Memristors in the Context of Security and AI

***Abstract*—Over the last decade, a lot of research has been conducted on memristors. Some of this research focuses on security aspects of memristors. The current study summarizes and compares five reviews concerned with security applications of memristors and potential threats when using memristors for training and storing neural networks. Differences among these reviews show that different perspectives are necessary to get a comprehensive overview of security aspects of memristors. By synthesizing the perspectives of different reviews, this study helps to get such an overview.**

***Index Terms*—memristors, security, physical unclonable function (PUF), true random number generator (TRNG)**

I. Introduction

Memristors have repeatedly been proposed as solutions to overcome limitations of traditional computer architectures. Chua [1] was the first to theoretically describe memristors as a basic circuit element in 1971. Even though there has been controversy over the characterization of memristors as a basic circuit element [2], a lot of research has evolved around this technology. Especially since the first implementation of a memristive device in 2008 [3], many studies on memristors' characteristics and potential applications have been published.

The goal of this study is to give an overview of security aspects of memristors. For this purpose, recent reviews on this topic are examined and compared with each other. The reviews cover two general perspectives on security aspects of memristors: (1) potential security applications of memristors like physical unclonable functions (PUFs) or true random number generators (TRNGs), and (2) security threats when using memristors for AI applications and possible countermeasures. Both of these perspectives are considered in this study. After providing some background information on memristors and security primitives in Section II, the reviews included in this study are compared (Section III) and discussed (Section IV), before drawing some conclusions in Section V.

II. Background

Memristors are circuit elements with a variable resistance that depends on the amount of electric charge flowing through a memristor until a certain point in time. Thus, the resistance of a memristor can be modified by applying a voltage bias. By distinguishing different levels of resistance, information can be stored on memristors. Usually two levels of resistance are distinguished for this purpose, a high resistance state (HRS) and a low resistance state (LRS) (corresponding to 0 and 1 in binary code).

Various materials have been used to fabricate memristors, but they all have a similar structure consisting of two electrodes with an active layer in-between them. The resistance of the active layer changes when a voltage bias is applied, based on different mechanisms depending on the material used [2]. To efficiently assemble multiple memristors, they are usually arranged in crossbar arrays.

The state of a memristor can be non-volatile or volatile. Non-volatile memristors are especially suitable to be used as memory devices because they are much faster than other non-volatile memory technologies. Volatile memristors switch from HRS to LRS when applying a voltage bias, and spontaneously switch back into HRS after removing the voltage bias. This behavior allows to emulate biological neurons, which also return to their resting state after being stimulated [4]. This makes memristors a promising technology for neuromorphic computing.

Another area of application for memristors is in-memory computation. Performing in-memory computations allows to bypass the von Neumann bottleneck because no data needs to be transferred between memory and computing unit. The capability of memristors to perform in-memory computation relies on the fact that data can be stored via resistance levels, and the resistance can be modified by applying a voltage bias. Thus, by encoding data into a voltage pulse (i.e., the amplitude and duration) that is applied to a memristor, the data stored on the memristor can be modified [2]. In particular, this allows to perform in-memory logic operations or in-memory vector-matrix multiplications [5].

Besides using memristors for data storage and advanced computation tasks, they have also been proposed for security applications. The two main security applications of memristors are PUFs and TRNGs.

PUFs exploit the intrinsic randomness of hardware devices that arises from small manufacturing variations. A PUF can be queried with an input (challenge) and returns a response that depends on the device and the challenge. Due to the intrinsic randomness of the device, the responses are unpredictable. Desirable characteristics of PUFs are uniqueness (i.e., different devices should return different responses for the same challenge), reliability (i.e., a PUF should repeatedly return the same response for the same challenge, even under different environmental conditions), and unclonability (i.e., for a given device, it should be difficult to manufacture another device that returns the same responses) [6].

PUFs can be divided into weak and strong PUFs [6]. Weak PUFs are characterized by having a high reliability, but they only provide a small number of challenge response pairs (possibly only one). Thus, the security of a weak PUF depends on not disclosing the response. By contrast, strong PUFs provide a large number of challenge response pairs, so that it is infeasible to enumerate all of these pairs. Therefore, the security of a strong PUF does not depend on keeping responses

secret, as long as challenge response pairs are only used once.

