HYPERLOCK: In-Memory Hyperdimensional Encryption in Memristor Crossbar Array

Author names, affiliations and correspondence information omitted for blind peer review

***Abstract*—We present a novel cryptography architecture based on memristor crossbar array, binary hypervectors, and neural network. Utilizing the stochastic and unclonable nature of memristor crossbar and error tolerance of binary hypervectors and neural network, implementation of the algorithm on memristor crossbar simulation is made possible. We demonstrate that with increasing dimension of the binary hypervectors, the non-idealities in memristor circuit can be effectively controlled. At the fine level of controlled crossbar non-ideality, noise from memristor circuit can be used to encrypt data while being sufficiently interpretable by neural network for decryption. We applied our algorithm on image cryptography for proof of concept, and to text en/decryption with 100% decryption accuracy despite of crossbar noises. Our work shows the potential and feasibility of using memristor crossbar as an unclonable stochastic encoder unit of cryptography on top of its existing functionality as a vector matrix multiplication acceleration device.**

***Index Terms*—Memristor Crossbar, Cryptography, Neural Network, Hyperdimensional Encryption**

I. Introduction

Memristors are non-volatile, configurable memory devices [1-2]. Their ability to permanently store variable conductance information makes them good candidates to build analog vector matrix multiplications (VMM) crossbar accelerators for in-memory computing. The crossbar performs VMM in O(1) and overcomes von Neumann bottleneck with in-memory computing [3], which enables many edge computing applications in machine learning such as CNN [6], LSTM [7] and neurmorphic computing [9]. However, despite their high efficiency, low footprint, and low power consumption [9-11], memristor crossbar suffer non-ideality issues such as sneak path current, device variability, stuck conductance, and cycle-to-cycle variability [12-15]. In some literature, non-ideality and behaviours of memristors are studied and their stochasiticity are exploit for hardware security applications [15] such as physical unclonable functions (PUFs) [16-18] and chaotic circuits [19-20]. Despite algorithms proposed in these studies covers from generating stochastic sequences for hardware verification to public and private key cryptography, none of which have touched on using memristor crossbar's VMM operation to encrypt the data directly.

In in-memory hyperdimensional encryption, we investigate the feasibility of using memristor crossbar's stochastic VMM operation for data encryption. Encrypting data directly with memristor crossbar poses a challenge for decryption because of the cycle to cycle variability can result in inconsistent cipher text for the same input. On the other hand, such randomness can be beneficial as it prevents repetitiveness of the encrypted text. In-memory hyperdimensional computing

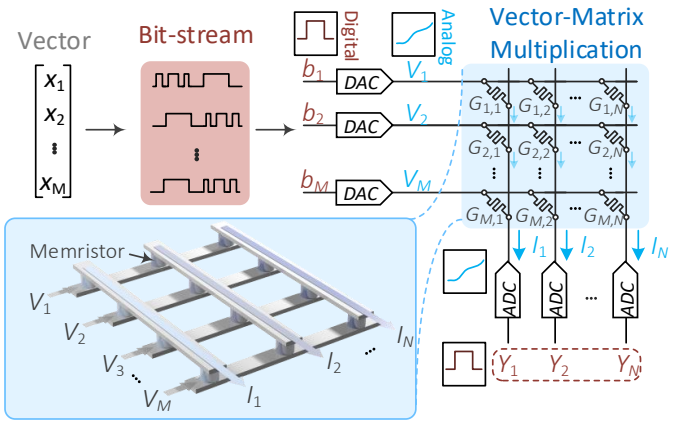

Fig. 1. Vector-matrix multiplication (VMM) over the memristor crossbar. 3D crossbar array has been displayed with input voltage and bit-line currents.

concept proposed by IBM [4] demonstrates the robustness of binary hypervectors against memrisor crossbar non-ideality. Such discovery leads to the inspiration of this work: to utilize binary hypervectors encryption to control the impact of noise generated by memristor crossbar, then train a shallow neural network for decryption. We demonstrate in simulation experiments: 1) the proof of concept on image cryptography, and 2) in text en/decryption to show that at a fine level of noise, the cipher text is near 100% decryptable by the neural network while being unique for each pass. Using memristor crossbar for this algorithm has a few benefits. First, the cost of VMM operation is low due to in-memory computing. Secondly, the algorithm can benefit from the intrinsic stochasticity of the memristor crossbar without the need of adding artificial noises. In the end, noises generated by crossbar provides additional security levels, and exact properties of the crossbar is unobtainable and unclonable by attackers. Power and time complexity of memristor crossbar are compared with CMOS digital implementation.

