# Making neural cryptography great again

## With SN P Systems

Mihail-Iulian Plesa

January 2023

# Table of contents

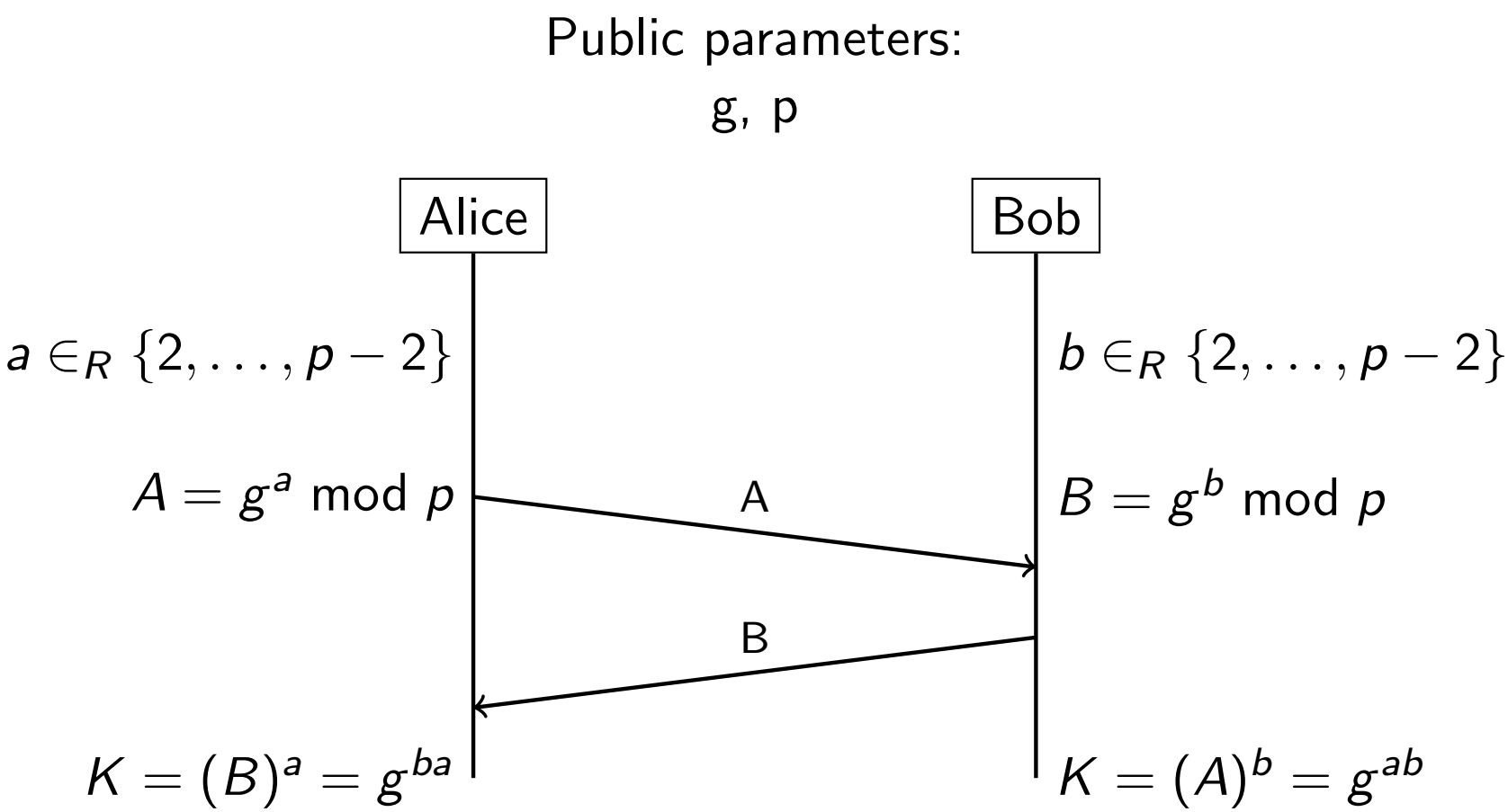
# Key Agreement Protocols

- Classical
- Post-quantum
- Quantum

# Classical

- Based on hard problems from number theory (DH, DL)
- No mathematical proof
- Hope it will work

# Diffie-Hellman (DH)



# Security of DH

## Conjecture

*Given $g$, $p$, $g^a$ and $g^b$ there is no known efficient algorithm for computing $g^{ab}$*

## Conjecture

*Given $g$, $p$ and $g^x$ there is no known efficient algorithm for computing $x$*

# Why do we need anything else?

## Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

**Abstract**

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

# Why do we need anything else?

**Factoring integers with sublinear resources on a superconducting quantum processor**

Bao Yan,[1,2,*] Ziqi Tan,[3,*] Shijie Wei,[4,*] Haocong Jiang,[5] Weilong Wang,[1] Hong Wang,[1] Lan Luo,[1] Qianheng Duan,[1] Yiting Liu,[1] Wenhao Shi,[1] Yangyang Fei,[1] Xiangdong Meng,[1] Yu Han,[1] Zheng Shan,[1] Jiachen Chen,[3] Xuhao Zhu,[3] Chuanyu Zhang,[3] Feitong Jin,[3] Hekang Li,[3] Chao Song,[3] Zhen Wang,[3,†] Zhi Ma,[1,‡] H. Wang,[3] and Gui-Lu Long[2,4,6,7,§]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*
[2]*State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing 100084, China*
[3]*School of Physics, ZJU-Hangzhou Global Scientific and Technological Innovation Center, Interdisciplinary Center for Quantum Information, and Zhejiang Province Key Laboratory of Quantum Technology and Device, Zhejiang University, Hangzhou 310000, China*
[4]*Beijing Academy of Quantum Information Sciences, Beijing 100193, China*
[5]*Institute of Information Technology, Information Engineering University, Zhengzhou 450001, China*
[6]*Beijing National Research Center for Information Science and Technology and School of Information Tsinghua University, Beijing 100084, China*
[7]*Frontier Science Center for Quantum Information, Beijing 100084, China*

**Shor's algorithm has seriously challenged information security based on public key cryptosystems. However, to break the widely used RSA-2048 scheme, one needs millions of physical qubits, which is far beyond current technical capabilities. Here, we report a universal quantum algorithm for integer factorization by combining the classical lattice reduction with a quantum approximate optimization algorithm (QAOA). The number of qubits required is $O(\log N/\log\log N)$, which is sublinear in the bit length of the integer $N$, making it the most qubit-saving factorization algorithm to date. We demonstrate the algorithm experimentally by factoring integers up to 48 bits with 10 superconducting qubits, the largest integer factored on a quantum device. We estimate that a quantum circuit with 372 physical qubits and a depth of thousands is necessary to challenge RSA-2048 using our algorithm. Our study shows great promise in expediting the application of current noisy quantum computers, and paves the way to factor large integers of realistic cryptographic significance.**