A specific type of PUFs are public PUFs (PPUFs) [7]. PPUFs are based on a publicly available simulation model for a PUF device. Running the simulation model takes much more time than using the physical device to generate a response for a given challenge. This allows to verify that someone has access to the physical device. The verifier can choose a challenge and simulate the response, send the response to the verifying party and ask for the corresponding challenge. Then it is feasible to test all possible challenges with the physical device, but not with the simulation model.

TRNGs are a security primitive for generating random numbers, which are vital for various cryptographic applications. TRNGs are based on unpredictable physical processes and therefore provide a high degree of randomness, but they are slow in generating random number sequences. By contrast, pseudo random number generators (PRNGs) provide an efficient method to generate long sequences of numbers. However, since PRNGs use a deterministic algorithm that only depends on its initial state (seed), their output is not truly random. A typical setup to combine the strengths of PRNGs and TRNGs is to use a TRNG to generate a seed for a PRNG, and then use the PRNG to generate many random numbers.

III. Reviews on security aspects of memristors

For the following overview and discussion, the reviews [8]–[12] are considered. These reviews have been selected by searching for the most relevant reviews concerned with security aspects of memristors in Google Scholar. For this purpose, the first five reviews among all articles since 2019 have been selected when using the search string "memristors security" and sorting the results by relevance (as provided by Google Scholar on 29 October 2023). The search was restricted to articles since 2019 to focus on the most recent developments, while covering enough time to include relevant reviews. This becomes evident when varying the time span for the search: when considering all articles since 2018, the selection of reviews does not change (compared considering all articles since 2019), while reviews with high citation impact would be excluded when reducing the time period even more.

Among the aforementioned reviews, [8]–[10] are the most relevant ones according to their citation counts as they all received more than 20 citations, whereas the reviews of Singh [11] and Zou et al. [12] both have less than ten citations (until November 2023). Even though these numbers have to be interpreted carefully because of the different publication years of the reviews, they show a clear bimodal distribution in terms of their citation impact.

Regarding the content of the reviews, [8]–[11] have a focus on the usage of memristors for security applications. While the reviews of Pang et al. [9], Lv et al. [8], and Singh [11] consider memristors in general, Wang et al. [10] focus on volatile memristors. Zou et al. [12] take a fundamentally different perspective on security aspects of memristors by describing security threats and possible countermeasures when using memristors for training and storing neural networks. The remainder of this section summarizes these reviews and describes differences between them.

A. Memristor-based PUFs

One major security application of memristors is to use them as PUFs. The reviews [8]–[11] give an overview of studies proposing memristor-based PUF designs. Fig. 1 visualizes the coverage of studies in the reviews. The figure illustrates that Pang et al. [9] consider a lot more studies than the other reviews. Lv et al. [8] cover considerably more studies than Wang et al. [10] and Singh [11], which both only cover three studies introducing approaches to implement memristor-based PUFs. Whereas both Pang et al. [9] and Lv et al. [8] both consider studies that no other of the four reviews covers, all studies considered by Wang et al. [10] or Singh [11] are covered by Pang et al. [9] or Lv et al. [8].

Pang et al. [9] distinguish four types of memristor-based PUFs. Two of these types are memristor-based weak and strong PUFs, following the general distinction between weak and strong PUFs described in Section II). Memristor-based weak PUFs exploit small variations in the switching probability (i.e., resistance variability) between different memristor cells of a crossbar array. The general idea behind this approach is to apply a voltage bias to memristor cells such that their switching probability is close to 50%. Then, the memristor cells randomly settle to a state, and these states can be used as a sequence of random bits. A straightforward approach to implement memristor-based strong PUFs is to compare the resistance between two memristor cells of a crossbar array. Due to the resistance variability of memristors, this also allows to generate random bits. In this approach, the position of the two cells to compare can be regarded as a PUF challenge. This yields a large number of possible challenge-response pairs, making the approach a suitable strong PUF design. Another approach to implement memristor-based strong PUFs mentioned in [9] is to use memristor-based arbiter PUFs or memristor-based ring oscillator PUFs.