II. Preliminaries

A. Memristor Crossbar Arrays

A typical structure of a crossbar is shown in Fig. 1, where memristors are programmed and sandwiched between the word lines and the bit-lines. When analog voltage vectors are applied through the word-lines, memristors act as multiplication units according to Ohm's law (i.e. $I = VG$), and the current output from the memristors are then accumulated through the bit-lines. Despite its ideal functionalities, real implementation of the crossbar often comes with non-ideality [12-15] that causes errors in the output current vectors.

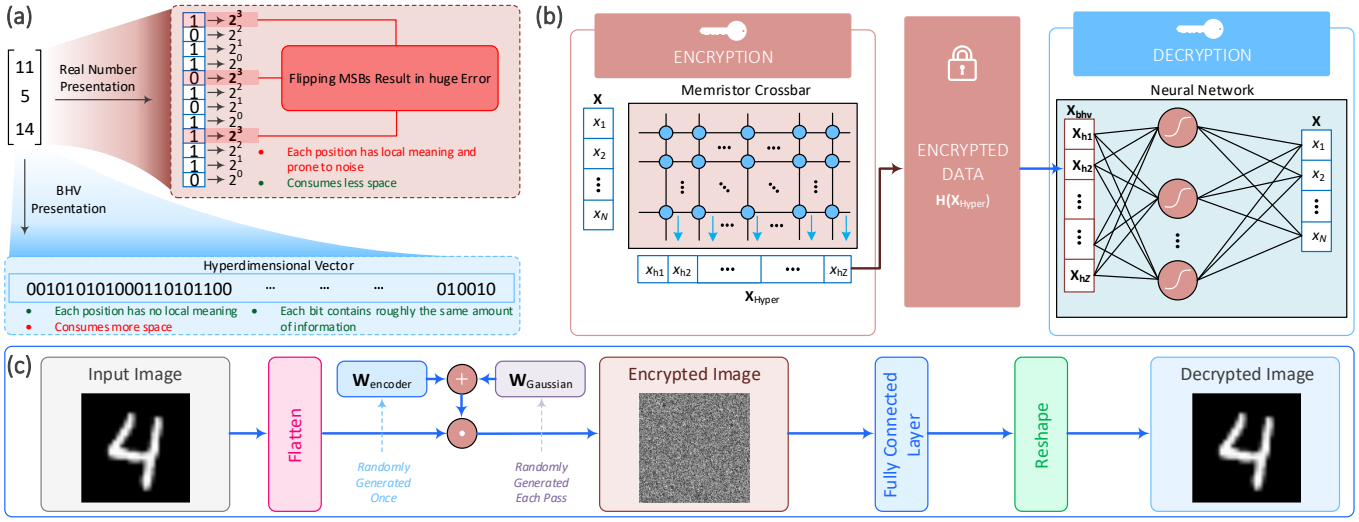

Fig. 2. (a) Binary hypervector representation versus real vector representation. (b)Proposed model architecture schematic for in-memory hyperdimensional encryption. (c) Image encryption and decryption implementation.

B. Binary Hypervectors

Binary hypervectors (BHV), first introduced by Kanerva [5] in his model of hyperdimensional (HD) computing, are binary vectors with dimensions in the orders of thousands. Unlike traditional real-valued vectors that are optimized for space, BHVs are optimized for robustness. In BHV, information is distributed evenly across all entries. Such representation provides resilience against noise, non-ideality, and low resolution in computer hardware as randomly flipping one bit of information has almost no impact on the vector's representation [22]. Since the information is evenly spread, BHV can be more robust when its dimension is increased. Real valued vectors, on the other hand, are prone to these noises, as failure at a critical bit can change the vector's representation significantly. Fig. 2(a) contrast the two vectors.

III. Proposed Model Architecture

Our proposed architecture (Fig. 2(b)) consists of a hyperdimensional stochastic encoder that encrypts the ordinary real value vectors into binary hypervectors, and a multi-layer perceptron (MLP) decoder that reconstructs the original vector.