# Post-quantum

- Based on hard problems from number theory (lattice problems)
- No mathematical proof
- Hope it will work

Different problems - same philosophy

# Quantum

- Based on quantum effects (collapse of the probability wave, entanglement, etc)
- No need for a mathematical proof
- It works

# BB84

| Alice's bit | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|
| Alice's basis | + | + | X | + | X | X | X | + |
| Alice's polarization | ↑ | → | ↖ | ↑ | ↖ | ↗ | ↗ | → |
| Bob's basis | + | X | X | X | + | X | + | + |
| Bob's measurement | ↑ | ↗ | ↖ | ↗ | → | ↗ | → | → |
| Public discussion | | | | | | | | |
| Shared Secret key | 0 | | 1 | | | 0 | | 1 |

[1] https://www.cse.wustl.edu/ jain/cse571-07/ftp/quantum/

# Neural Key Exchange

## Secure exchange of information by synchronization of neural networks

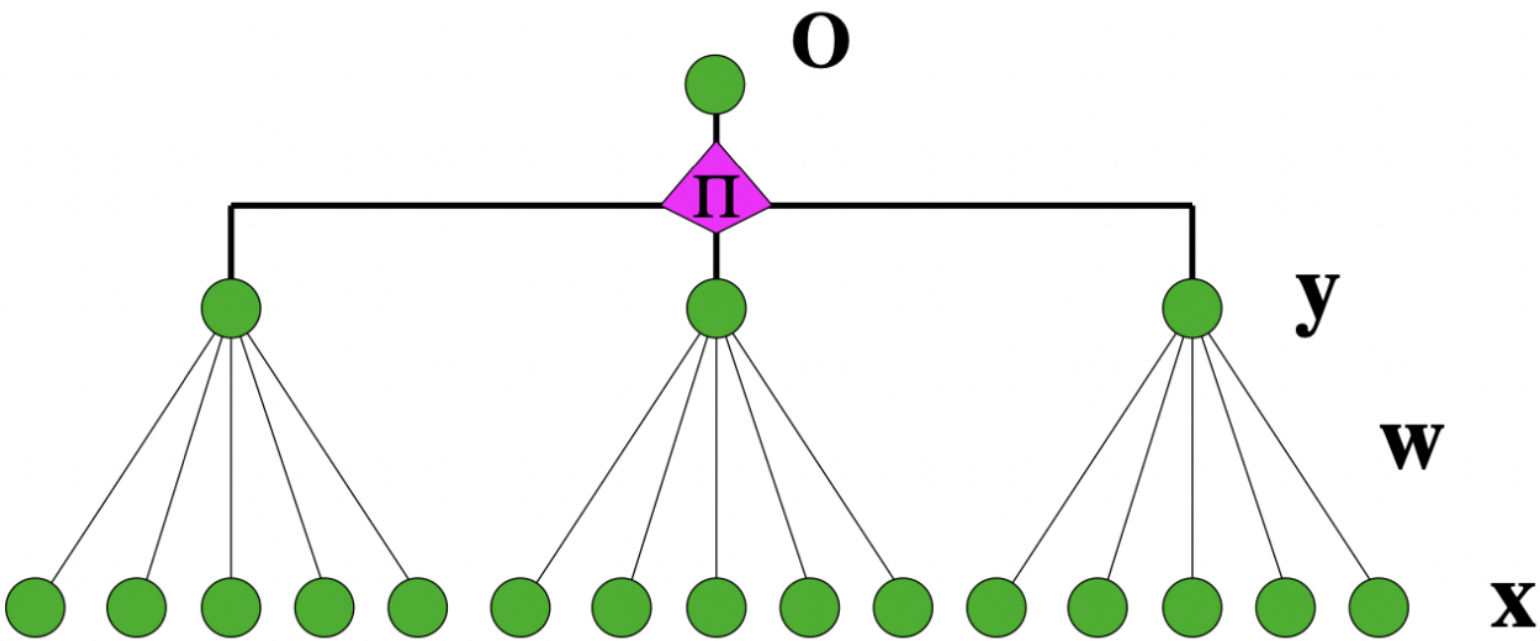
Ido Kanter[1], Wolfgang Kinzel[2] and Eran Kanter[1]

(1) Department of Physics, Bar Ilan University, 52900 Ramat Gan, Israel
(2) Institut für Theoretische Physik, Universität Würzburg, Am Hubland, D-97074 Würzburg, Germany
3. 9. 2001

**Abstract**

A connection between the theory of neural networks and cryptography is presented. A new phenomenon, namely synchronization of neural networks is leading to a new method of exchange of secret messages. Numerical simulations show that two artificial networks being trained by Hebbian learning rule on their mutual outputs develop an antiparallel state of their synaptic weights. The synchronized weights are used to construct an ephemeral key exchange protocol for a secure transmission of secret data. It is shown that an opponent who knows the protocol and all details of any transmission of the data has no chance to decrypt the secret message, since tracking the weights is a hard problem compared to synchronization. The complexity of the

# Tree Parity Machine (TPM)



# Tree Parity Machine

**Definition**

Two TPMs are synchronized if their sequences of weights are identical

# Neural Key Agreement - In Theory

- Based on the synchronization of two TPMs
- No number theory assumptions
- Quantum secure

# Neural Key Agreement - In Practice

- Geometric attack [1]
- Genetic attack [2]
- Majority attack [3]

An attacker can recover more than 90% of the key just by listening to the legitimate participants of the protocol

[1] Klimov, A., Mityagin, A. and Shamir, A., 2002, December. Analysis of neural cryptography. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 288-298). Springer, Berlin, Heidelberg.

[2] Ruttor, A., Kinzel, W., Naeh, R. and Kanter, I., 2006. Genetic attack on neural cryptography. Physical Review E, 73(3), p.036121

[3] Shacham, L.N., Klein, E., Mislovaty, R., Kanter, I. and Kinzel, W., 2004. Cooperating attackers in neural cryptography. Physical Review E, 69(6), p.066137.