Besides the distinction of memristor-based weak and strong PUFs, Pang et al. [9] identify two further types of memristor-based PUFs corresponding to specific use cases. The first of these use cases are memristor-based PPUFs. As described in Section II, a publicly available simulation model of a PPUF device (here: a memristor crossbar) serves as a verification tool. The simulation model for a memristor crossbar is based on the physical characteristics of its memristor cells, and simulation requires computationally heavy operations. Simulating the output for a large memristor crossbar is infeasible because this would require to simulate all possible current paths, which grow exponentially with the crossbar's size. However, simulating the output of a subset of memristor cells is feasible. In contrast to the simulation model, the corresponding physical device can be used to compute the output of the whole memristor crossbar within a reasonable amount of time. This discrepancy in runtime between simulation model and physical device can be exploited to define an authentication protocol [13].

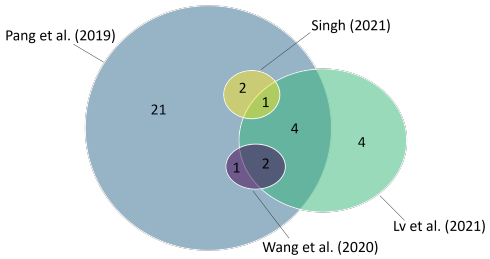

Fig. 1. Number of studies introducing a memristor-based PUF approach that are covered in the reviews of Pang et al. [8]–[11].

The last type of memristor-based PUFs identified by Pang et al. [9] is characterized by its strong resistance machine learning attacks. Machine learning attacks can be a threat to PUFs if there are strong correlations among the PUF's challenge response pairs. The proposed design uses two layers of memristor cells. The challenge specifies a set of cells in the first layer, and the (random) resistance states of these cells in the first layer are used to select one cell in the second layer. The output of this cell in the second layer is then used as the PUF's response bit. This approach is better suited to prevent machine learning attacks than a PUF design based on a single memristor layer or a conventional arbiter PUF.

Similar to Pang et al. [9], Lv et al. [8] also identify nano-PPUFs as one type of memristor-based PUFs. However, apart from this type of memristor-based PUFs, Lv et al. [8] propose a different categorization compared to Pang et al. [9]. One of the corresponding types of memristor-based PUFs that Lv et al. [8] refer to are hybrid memristor-CMOS PUF circuits. In this setup, memristor cells are combined with ordinary CMOS-based PUF designs like arbiter PUFs or ring oscillator PUFs. Since the structure of memristors is compatible with the structure of CMOS hardware, they can be efficiently integrated in one device. Combining memristors and CMOS hardware in one device can not only increases a PUF's security, but also allows to reduce hardware resources and avoid costly postprocessing compared to PUFs solely based on CMOS hardware. Furthermore, Lv et al. [8] identify a PUF design based on a diffusive (volatile) memristor crossbar as another type of memristor-based PUFs. For this PUF design, Lv et al. [8] refer to a study that is categorized as a memristor-based weak PUF approach in [9].

Finally, Lv et al. [8] describe a memristor-based approach to prove the destruction of cryptographic keys stored on a device. Such a functionality allows to revoke a key on a remote device or restrict its validity. Proving the destruction of keys on CMOS-based devices is difficult due to their volatile nature, whereas the non-volatile storage capability of memristors can be exploited for such a task. The idea of this approach is to use the resistance variability in the LRS among the cells of a memristor crossbar array to generate a fingerprint of a device. The same crossbar array is used to store a cryptographic key. Then, the fingerprint can only be extracted if no data (i.e., no cryptographic key) is stored on the crossbar array, because only in this case can the resistance in the LRS be measured for all the cells. Since the variability in the LRS depends on random manufacturing variations, this approach can be regarded as a PUF design.

Wang et al. [10] focus on volatile memristors and describe only one study introducing a PUF design. This study is also included in [9] (referred to as weak PUF) and [8] (referred to as diffusive memristor-based PUF). Besides this study, Wang et al. [10] mention two other studies that are only cited as exemplary PUF designs based on non-volatile memristors without describing their approaches. Singh [11] also only considers three studies for an overview of memristor-based PUFs. Based on these three studies, Singh [11] identifies two types of memristor-based PUF designs. First, a design based on a memristor crossbar array is sketched that follows the idea of memristor-based weak PUFs described in [9]. Second, Singh [11] identifies a memristor-based PPUF design, similar to and based on the same study as in [9] and [8].

