arrays S_0, S_1, masked key arrays K_0, K_1, length $n = HERA\_K$.
Ensure: Updated masked state arrays S_0, S_1.

1: **for** $i = 0$ **to** $n - 1$ **do**
2: $r_i \leftarrow \text{UniformRandom}(\mathbb{Z}_p)$
3: $m_{0,i} \leftarrow (K_{0,i} \cdot r_i) \mod p$
4: $m_{1,i} \leftarrow (K_{1,i} \cdot r_i) \mod p$
5: $S'_{0,i} \leftarrow (S_{0,i} + m_{0,i}) \mod p$
6: $S'_{1,i} \leftarrow (S_{1,i} + m_{1,i}) \mod p$
7: **end for**

Formal proof: MixColumns

Algorithm 7 HERA MixColumns Function (Masked)

Require: Masked state shares $S_0 = \{c_{i,j,0}\}$, $S_1 = \{c_{i,j,1}\}$

Ensure: Updated masked state shares $S'_0 = \{c'_{i,j,0}\}$, $S'_1 = \{c'_{i,j,1}\}$

1: **for** $i = 0$ **to** 3 **do**

2: $\quad sum_0 \leftarrow (c_{i,0,0} + c_{i,1,0} + c_{i,2,0} + c_{i,3,0}) \mod p.$

3: $\quad sum_1 \leftarrow (c_{i,0,1} + c_{i,1,1} + c_{i,2,1} + c_{i,3,1}) \mod p.$

4: $\quad$ **for** $j = 0$ **to** 3 **do**

5: $\quad\quad c'_{i,j,0} \leftarrow (sum_0 + c_{i,j,0} + c_{i,(j+1 \mod 4),0} + c_{i,(j+2 \mod 4),0}) \mod p.$

6: $\quad\quad c'_{i,j,1} \leftarrow (sum_1 + c_{i,j,1} + c_{i,(j+1 \mod 4),1} + c_{i,(j+2 \mod 4),1}) \mod p.$

7: $\quad$ **end for**

8: **end for**

Emira Salkić, ISEC

Formal proof: MixRows

Algorithm 8 HERA MixRows Function (Masked)

Require: Masked state shares $S_0 = \{r_{i,j,0}\}$, $S_1 = \{r_{i,j,1}\}$
Ensure: Updated masked state shares $S'_0 = \{r'_{i,j,0}\}$, $S'_1 = \{r'_{i,j,1}\}$

for $j = 0$ **to** 3 **do**
  $sum_0 \leftarrow (r_{0,j,0} + r_{1,j,0} + r_{2,j,0} + r_{3,j,0}) \mod p.$
  $sum_1 \leftarrow (r_{0,j,1} + r_{1,j,1} + r_{2,j,1} + r_{3,j,1}) \mod p.$
  for $i = 0$ **to** 3 **do**
    $r'_{i,j,0} \leftarrow (sum_0 + r_{i,j,0} + r_{(i+1 \mod 4),j,0} + r_{(i+2 \mod 4),j,0}) \mod p.$
    $r'_{i,j,1} \leftarrow (sum_1 + r_{i,j,1} + r_{(i+1 \mod 4),j,1} + r_{(i+2 \mod 4),j,1}) \mod p.$
  end for
end for

Formal proof: S-box

Algorithm 6 HERA S-Box Cube Function (Masked)

Require: Masked state shares $s_0 = \{s_{0,0}, s_{0,1}, \ldots, s_{0,n}\}$, $s_1 = \{s_{1,0}, s_{1,1}, \ldots, s_{1,n}\}$

Ensure: Updated masked state shares s'_0, s'_1, satisfying $s' = (s^3) \mod p$.

1: **for** $i = 0$ **to** $n - 1$ **do**
2: Generate random value $t_1 \in \mathbb{Z}_p$.
3: $t_2 \leftarrow (s_{0,i} + t_1) \mod p$.
4: $t_3 \leftarrow (s_{1,i} + p - t_1) \mod p$.
5: SECURE_MUL$(s_{0,i}, s_{1,i}, t_2, t_3)$, resulting in intermediate masked shares $(z_{0,i}, z_{1,i})$
6: SECURE_MUL$(z_{0,i}, z_{1,i}, t_2, t_3)$, resulting in updated masked shares $(s'_{0,i}, s'_{1,i})$
7: $s_{0,i} \leftarrow s'_{0,i}$
8: $s_{1,i} \leftarrow s'_{1,i}$
9: **end for**

Formal proof: Masked Gadget

Algorithm 5 HERA Secure Multiplication (Masked Gadget)

Require: Masked inputs (x_0, x_1) and (y_0, y_1)

Ensure: Masked outputs (z_0, z_1), where $z = (x \cdot y) \mod p$

1: Generate random value $r_{0,1} \in \mathbb{Z}_p$

2: $r_{1,0} \leftarrow \big((x_0 \cdot y_1) + (x_1 \cdot y_0)\big) \mod p$

3: $z_0 \leftarrow \big((x_0 \cdot y_0) + p - r_{0,1}\big) \mod p$

4: $z_1 \leftarrow \big((x_1 \cdot y_1) + r_{1,0}\big) \mod p$