# Further improvements

- Input with feedback [1]
- Output with errors [2]

Trade efficiency for security

[1] Ruttor, A., Kinzel, W., Shacham, L. and Kanter, I., 2004. Neural cryptography with feedback. Physical Review E, 69(4), p.046110.

[2] Allam, A.M. and Abbas, H.M., 2009, June. Improved security of neural cryptography using don't-trust-my-partner and error prediction. In 2009 International Joint Conference on Neural Networks (pp. 121-127). IEEE

# Anti Spiking Neural Tree Parity Machine P System (ASNTPM P System)

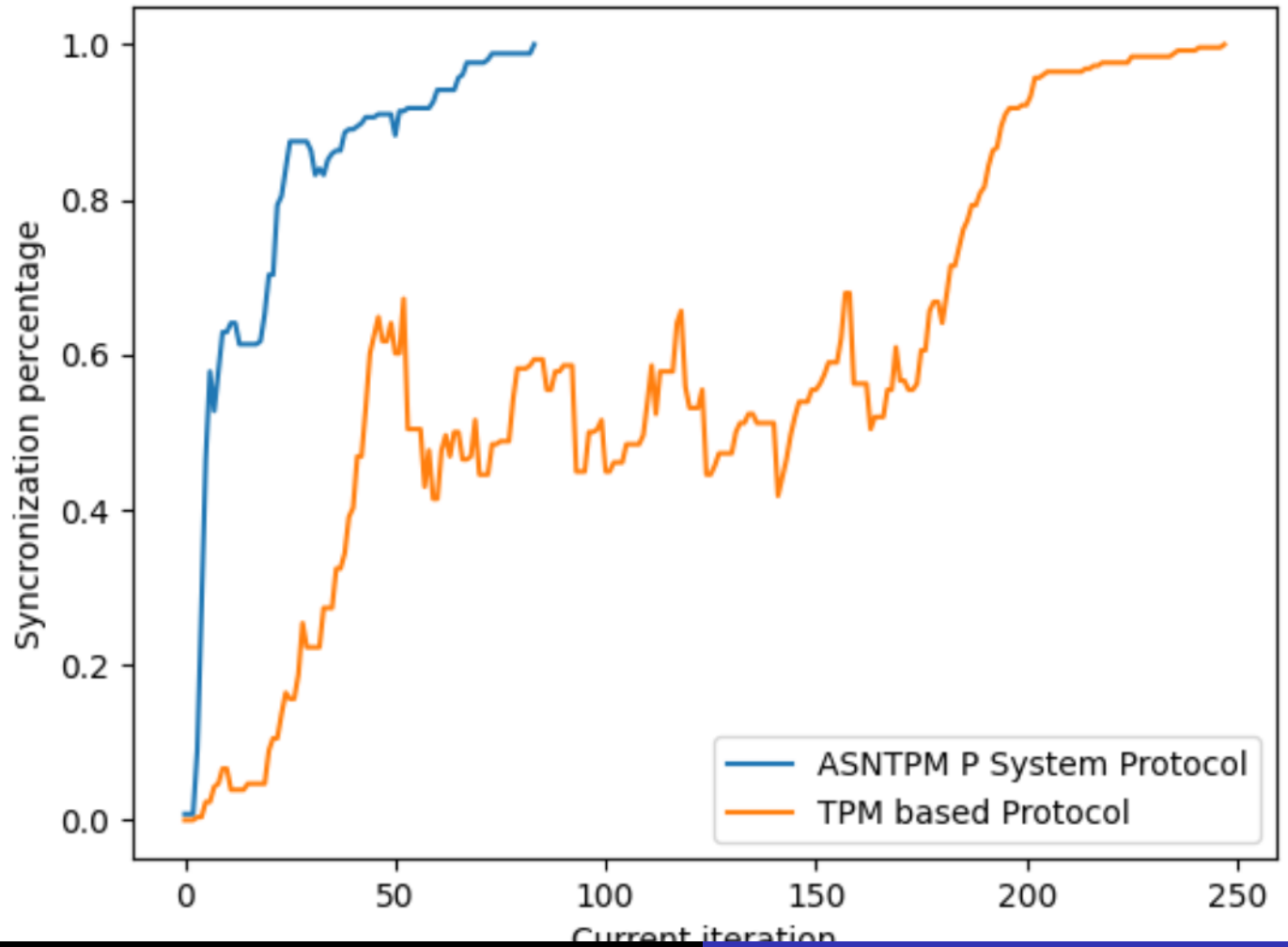
- SN P neuron model
- $K$ - the number of neurons from the hidden layer
- $N$ - the number of neurons from the input layer connected to a single hidden neuron
- $L$ - the maximum value of a weight

[1]Plesa, M.I., Gheoghe, M., Ipate, F. and Zhang, G., 2022. A key agreement protocol based on spiking neural P systems with anti-spikes. Journal of Membrane Computing, pp.1-11.