B. Memristor-based TRNGs

The second major security application of memristors are memristor-based TRNGs. Similar to memristor-based PUFs, the reviews [8]–[11] give an overview of memristor-based TRNG approaches. Again, Pang et al. [9] cover more studies than the other reviews (see Fig. 2). The reviews generally cover fewer studies proposing memristor-based TRNG designs than studies proposing memristor-based PUF designs. In contrast to the studies on memristor-based PUFs, there is less overlap between the reviews regarding studies on memristor-based TRNGs. For example, each review considers at least one study that is not considered by any of the other reviews.

A straightforward approach to implement memristor-based TRNGs described by Pang et al. [9] and Lv et al. [8] is to exploit memristors' probabilistic switching behavior. When applying a voltage bias, the cumulative probability function of cells switching between HRS and LRS follows a lognormal distribution. The shape of this distribution depends on the amplitude and duration of the voltage bias. This can be used to generate random bit patterns with a memristor crossbar array by choosing an adequate voltage bias amplitude and duration (e.g., implying a 50% switching probability).

Another mechanism that allows to implement memristor-based TRNGs described by Pang et al. [9] is random telegraph noise (RTN). RTN occurs due to random trapping and detrapping of charge carriers, leading to sudden unpredictable changes in current levels [2]. Compared to CMOS-based TRNGs, implementations exploiting the memristors' RTN require less power and can be implemented without a preamplifier. However, RTN-based TRNGs can only achieve a low output frequency. Another drawback is that the RTN frequency and amplitude are difficult to control, which may lead to instabilities in the random number output and require postprocessing the output.

The next approach to implement memristor-based TRNGs described by Pang et al. [9] is to use current fluctuations in a memristor cell (while the cell is settled in either HRS or

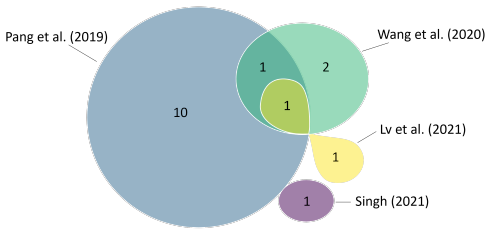

Fig. 2. Number of studies introducing a memristor-based TRNG approach that are covered in the reviews of Pang et al. [8]–[11].

LRS). More specifically, current differentials over time can be modeled to follow a Gaussian distribution. This pattern is attributed to Brownian motion, which is a random physical process. Thus, the current differences are a good source of randomness for TRNGs [14]. While using current differences for implementing memristor-based TRNGs results in a more stable output and is easier to control than RTN-based TRNGs, Pang et al. [9] identify low output frequency as a disadvantage of this approach.

Pang et al. [9] also describe a memristor-based TRNG design that provides a higher output frequency than the aforementioned approaches. In this design, resistance variations of memristor cells across set-reset cycles are used as a source of randomness. To achieve a high output frequency based on cycle-to-cycle resistance variation, a memristor with high switching speed and high endurance is necessary. Another limitation of this approach is its high power consumption due to the high switching frequency.

The last type of memristor-based TRNGs described by Pang et al. [9] uses the write delay time of a volatile memristor as a source of randomness. In particular, the delay time between a voltage pulse and the following switch of a volatile memristor from HRS to LRS is used. Since this delay time follows a random distribution, it can be used to implement a TRNG. Besides good randomness and stability, the main advantage of this approach compared to CMOS-based TRNGs and the aforementioned memristor-based TRNGs is that no postprocessing is necessary. Since this design also requires frequent switches between HRS and LRS, Pang et al. [9] mention challenges similar to those for TRNGs based on cycle-to-cycle resistance variation: the memristor must have a high endurance and high switching speed, and the repeated switching implies a high power consumption.