A. Hyperdimensional Stochastic Encoder

The stochastic encoder is characterized by Equation (1), where $W_{encoder}$ is a tall, randomly initialized matrix that linearly transforms the original low dimensional input vector $\mathbf{x}$ into a *hypervector*, $f_{noise}(t, W_{encoder}, \mathbf{x})$ is some noise function that depends on time t, $W_{encoder}$, and $\mathbf{x}$, and $\mathbf{H}$ is a binarization function defined by Equation (2), where ϵ is a hyperparameter. The result, $\mathbf{x}_{bhv}$, is an encrypted binary hypervector.

$$\mathbf{x}_{bhv} = \mathbf{H}(W_{encoder}\mathbf{x} + f_{noise}(t, W_{encoder}, \mathbf{x})) \quad (1)$$

$$\mathbf{H}(x) = \begin{cases} 0, & x < \epsilon \\ 1, & x > \epsilon \end{cases} \quad (2)$$

The above formulation models the VMM of an intrinsic memristor crossbar array followed by a threshold. The randomly initialized matrix $W_{encoder}$ can be thought as an untuned memristor crossbar, and the noise function $f_{noise}(t, W_{encoder}, \mathbf{x})$ are the crossbar non-idealities which depend on the crossbar conductance (finite conductance states and conductance variability), input voltage vectors (e.g. sneak path current), and time (cycle to cycle variability).

The intuition behind a hyperdimensional stochastic encoder is that the VMM operation will create hypervectors, evenly distributes input vector's information across all entries. As a result, information at each entry can be represented by binary states. On the other hand, impact on performance from noise reduces as the dimension of the BHV gets larger. By altering the output dimension of the stochastic encoder, we control the level of noise presented in the encrypted BHV.

B. Neural Network Decoder

Encrypted BHV is fed into a fully connected neural network, dimension reduced by the weighted edges to reconstruct the original input vector. Weights of the neural network can be obtained through supervised learning.

IV. Image Encryption

We first apply our proposed model for image en/decryption (Fig. 2(c)) to verify the assumptions we had in our formulation. We create a naive simulation on PyTorch, where the stochastic encoder is implemented by Equation (3).

$$\mathbf{x}_{bhv} = \mathbf{H}((W_{encoder} + W_{Gaussian})\mathbf{x}) \quad (3)$$

$W_{encoder}$ is a fixed matrix randomly generated by PyTorch's uniform initialization function in the interval $(-2, 2)$ and $W_{Gaussian}$ is a Gaussian noise matrix generated by PyTorch's normal distribution function at each pass. The input vector $\mathbf{x}$ is the flattened image vector. After encoded by Equation (3), the encrypted BHV is fed into a single layer MLP to reconstruct