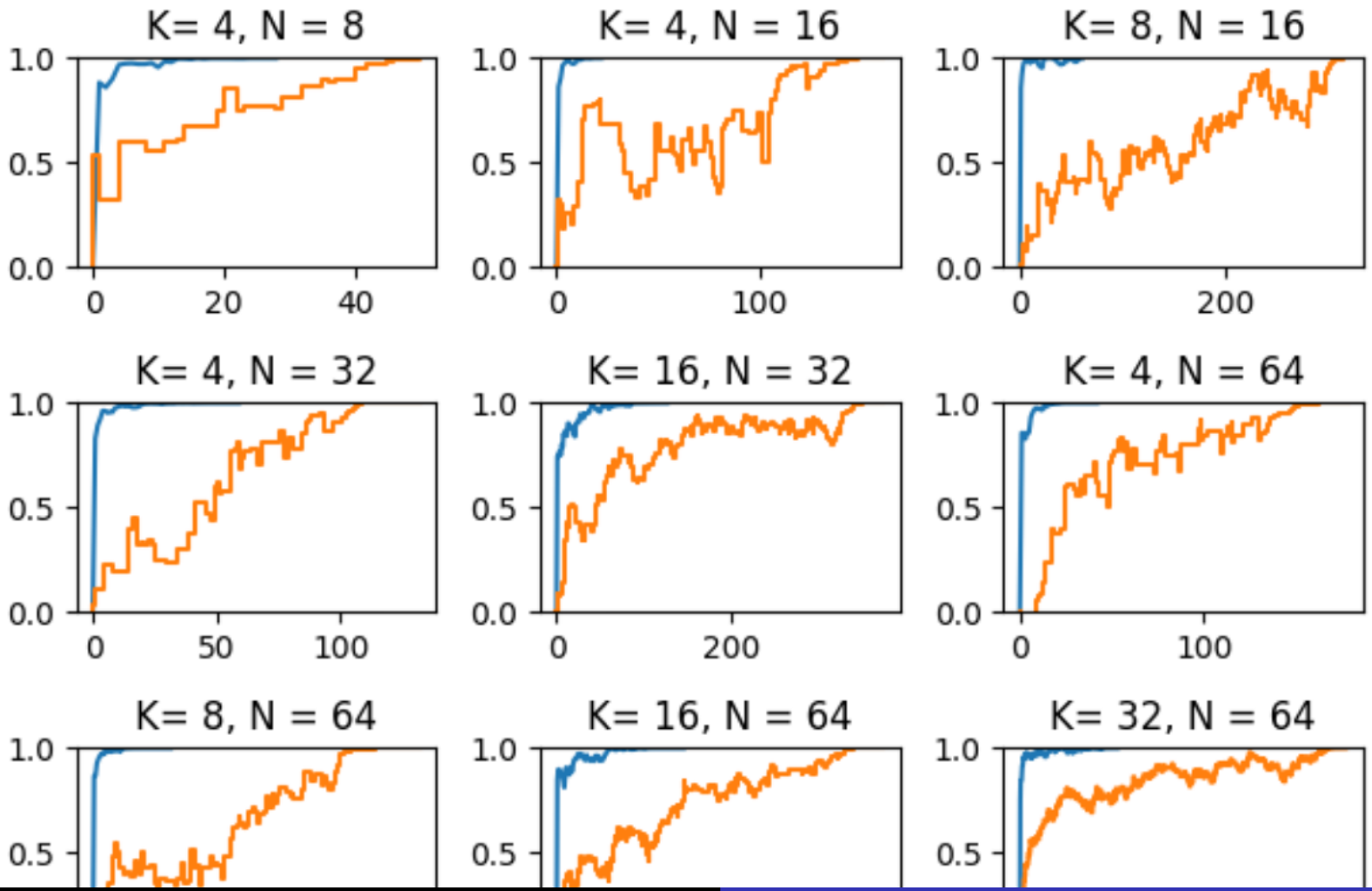
# ASNTPM - Efficiency

| K | N | Syncronization time | | | |
|---|---|---|---|---|---|
| | | TPM | | ASNTPM P system | |
| | | Mean | Standard deviation | Mean | Standard deviation |
| 4 | 8 | 138.49 | 64.23 | 31.62 | 12.26 |
| 4 | 16 | 135.16 | 63.81 | 39.02 | 16.39 |
| 8 | 16 | 247.18 | 80.78 | 65.45 | 23.72 |
| 4 | 32 | 138.77 | 52.06 | 49.99 | 17.44 |
| 8 | 32 | 278.16 | 89.87 | 77.30 | 25.37 |
| 16 | 32 | 450.63 | 90.65 | 112.78 | 32.68 |
| 4 | 64 | 149.59 | 53.45 | 60.15 | 23.03 |
| 8 | 64 | 319.74 | 102.59 | 86.39 | 29.78 |
| 16 | 64 | 519.39 | 122.25 | 128.64 | 36.48 |
| 32 | 64 | 727.02 | 130.50 | 188.32 | 54.15 |