For describing the approach to use the write delay time of a volatile memristor to implement a TRNG, Pang et al. [9] refer to one particular study. This study is also considered in the reviews of Lv et al. [8] and Wang et al. [10]. Whereas Pang et al. [9] mention high power consumption as a challenge of this TRNG, Lv et al. [8] and Wang et al. [10] argue that it has a low power consumption compared to other TRNGs (CMOS-based TRNGs, TRNGs based on nonvolatile memristors, and further TRNGs based on volatile memristors). The viewpoint of Lv et al. [8] and Wang et al. [10] is also in line with the original study proposing this TRNG implementation [15]. However, Pang et al. [9] focus on comparing the approach to other types of memristor-based TRNGs described in their review, some of which do not require to repeatedly switch the memristor cells and therefore consume less power than the TRNG implementation of Jiang et al. [15]. Thus, the different perspectives on the role of power consumption in the design proposed by Jiang et al. [15] is a matter of reference approaches considered for comparison rather than differences in the evaluation of the approach itself.

Besides the TRNG design proposed by Jiang et al. [15], Wang et al. [10] also include a second study that proposes a TRNG design based on the write delay time in volatile memristors in their review. This implementation achieves a smaller overall circuit area compared to the design proposed by Jiang et al. [15] by replacing a comparator and a resistor with a second memristor and an AND gate. However, the operation voltage of this design is higher than for the approach of Jiang et al. [15], leading to a higher power consumption when generating random numbers [10]. Despite explicitly focusing on volatile memristors, Wang et al. [10] also consider TRNG designs based on non-volatile memristors. In particular, approaches based on RTN and probabilistic switching of memristors are described.

The review of Singh [11] considers only one study proposing a memristor-based TRNG design. In this design, memristors are included in ring oscillators, which increases the entropy compared to purely CMOS-based ring oscillators.

C. Further security applications of memristors

Lv et al. [8] and Singh [11] mention further security applications of memristors besides PUFs and TRNGs in their reviews. However, only short and high-level descriptions are provided, which is why this section only gives a brief overview of these applications.

One security application mentioned by both Lv et al. [8] and Singh [11] is to use memristors for implementing chaotic circuits. Chaotic circuits are deterministic nonlinear systems generating an unpredictable signal that can be used to encrypt messages. Memristors are suitable components for chaotic circuits due to their nonlinear behavior. Other security applications of memristors mentioned by Singh [11] are using them for tamper detection, forensics, or efficient encryption techniques. Tamper detection and forensics may be facilitated by the phenomenon that the resistance of memristors slightly changes as a result of read operations. This reduction in resistance allows to detect unauthorized access to data stored on a memristor. Efficient encryption of data stored on memristors can be achieved based on the randomness of sneak paths in a memristor crossbar array.

D. Security threats of using memristors in AI

Zou et al. [12] are not concerned with security applications of memristors in their review, but with security threats when using memristors for training and storing neural networks. Memristors can be a valuable technology for these tasks due to their capability to perform in-memory computations and

their potential non-volatility. However, using memristors for training and storing neural networks may also bring along security threats. When discussing such threats and potential countermeasures, Zou et al. [12] distinguish between black-box and white-box attack models. The black-box attack model assumes that the attacker can manipulate inputs, observe outputs, and observe side-channels. In the white-box attack model, the attacker also has access to the trained weights of a neural network model.

One type of black-box attacks considered by Zou et al. [12] are learning attacks. The goal of learning attacks is to steal a proprietary neural network[1] that is not openly available itself, but for which input-output pairs can be collected (e.g., by querying the model with selected inputs). The input-output pairs can then be used to train a new neural network, which predicts the first model's predictions. If an attacker has physical access to a device storing the weights of a neural network, memristors can help to prevent learning attacks. For certain memristors, their resistance changes over time or with every read operation. Thus, data stored on such memristors needs to be refreshed periodically or after some read operations. If a neural network is stored on such a device and refreshing the memristor cells is only allowed for authorized users, an unauthorized user can only collect a limited number of input-output pairs. This can prevent stealing the model stored on the device because sufficient training data is necessary for achieving a good performance of a neural network.

The second type of black-box attacks described by Zou et al. [12] are side-channel attacks. Side-channel attacks exploit side effects of a device during operation such as power consumption or runtime to extract confidential information. Memory access patterns have been described as a possible target for side-channel attacks to gain information about the structure of a neural network [16]. Zou et al. [12] argue that this attack may also be possible for neural networks stored on memristive devices. A countermeasure against such attacks are oblivious RAM algorithms, which hide memory access patterns [12].