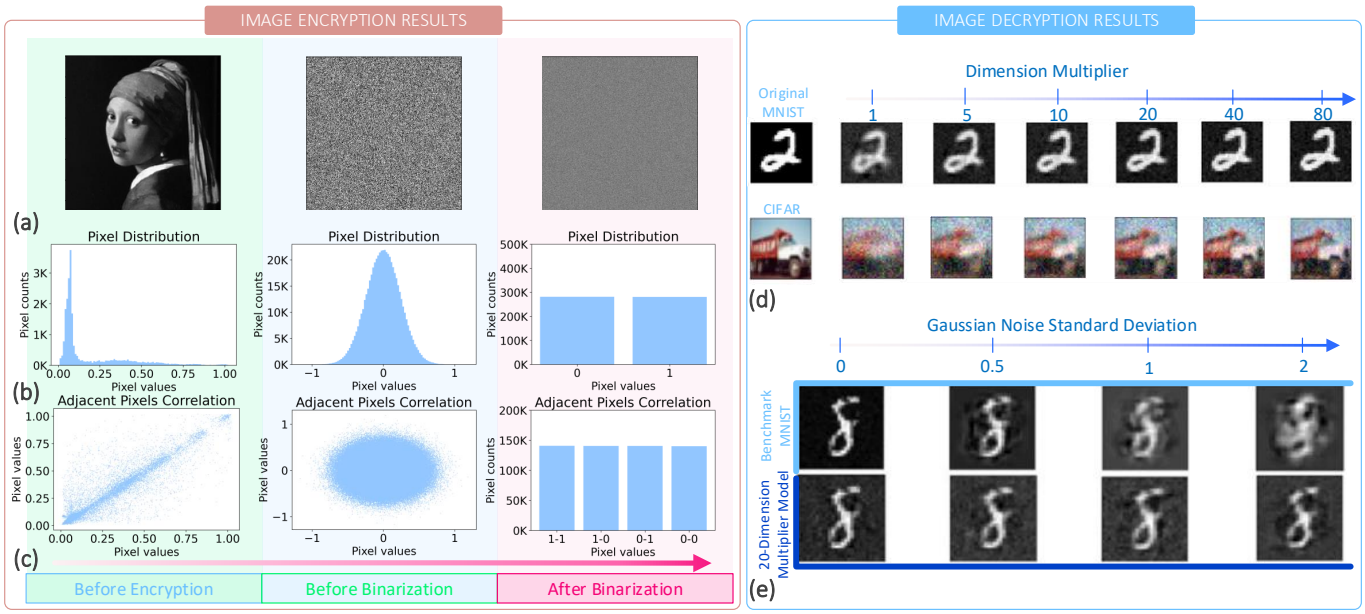

Fig. 3. Image encryption and decryption results. Image encryption result is on a 150×150 pixels *Girl with a Pearl Earring* image. (a) Raw image at three stages of encryption: before, after dimension expansion, and after threshold. (b) Pixel correlation at the three stages. (c) Adjacent pixel correlation at the three stages. Note that pixel correlation at the third stage is represented by a bar graph due to there are only 4 adjacent pixel combinations, namely: $0-0$, $0-1$, $1-0$, $1-1$. (d) Image decryption results: Reconstructed images at different level of *dimension multiplier* (the factor in which the input image vector dimension is expanded) in the stochastic encoding. (e) Reconstructed MNIST images at different standard deviations of Gaussian noise.

the original image. During training, we used root mean square error as cost function and stochastic gradient descent (SGD) to train the model until the validation loss stabilizes. We present en/decryption result in Fig. 3.

A. Dataset and Benchmark model

We trained the model on MNIST and CIFAR-10 image classification dataset respectively. MNIST consists of 60K 28×28 black and white images of handwritten digits from $0-9$, and CIFAR-10 consists of 60K 32×32 color images representing 10 classes of objects. Instead of performing classification, we trained the MLP to reconstruct these images.

We trained a benchmark model where $W_{encoder}$ in Equation (3) is a square matrix which does not expand the input vector $\mathbf{x}$ to hyperdimension, and we got rid of the threshold function. Ideally, the MLP decoder will learn the inverse of $W_{encoder}$ when noise is not presented, achieving little to no image reconstruction noise for decryption in perfect scenario.

B. Simulation Results

We show, in Fig. 3(a-c), the raw encryption result along with the change in pixel distribution and correlation of encrypting a 150×150 pixels *Girl with a Pearl Earring* image using our method. We show no obvious correlations between pixels. In Fig. 3(d), we demonstrate the positive correlation between dimension expansion and reconstructed image quality and, in Fig. 3(e), the robustness of BHV model against noise compared to the benchmark model. Reconstructed digit still remains legible even when the Gaussian noise standard deviation is equal to $W_{encoder}$'s initialization range (-2, 2).

TABLE I
SAMPLE CROSSBAR HYPERPARAMETERS IN EXPERIMENT

size	$\mathbf{R}_{LRS}(\mathrm{K}\Omega)$	$\mathbf{R}_{HRS}(\mathrm{K}\Omega)$	σ^1	$\mathbf{P}_{on}$	$\mathbf{P}_{off}$	**D.A.**[2]
5×250	1	100	0.1	0.01	0.01	99.55%
5×500	1	100	0.1	0.01	0.01	99.96%
10×500	1	10	0.1	0.02	0.02	100.0%
10×1000	1	10	0.4	0.05	0.05	99.97%
15×300	1	10	0.2	0.02	0.02	100.0%
15×600	1	10	0.7	0.02	0.02	98.17%

[1]Each conductance is perturbed by Gaussian noise with standard deviation equal to σ multiplied by the crossbar conductance range.
[2]Decryption accuracy over 10K characters.