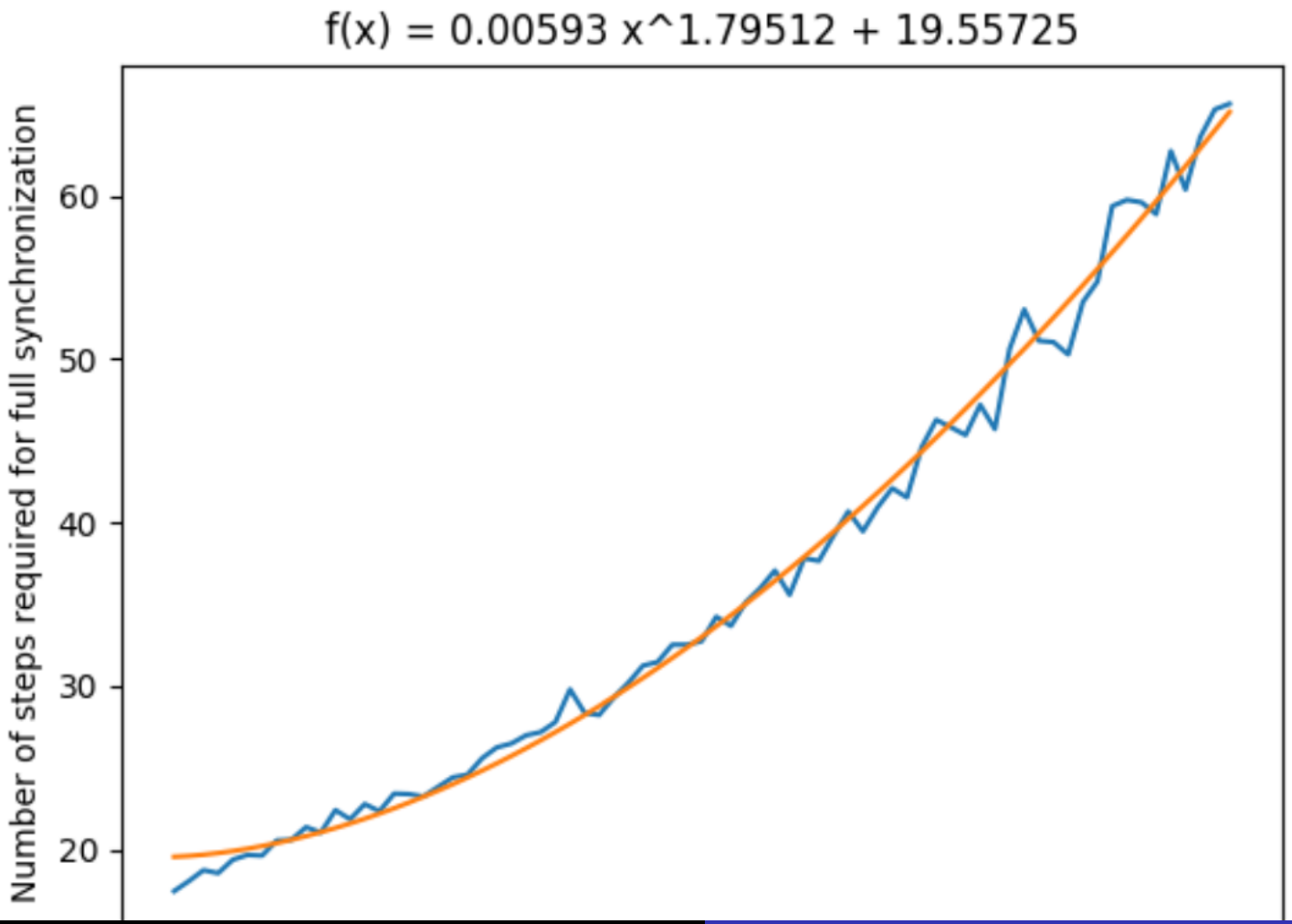
# ASNTPM - Efficiency



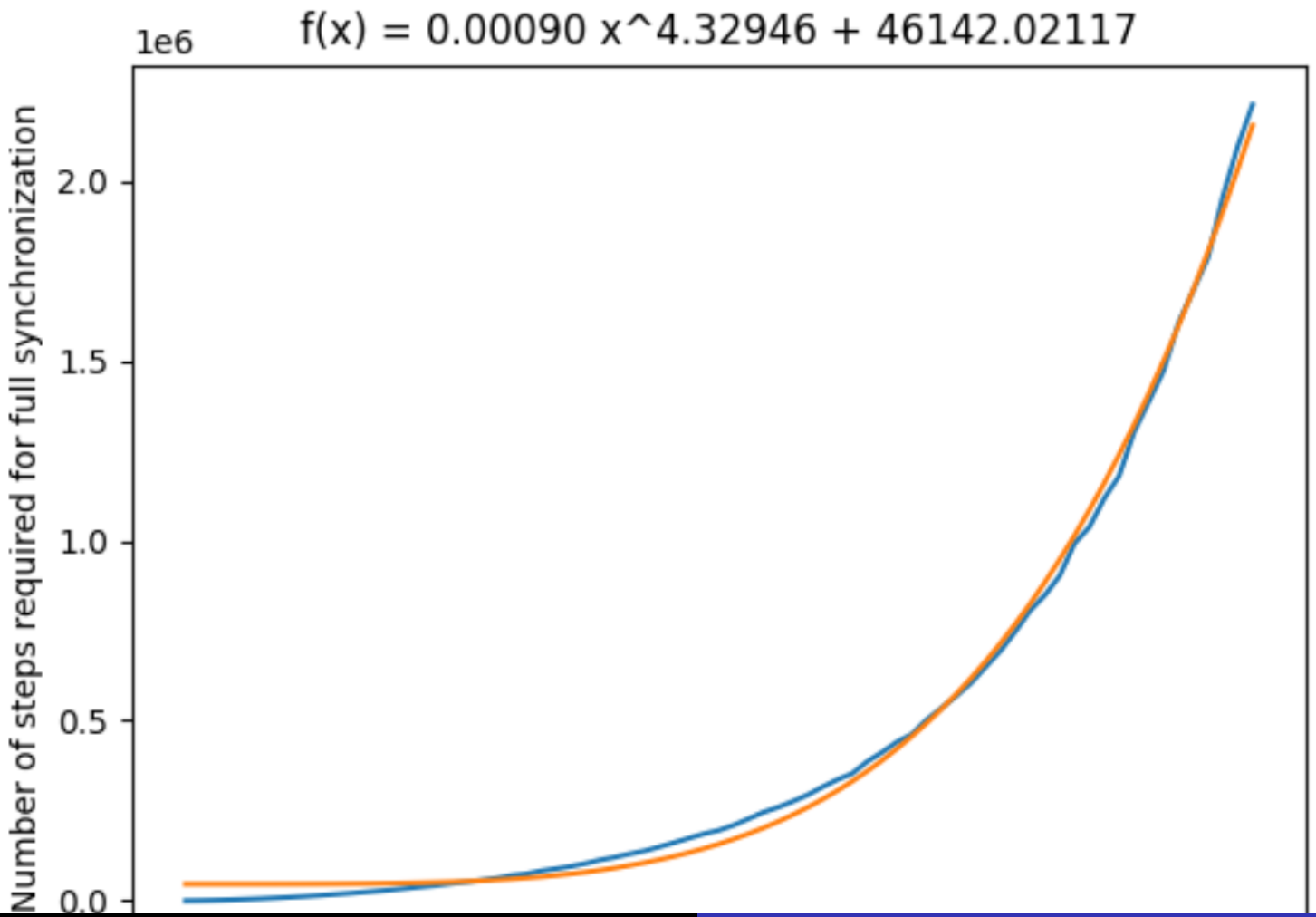
# ASNTPM - Efficiency



# ASNTPM - Efficiency



# ASNTPM - Efficiency



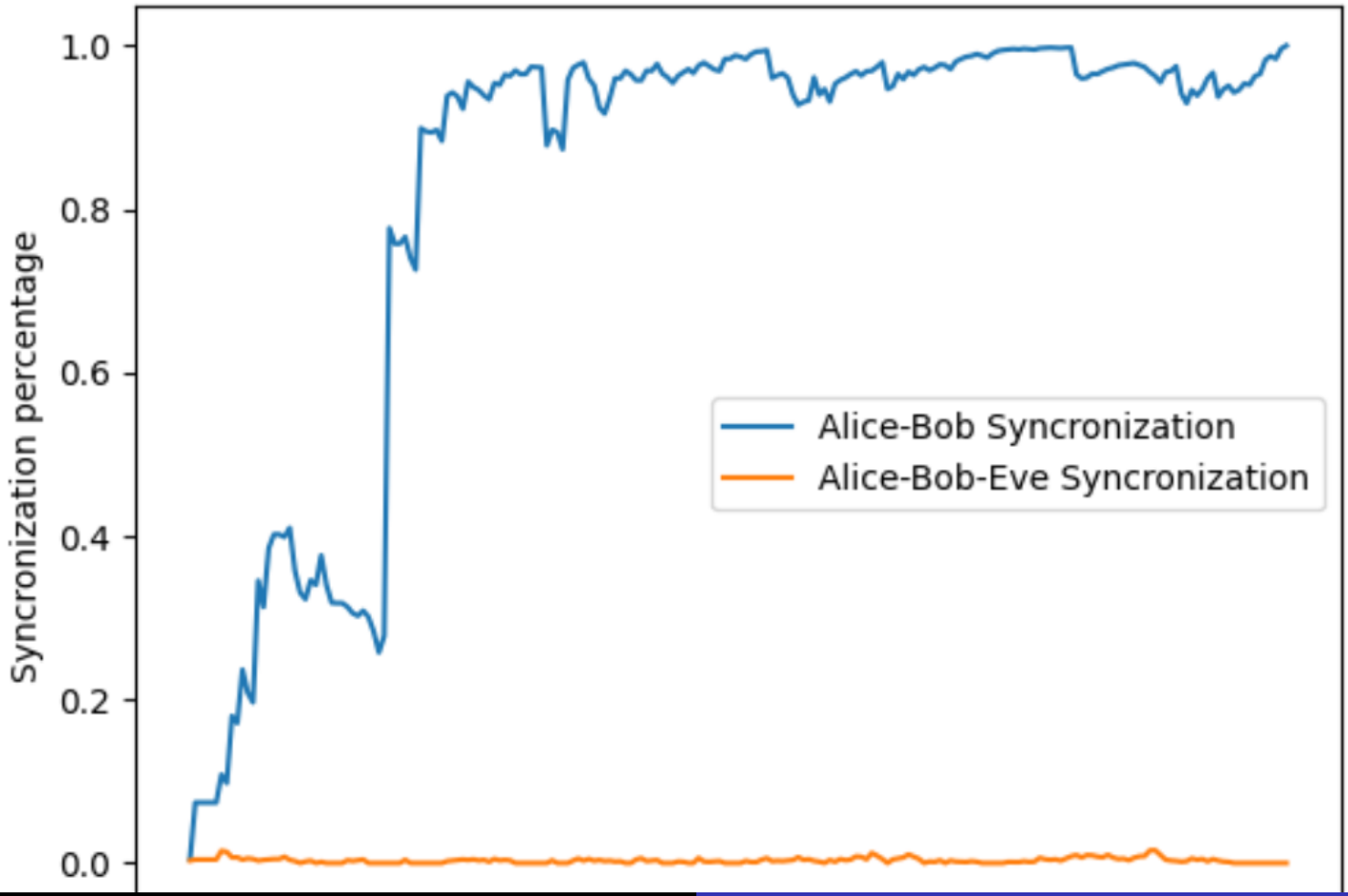
# ASNTPM - Security

| K | N | Mean entropy of the key | |
|---|---|---|---|
| | | ASNTPM P system | TPM |
| | | Mean | Mean |
| 4 | 8 | 4.86 | 4.82 |
| 4 | 16 | 5.87 | 5.79 |
| 8 | 16 | 6.84 | 6.79 |
| 4 | 32 | 6.86 | 6.78 |
| 8 | 32 | 7.85 | 7.77 |
| 16 | 32 | 8.82 | 8.77 |
| 4 | 64 | 7.87 | 7.76 |
| 8 | 64 | 8.85 | 8.76 |
| 16 | 64 | 9.83 | 9.76 |
| 32 | 64 | 10.83 | 10.76 |

# ASNTPM - Security

| K | N | Mean synchronization percentage of the attacker | |
|---|---|---|---|
| | | ASNTPM P system | TPM |
| | | Mean | Mean |
| 4 | 8 | 0.05 | 0.68 |
| 4 | 16 | 0.02 | 0.40 |
| 8 | 16 | 0.01 | 0.45 |
| 4 | 32 | 0.01 | 0.53 |
| 8 | 32 | 0.02 | 0.40 |
| 16 | 32 | 0.02 | 0.30 |
| 4 | 64 | 0.02 | 0.34 |
| 8 | 64 | 0.02 | 0.30 |
| 16 | 64 | 0.03 | 0.34 |
| 32 | 64 | 0.03 | 0.32 |

# ASNTPM - Security



# Cryptography works with proofs

- Formal adversary model
- Formal security proof in that model

# We need a "hard problem"

- Classical key agreement protocols: DH, DDH, Factoring, DL, etc
- Post-quantum key agreement protocols: SVP, CVP, LWE, RLWE, etc
- Quantum cryptography: entanglement, no cloning, the collapse of the probability wave, etc.

Prove that breaking the protocol is equivalent to solving a "hard problem"

# Neural synchronization as a hard problem

- **No hard problem is defined in neural cryptography**
- **No formal adversary model**
- **No formal security proof**