Assuming a white-box attack model implies more security threats for neural networks stored on memristive devices than a black-box attack model. Since memristors can be non-volatile, weights can be stored permanently on memristive devices. Thus, if an attacker has physical access to a memristive device storing the weights of a neural network, the attacker may be able to simply read them or use microprobing techniques [12].

The general strategy to mitigate this threat is to modify a neural network's weights such that no inference is possible any more without additional information. A straightforward way to achieve such an obfuscation of neural network weights stored on a memristive device is to encrypt each weight. However, this approach adds considerable overhead due to repeated encryption and decryption operations during inference. This overhead can be reduced by only encrypting the most significant weight of each layer of the neural network. Without the most significant weight in each layer, no reasonable inference is possible any more [12]. Another approach to protect a neural network's weights is to permute them. The permuted weights allow efficient inference if the permutation applied to the original weights is known. Otherwise, the permuted weights do not reveal enough information to perform inference operations.

To bind a neural network to a specific device, the aforementioned techniques to obfuscate weights can be combined with fingerprinting the device. The randomness inherent to memristors can be used for this purpose, similar to memristor-based PUFs or TRNGs.

[1]Learning attacks can be applied to any type of machine learning model that calculates predictions for given inputs. Here, only neural networks are considered because Zou et al. [12] also focus on neural networks.

IV. Discussion

Even though the reviews [8]–[11] all deal with security applications of memristors, they cover different sets of studies. Pang et al. [9] generally include considerably more studies than the other reviews. Likewise, Lv et al. [8] cover considerably more studies on memristor-based PUF approaches than Wang et al. [10] and Singh [11], but these three reviews differ only slightly with regard to the coverage of studies on memristor-based TRNG approaches. Thus, Pang et al. [9] provide the most extensive overview of memristor-based PUFs and TRNGs among the reviews. The relatively small number of studies included in Wang et al. [10] can be attributed to their limited focus on volatile memristors.

Lv et al. [8] and Singh [11] provide a wider perspective on security applications than Pang et al. [9] by mentioning additional applications besides PUFs and TRNGs. However, the reviews of Lv et al. [8] and Singh [11] contain shorter and rather high-level descriptions compared to Pang et al. [9], which makes it difficult to get a complete picture of security applications of memristors.

Pang et al. [9] and Lv et al. [8] both provide a categorization of memristor-based PUFs. However, the two reviews refer to different categories for this purpose. Pang et al. [9] consider potential use cases for their categorization: the distinction between memristor-based weak and strong PUFs is rather general, whereas memristor-based PPUFs and the PUF design aimed at defending against machine learning attacks are rather specific use cases. The two specific PUF approaches can also be regarded as strong PUFs because they allow to generate a large number of challenge response pairs.

Lv et al. [8] consider not only potential use cases (PPUFs, proving key destruction), but also particular characteristics of the memristive devices (e.g., hybrid memristor-CMOS structure vs. memristor crossbar arrays, volatile vs. non-volatile memristors) for their categorization of memristor-based PUFs. While a categorization based on potential use cases allows to select the right approach for a given problem, referring to characteristics of memristors gives a better overview of existing techniques to construct memristor-based PUFs.

The memristor-based TRNGs described by Pang et al. [9] are categorized based on the sources of randomness on which the TRNG designs are based upon. The other reviews provide

no such categorization of memristor-based TRNGs, but only describe the small number of studies they include individually without grouping them into generalized categories.

The fact that security applications of memristors apart from PUFs and TRNGs (e.g., chaotic circuits) are either not even mentioned in the reviews or only very briefly suggests that such applications do not play an important role. However, if a review aims to give a comprehensive overview of security applications of memristors, all applications discussed in the literature should be considered. Alternatively, if reviews focus on certain applications, this restriction should at least be stated explicitly.