V. TEXT ENCRYPTION

In this section, we generalize our proposed model to a text en/decryption model. The proposed model for this application is shown in Fig. 4(a).

A. Model Algorithm, Dataset and Simulation Setup

The model is described by Algorithm 1. New model can be generated *without* new crossbar simply by regenerating secret vectors and run Train(). The dataset used in simulations are characters of 94 classes, corresponding to ASCII 32 to 126. Our training, validation, and test sets consist of 100K, 100K, and 10K of randomly generated characters from the 94 classes, respectively. We tested the above algorithm on a memristor crossbar array simulation based on PyTorch. Conductance variability is modeled based on Gaussian distribution and random stuck on/off memristors. Samples of crossbar hyperparameters are presented in Table 1 along with the test set decryption accuracy.

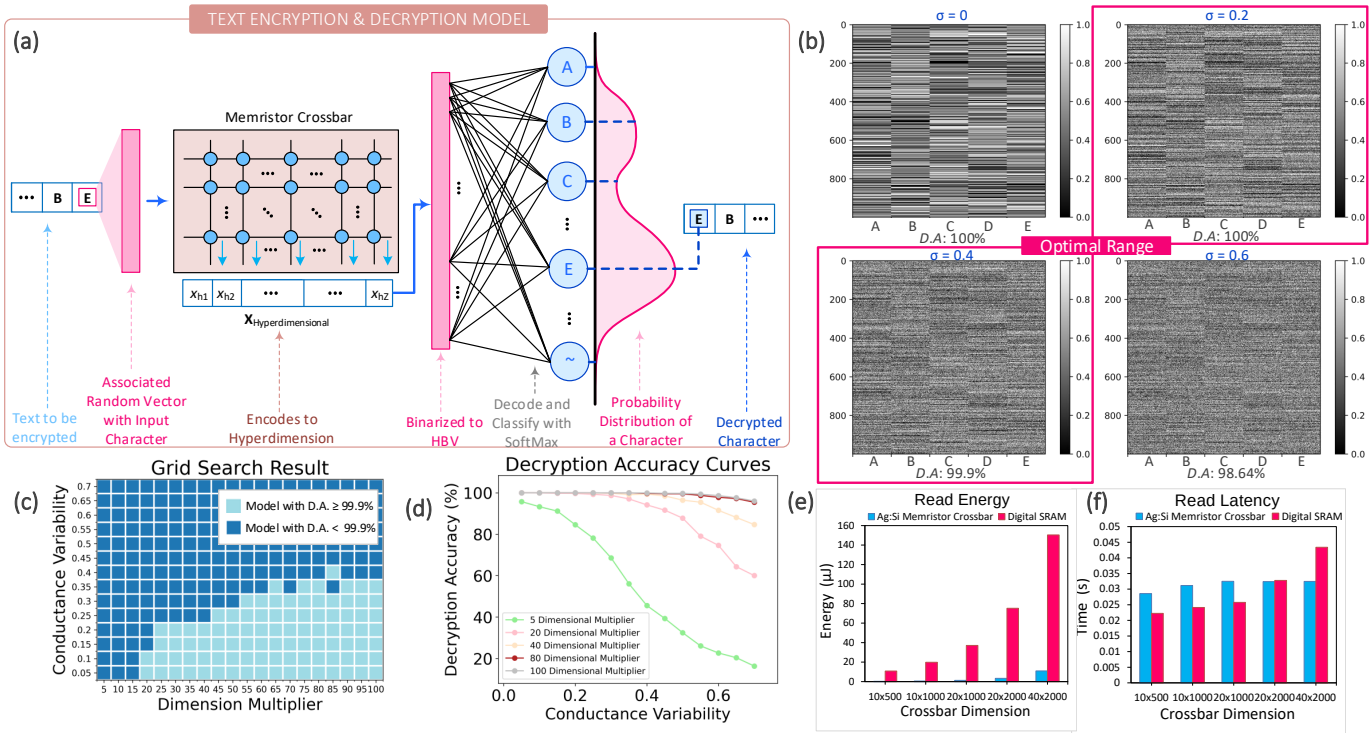

Fig. 4. Application for Text Decryption and Encryption results.(a) Text en/decryption model. (b) Encrypted text by different level of noise within the memristor crossbar. Each column is a BHV and each letter (A, B, C, D and E) is encoded 200 times. Optimal model is achieved when both noise and decryption accuracy are high.(c) Decryption performance of models with various dimension multipliers and crossbar noise level. Grid search on 10 dimensional secret key models with 0.05 $\mathbf{P}_{on}$ and 0.05 $\mathbf{P}_{off}$. "Good" model (shaded) are those with decryption accuracy $\geq$ 99.9% on test set. (d) Decryption accuracy vs conductance variability for selected models. Note that more noise means more security for the encoded BHV, while less dimension multiplier means less computation cost. (e) Power and complexity analysis of memristor crossbar implementation. (f) Time and complexity analysis.

Algorithm 1 Text En/Decryption

Input: *text*, *char_list*, *train_set*
Def Generate(*char_list*): ▷ Returns a randomly generated low-dimensional secret vector **for** each char in *char_list*.
Def Map(S, SV): ▷ Maps a String to its Secret Vector.
Def VMM(**A**,**x**): ▷ Performs **Ax** using the **crossbar**.
Def Train(): ▷ Returns a NN *Model* (linear mapping + SoftMax) for decryption, trained by negative log likelihood loss and SGD over examples in *train_set*.
procedure TEXT EN/DECRYPTION( )
  Model = Train() ▷ Give *Model* to receiver.
  secret_keys = Generate(*char_list*)
  for *char* in *text* **do**
    $\mathbf{x}_{hyper}$ = VMM(Map(*char*, *secret_keys*))