# ASNTPM - constructing a hard problem

**Problem**

Can we design an efficient algorithm for synchronizing two ASNTPMs without mutual learning?

# ASNTPM - constructing a hard problem

## Our strategy

Treat every attack on TPMs as a possible algorithm for synchronizing two ASNTPMs without mutual learning.

# ASNTPM - constructing a hard problem

**Algorithm 1** ASNTPM P System initialization

**function** INITIALIZE($\Pi, X$)
  **for** $i = 1;\ i \leq K;\ i = i + 1$ **do**
    **for** $j = 1\ j \leq N\ j = j + 1$ **do**
      **if** $X[i * N + j] \leq 0$ **then**
        $\overline{N}(\Pi, \sigma_{in_{ij}}) = |X[i * N + j]|$
      **else**
        $N(\Pi, \sigma_{in_{ij}}) = X[i * N + j]$
      **end if**
      $W_\Pi[i][j] \xleftarrow{\$} [0, 2L]$
    **end for**
  **end for**
  **for** $i = 1;\ i \leq K;\ i = i + 1$ **do**
    $\overline{N}(\Pi, \sigma_{h_i}) = 0$
    $N(\Pi, \sigma_{h_i}) = 0$
  **end for**
  $\overline{N}(\Pi, \sigma_{out}) = 0$
  $N(\Pi, \sigma_{out}) = 0$
**end function**

# ASNTPM - constructing a hard problem

**Algorithm 2** ASNTPM P System running

1: **function** RUN($\Pi$)
2: **for** $i = 1;\ i \leq K;\ i = i + 1$ **do**
3: **for** $j = 1\ j \leq N\ j = j + 1$ **do**
4: $\overline{N}(\Pi, \sigma_{h_i}) = \overline{N}(\Pi, \sigma_{h_i}) + W_\Pi[i][j] * \overline{N}(\Pi, \sigma_{in_{ij}})$
5: $N(\Pi, \sigma_{h_i}) = N(\Pi, \sigma_{h_i}) + W_\Pi[i][j] * N(\Pi, \sigma_{in_{ij}})$
6: **end for**
7: **end for**
8: **for** $i = 1;\ i \leq K;\ i = i + 1$ **do**
9: $\overline{N}(\Pi, \sigma_{out}) = \overline{N}(\Pi, \sigma_{out}) + \overline{N}(\Pi, \sigma_{h_i})$
10: $N(\Pi, \sigma_{out}) = N(\Pi, \sigma_{out}) + N(\Pi, \sigma_{h_i})$
11: **end for**
12: **end function**