The review of Zou et al. [12] shows that the usage of memristors for learning and storing neural networks may also have security implications. Some security threats and countermeasures mentioned in this review specifically apply to memristors (e.g., side-channel attacks on memristors, or using resistance drift in memristors to thwart learning attacks). However, other threats and countermeasures are relevant independently of the underlying hardware. For example, learning attacks or encrypting the weights of a model can be performed on memristive as well as other devices. This distinction between techniques specifically aiming at memristive devices and hardware-agnostic techniques is somewhat unclear in [12], which makes it difficult to assess security implications of memristors in artificial intelligence systems. Nevertheless, reviewing security threats and countermeasures when using memristors for learning and storing neural networks adds an important perspective on security aspects of memristors.

V. Conclusion

Over the last decade, various studies have addressed security aspects of memristors, and several reviews have synthesized parts of this research. The current study summarizes and compares five of these reviews that have been published recently. The differences among the reviews show that each of them has a different focus. With regard to security applications of memristors, Pang et al. [9] provide the most extensive overview of memristor-based PUFs and TRNGs, whereas Lv et al. [8] and Singh [11] provide a wider perspective by considering further security applications in addition to PUFs and TRNGs. Wang et al. [10] has a narrow focus on volatile memristors, but is also concerned with security applications like PUFs and TRNGs. By contrast, Zou et al. [12] contribute a very different perspective by addressing security threats and countermeasures when using memristors for training and storing neural networks.

The most relevant reviews on security aspects of memristors among those published since 2019 (according to Google Scholar) have been considered for this study. While these reviews are likely to be representative for the corresponding research field, further reviews may provide additional perspectives and consider other studies than the reviews considered here. For example, the role of memristors' material or a comparison of metrics for measuring PUF performance may be examined. Such perspectives should also be taken into account for a comprehensive picture of security aspects of memristors.

References

[1] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] M. Lanza, A. Sebastian, W. D. Lu, M. Le Gallo, M.-F. Chang, D. Akinwande, F. M. Puglisi, H. N. Alshareef, M. Liu, and J. B. Roldan, "Memristive technologies for data storage, computation, encryption, and radio-frequency communication," *Science*, vol. 376, no. 6597, Jun. 2022.

[3] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[4] Z. Wang, S. Joshi, S. Savel'Ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo *et al.*, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, no. 2, pp. 137–145, 2018.

[5] X. Yang, B. Taylor, A. Wu, Y. Chen, and L. O. Chua, "Research progress on memristor: From synapses to computing systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, pp. 1845–1857, 2022.

[6] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[7] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Information Hiding*, S. Katzenbeisser and A.-R. Sadeghi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 206–220.

[8] S. Lv, J. Liu, and Z. Geng, "Application of memristors in hardware security: A current state-of-the-art technology," *Advanced Intelligent Systems*, vol. 3, no. 1, 2021.

[9] Y. Pang, B. Gao, B. Lin, H. Qian, and H. Wu, "Memristors for hardware security applications," *Advanced Electronic Materials*, vol. 5, no. 9, 2019.

[10] R. Wang, J.-Q. Yang, J.-Y. Mao, Z.-P. Wang, S. Wu, M. Zhou, T. Chen, Y. Zhou, and S.-T. Han, "Recent advances of volatile memristors: Devices, mechanisms, and applications," *Advanced Intelligent Systems*, vol. 2, no. 9, 2020.

[11] J. Singh, "Implementation of memristor towards better hardware/software security design," *Transactions on Electrical and Electronic Materials*, vol. 22, no. 1, pp. 10–22, 2021.

[12] M. Zou, N. Du, and S. Kvatinsky, "Review of security techniques for memristor computing systems," *Frontiers in Electronic Materials*, vol. 2, 2022.

[13] J. Rajendran, G. S. Rose, R. Karri, and M. Potkonjak, "Nano-ppuf: A memristor-based security primitive," in *2012 IEEE Computer Society Annual Symposium on VLSI*, 2012, pp. 84–87.

[14] Z. Wei, Y. Katoh, S. Ogasahara, Y. Yoshimoto, K. Kawai, Y. Ikeda, K. Eriguchi, K. Ohmori, and S. Yoneda, "True random number generator using current difference based on a fractional stochastic model in 40-nm embedded reram," in *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2016, pp. 4–8.

[15] H. Jiang, D. Belkin, S. E. Savel'ev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, M. Rao *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nature communications*, vol. 8, no. 1, p. 882, 2017.

[16] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.