    $\mathbf{x}_{hbv}$ = $\mathbf{H}(\mathbf{x}_{hyper})$ ▷ This is encryption.
    ... send over to receiver ...
    c_decrypted = $Model(\mathbf{x}_{hbv})$ ▷ This is decryption.
  end for
end procedure

VI. Experimental Results

We show the encrypted letters at different levels of crossbar noise in Fig. 4(b), with their corresponding decryption accuracy. Non-ideality within the memristor crossbar array makes the encoded BHV different at each pass, thereby ensuring the security of the encrypted text. In Fig. 4(c), we show the decryption performance on a crossbar with, 0.05 $\mathbf{P}_{on}$ and 0.05 $\mathbf{P}_{off}$, at various levels of dimension multipliers and conductance variability. A model is considered "good" if the result decryption accuracy is at least 99.9% on the test set. In Fig. 4(d), we show the decryption accuracy curves of various models. We simulated energy and time complexity on analog Ag:Si memristor crossbar and digital SRAM [23] at different crossbar sizes for the stochastic encoder with NeuroSim MLP [24]. Fig 4(e-f) compares the result. The Ag:Si memristor consumes 30× less energy than digital SRAM implementation while having consistent read latency at increasing VMM scale. In addition, digital implementation does not have the security advantage from crossbar non-ideality.

VII. Conclusion

Overall, we demonstrate a novel cryptography algorithm designed specifically for memristor crossbar. In image encryption experiment, we verified our hypothesis using binary hypervectors to control crossbar noise levels. We then developed a stochastic text encryption system, and demonstrated near 100% decryption accuracy in the text decryption with selected crossbar models. This work is a proof of concept of how memristor crossbars with its non-ideal nature can be used to directly encrypt data, paving the foundations for future works in this direction.

REFERENCES

[1] L. Chua, "Memristor-The missing circuit element," in IEEE Transactions on Circuit Theory, vol. 18, no. 5, pp. 507-519, September 1971, doi: 10.1109/TCT.1971.1083337.

[2] Strukov, D., Snider, G., Stewart, D. et al. The missing memristor found. Nature 453, 80–83 (2008).

[3] Amirsoleimani, A., Alibart, F., Yon, V., Xu, J., Pazhouhandeh, M.R., Ecoffey, S., Beilliard, Y., Genov, R. and Drouin, D. (2020), In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. Adv. Intell. Syst., 2: 2000115. https://doi.org/10.1002/aisy.202000115

[4] Karunaratne, G., Le Gallo, M., Cherubini, G. et al. In-memory hyperdimensional computing. Nat Electron 3, 327–337 (2020).