# ASNTPM - constructing a hard problem

**Algorithm 3** The learning function

**function** UPDATEWEIGHTS($\Pi$)
  **for** $i = 1;\ i \leq K;\ i = i + 1$ **do**
    **if** $[[N(\Pi, \sigma_{h_i}) = N(\Pi, \sigma_{out})] \vee [\overline{N}(\Pi, \sigma_{h_i}) = \overline{N}(\Pi, \sigma_{out})]]$ **then**
      **for** $j = 1\ j \leq N\ j = j + 1$ **do**
        **if** $N(\Pi, \sigma_{out}) > 0$ **then**
          $W_\Pi[i][j] = \left|W_\Pi[i][j] + N(\Pi, \sigma_{in_{ij}})\right|$
        **else if** $\overline{N}(\Pi, \sigma_{out}) > 0$ **then**
          $W_\Pi[i][j] = \left|W_\Pi[i][j] - \overline{N}(\Pi, \sigma_{in_{ij}})\right|$
        **end if**
      **end for**
      **if** $W_\Pi[i][j] > L$ **then**
        $W_\Pi[i][j] = L$
      **end if**
    **end if**
  **end for**
**end function**

# ASNTPM - constructing a hard problem

**Algorithm 4** The synchronization percentage

```
1: function SynchronizationPercentage(Π1, Π2)
2:     total ← 0
3:     counter ← 0
4:     for i = 1 i ≤ K i = i + 1 do
5:         for j = 1 j ≤ N j = j + 1 do
6:             if W_Π1[i][j] ≠ 2L ∧ W_Π2[i][j] ≠ 2L then
7:                 total ← total + 1
8:                 if W_Π1[i][j] = W_Π2[i][j] then
9:                     counter ← counter + 1
10:                end if
11:            end if
12:        end for
13:    end for
14:    return counter/total
15: end function
```

# ASNTPM - constructing a hard problem

**Algorithm 5** Synchronization of two ASNTPM P Systems

**function** SyncA($\Pi_1, \Pi_2$)
  **while** SynchronizationPercentage($\Pi_1, \Pi_2$) $\neq 1$ **do**
    $x \xleftarrow{R} \mathbb{Z}^{KN}$
    Initialize($\Pi_1, x$)
    Initialize($\Pi_2, x$)
    Run($\Pi_1$)
    Run($\Pi_2$)
    **if** $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \overline{N}(\Pi_1, \sigma_{out}) = \overline{N}(\Pi_2, \sigma_{out})$ **then**
      UpdateWeights($\Pi_1$)
      UpdateWeights($\Pi_2$)
    **end if**
  **end while**
**end function**

# ASNTPM - constructing a hard problem

**Algorithm 6** Simple synchronization process of three ASNTPM

**function** SyncB($\Pi_1, \Pi_2, \Pi_3$)
  **while** SynchronizationPercentage($\Pi_1, \Pi_2$) $\neq 1$ **do**
    $x \overset{R}{\leftarrow} \mathbb{Z}^{KN}$
    Initialize($\Pi_1, x$)
    Initialize($\Pi_2, x$)
    Initialize($\Pi_3, x$)
    Run($\Pi_1$)
    Run($\Pi_2$)
    Run($\Pi_3$)
    **if** $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \overline{N}(\Pi_1, \sigma_{out}) = \overline{N}(\Pi_2, \sigma_{out})$ **then**
      UpdateWeights($\Pi_1$)
      UpdateWeights($\Pi_2$)
      **if** $N(\Pi_1, \sigma_{out}) = N(\Pi_3, \sigma_{out}) \vee \overline{N}(\Pi_1, \sigma_{out}) = \overline{N}(\Pi_3, \sigma_{out})$ **then**
        UpdateWeights($\Pi_3$)
      **end if**
    **end if**
  **end while**
  **return** SynchronizationPercentage($\Pi_1, \Pi_3$)
**end function**

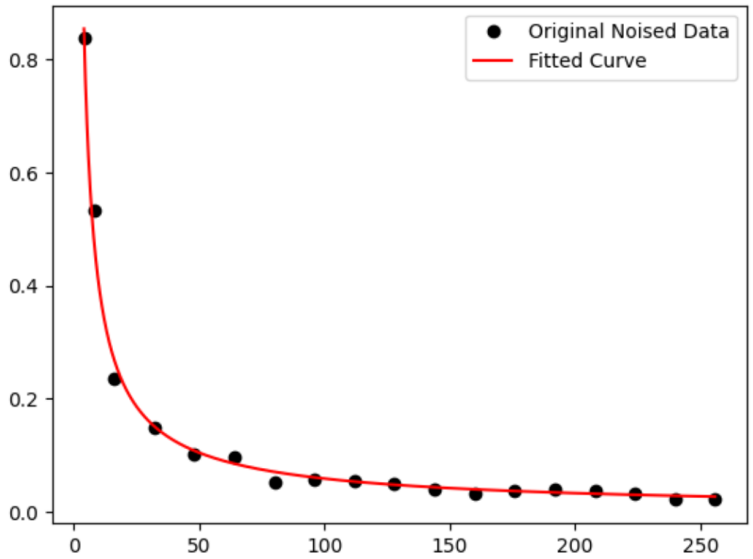
# ASNTPM - constructing a hard problem

**Algorithm 7** The geometric solution for the synchronization of three ASNTPM P Systems

**function** SyncC($\Pi_1, \Pi_2, \Pi_3$)
  **while** SynchronizationPercentage($\Pi_1, \Pi_2$) $\neq 1$ **do**
    $x \xleftarrow{R} \mathbb{Z}^{KN}$
    Initialize($\Pi_1, x$)
    Initialize($\Pi_2, x$)
    Initialize($\Pi_3, x$)
    Run($\Pi_1$)
    Run($\Pi_2$)
    Run($\Pi_3$)
    **if** $N(\Pi_1, \sigma_{out}) = N(\Pi_2, \sigma_{out}) \vee \overline{N}(\Pi_1, \sigma_{out}) = \overline{N}(\Pi_2, \sigma_{out})$ **then**
      UpdateWeights($\Pi_1$)
      UpdateWeights($\Pi_2$)
      **if** $N(\Pi_1, \sigma_{out}) = N(\Pi_3, \sigma_{out}) \vee \overline{N}(\Pi_1, \sigma_{out}) = \overline{N}(\Pi_3, \sigma_{out})$ **then**
        UpdateWeights($\Pi_3$)
      **end if**
      **if** $N(\Pi_1, \sigma_{out}) \neq N(\Pi_3, \sigma_{out}) \wedge \overline{N}(\Pi_1, \sigma_{out}) \neq \overline{N}(\Pi_3, \sigma_{out})$ **then**
        $distance = \|W_{\Pi_3}[1] - x\|$
        $minimum = distance$
        $index = 1$
        **for** $i = 2;\ i \leq K;\ i = i + 1$ **do**
          $distance = \|W_{\Pi_3}[i] - x\|$
          **if** $distance < minimum$ **then**
            $mininum = distance$
            $index = i$
          **end if**
        **end for**
        $aux = \overline{N}(\Pi_3, \sigma_{h_{index}})$
        $\overline{N}(\Pi_3, \sigma_{h_{index}}) = N(\Pi_3, \sigma_{h_{index}})$
        $N(\Pi_3, \sigma_{h_{index}}) = aux$
        $\overline{N}(\Pi_3, \sigma_{output}) = \overline{N}(\Pi_1, \sigma_{output})$
        $N(\Pi_3, \sigma_{output}) = N(\Pi_1, \sigma_{output})$
        UpdateWeights($\Pi_3$)
      **end if**
    **end if**
  **end while**
  **return** SynchronizationPercentage($\Pi_1, \Pi_3$)
**end function**

# ASNTPM - constructing a hard problem

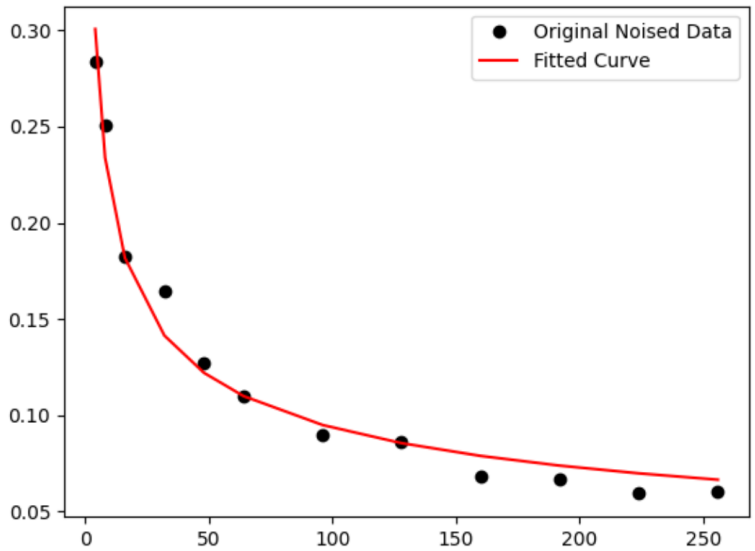
Figure: Sync percentage of the attacker as $K$ using the classical synchronization algorithm




$$\rho\left(\Pi_1, \Pi_3\right) = 2.7K^{-0.83}$$

# ASNTPM - constructing a hard problem

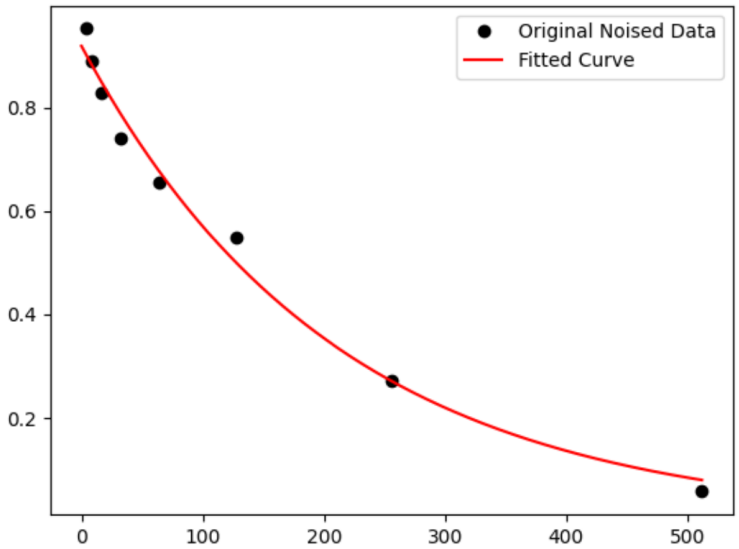
Figure: Sync percentage of the attacker as $N$ using the classical synchronization algorithm




$$\rho\left(\Pi_1, \Pi_3\right) = 0.49 N^{-0.36}$$

# ASNTPM - constructing a hard problem

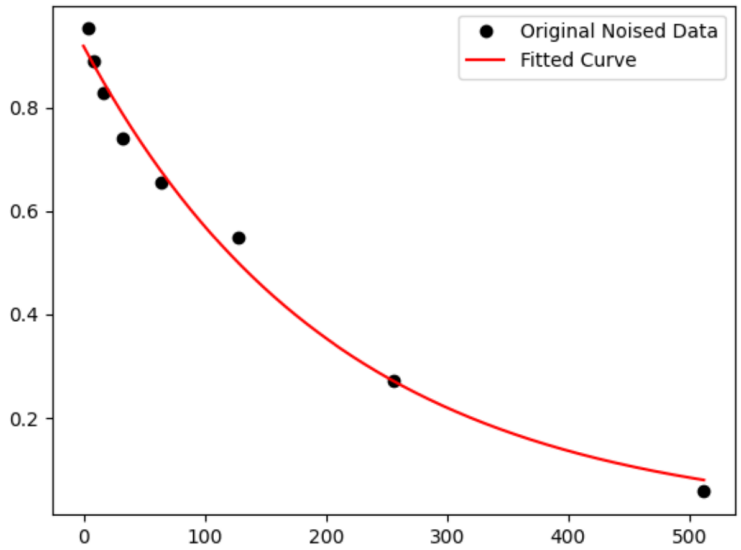
Figure: Sync percentage of the attacker as $K$ using the geometric synchronization algorithm




$$\rho\left(\Pi_1, \Pi_3\right) = 0.91e^{-0.004K}$$

# ASNTPM - constructing a hard problem

Figure: Sync percentage of the attacker as $N$ using the geometric synchronization algorithm




$$\rho\left(\Pi_1, \Pi_3\right) = 0.99e^{-0.003N}$$

# ASNTPM - Conclusions

- **Like quantum but cheaper**
- **Serious cryptography requires proofs, not intuition**
- **We cannot rely on the simple fact that "it works" (Shamir told us why)**
- **No previous work has attempted to do this**