[5] Kanerva, P. Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors. Cogn Comput 1, 139–159 (2009).

[6] Yao, P., Wu, H., Gao, B. et al. Fully hardware-implemented memristor convolutional neural network. Nature 577, 641–646 (2020). https://doi.org/10.1038/s41586-020-1942-4.

[7] Li, C., Wang, Z., Rao, M. et al. Long short-term memory networks in memristor crossbar arrays. Nat Mach Intell 1, 49–57 (2019). https://doi.org/10.1038/s42256-018-0001-4.

[8] Rahimi Azghadi, M., Chen, Y., Eshraghian, J.K., Chen, J., Lin, C., Amirsoleimani, A., Mehonic, A., Kenyon, A.J., Fowler, B., Lee, J.C. and Chang, Y. (2020), Complementary Metal-Oxide Semiconductor and Memristive Hardware for Neuromorphic Computing. Adv. Intell. Syst., 2: 1900189. https://doi.org/10.1002/aisy.201900189

[9] Ielmini, D., Wong, HS.P. In-memory computing with resistive switching devices. Nat Electron 1, 333–343 (2018). https://doi.org/10.1038/s41928-018-0092-2

[10] Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. et al. Memory devices and applications for in-memory computing. Nat. Nanotechnol. 15, 529–544 (2020). https://doi.org/10.1038/s41565-020-0655-z

[11] Le Gallo, M., Sebastian, A., Mathis, R. et al. Mixed-precision in-memory computing. Nat Electron 1, 246–253 (2018). https://doi.org/10.1038/s41928-018-0054-8

[12] Yi, W., Savel'ev, S., Medeiros-Ribeiro, G. et al. Quantized conductance coincides with state instability and excess noise in tantalum oxide memristors. Nat Commun 7, 11142 (2016). https://doi.org/10.1038/ncomms11142

[13] Ambrogio, S., Narayanan, P., Tsai, H. et al. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature 558, 60–67 (2018). https://doi.org/10.1038/s41586-018-0180-5

[14] Du, C., Cai, F., Zidan, M.A. et al. Reservoir computing using dynamic memristors for temporal information processing. Nat Commun 8, 2204 (2017). https://doi.org/10.1038/s41467-017-02337-y

[15] Lv, S., Liu, J., Geng, Z. (2020). Application of Memristors in Hardware Security: A Current State-of-the-Art Technology. Advanced Intelligent Systems, 3.

[16] Zhang R , Jiang H , Wang ZR , Lin P , Zhuo Y , Holcomb D , Zhang DH , Yang JJ , Xia Q . Nanoscale diffusive memristor crossbars as physical unclonable functions. Nanoscale. 2018 Feb 8;10(6):2721-2726. doi: 10.1039/c7nr06561b. PMID: 29419836.

[17] Jiang, H., Li, C., Zhang, R. et al. A provable key destruction scheme based on memristive crossbar arrays. Nat Electron 1, 548–554 (2018). https://doi.org/10.1038/s41928-018-0146-5.

[18] Nili, H., Adam, G.C., Hoskins, B. et al. Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors. Nat Electron 1, 197–202 (2018). https://doi.org/10.1038/s41928-018-0039-7.

[19] Yang, C., Hu, Q., Yu, Y., Zhang, R., Yao, Y., Cai, J. (2015). Memristor-Based Chaotic Circuit for Text/Image Encryption and Decryption. 2015 8th International Symposium on Computational Intelligence and Design (ISCID). doi:10.1109/iscid.2015.156.

[20] Jiening Wu, Lidan Wang, Guanrong Chen, Shukai Duan. A memristive chaotic system with heart-shaped attractors and its implementation. Chaos, Solitons Fractals, Volume 92, 2016, Pages 20-29, ISSN 0960-0779.

[21] P. Chen and S. Yu, "Technological Benchmark of Analog Synaptic Devices for Neuroinspired Architectures," in IEEE Design Test, vol. 36, no. 3, pp. 31-38, June 2019, doi: 10.1109/MDAT.2018.2890229.

[22] P. Chen, X. Peng and S. Yu, "NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 12, pp. 3067-3080, Dec. 2018, doi: 10.1109/TCAD.2018.2789723.