Nanoelectronic Security Design for Resource Constrained Internet of Things Devices

Mesbah Uddin, Md Badruddoja Majumder, and Garrett S. Rose

IEEE Consumer Electronics Magazine (CEM), Oct 2018.

The online home for this paper may be found at: http://seneca.eecs.utk.edu/

Citation Information (BibTex):

```
@INPROCEEDINGS{Mesbah-CEM:2019,
  author="M. Uddin and M. B. Majumder and Garrett S. Rose
  title="Nanoelectronic Security Design for Resource
  Constrained Internet of Things Devices
  booktitle="IEEE Consumer Electronics Magazine
  month="Oct",
  year="2018",
```

Nanoelectronic Security Design for Resource Constrained Internet of Things Devices

Mesbah Uddin, *Student Member, IEEE,* Md. Badruddoja Majumder, *Student Member, IEEE,*
Garrett S. Rose, *Member, IEEE*

***Abstract*—The term internet of things (IoT) is often defined in a variety of different ways but typically refers to ordinary electronic devices and systems (e.g., toaster ovens and wearable sensors) with added computational capabilities that utilize internet connectivity. Due to the embedded nature of IoT systems, their computer systems are resource constrained in the sense that they must be small and consume minimal power. At the same time, the IoT concept introduces new security concerns that must be addressed at an early stage of the IoT device design process. Since security operations such as encryption and authentication are often resource hungry, solutions are needed that provide reasonable levels of security with minimal area and power consumption. Emerging nanoscale technologies are also being developed which require very low area utilization and low power consumption, important features for IoT systems. Here we present a couple examples of security systems built from nanoscale electronic devices that could provide solutions for low resource security in emerging IoT devices.**

***Index Terms*—Internet of Things, IoT, IoT security, Physically unclonable function, PUF, emerging nanotechnology, RRAM, hardware security.**

I. Introduction

The "internet of things (IoT)" concept creates intriguing opportunities that make our daily life smarter and easier. Things (i.e. IoT devices) related to our daily lives are increasingly being connected to Internet such that they can communicate with one another in a system of IoT as shown in Fig. 1. The connectivity provided for everyday personal devices such as smart phones, smart watches, and cars, to name a few, under the IoT concept poses overwhelming security and privacy concerns. For example, IoT systems require the information gathered from our daily personal devices for building smart applications such as smart health monitoring, smart car navigation, smart home, smart banking and so on. However, adversaries may take advantage of such information shared over the internet to perform malicious activities. Therefore, security is one of the major constraints for designing such IoT applications [1]. Security criteria for preventing device piracy and counterfeiting are also important for end users to trust and validate the authenticity of their IoT hardware.

IoT devices are mostly small, battery powered devices. Therefore, size and power are two critical design constraints of IoT devices. As IoT devices are often communicating with one another, they must maintain a strict timing schedule. In general, most IoT devices also require fast operation for real time applications. As existing CMOS technology has almost come its limit for scaling, design for performance improvement is becoming more and more challenging. On the other hand, emerging nanoscale technologies (e.g. memristors, carbon nanotubes, graphene) show great promise for building small scale and energy efficient hardware, including emerging security primitives. This article focuses on the prospect of nanoelectronic design as a solution to building security primitives for resource constrained IoT devices.

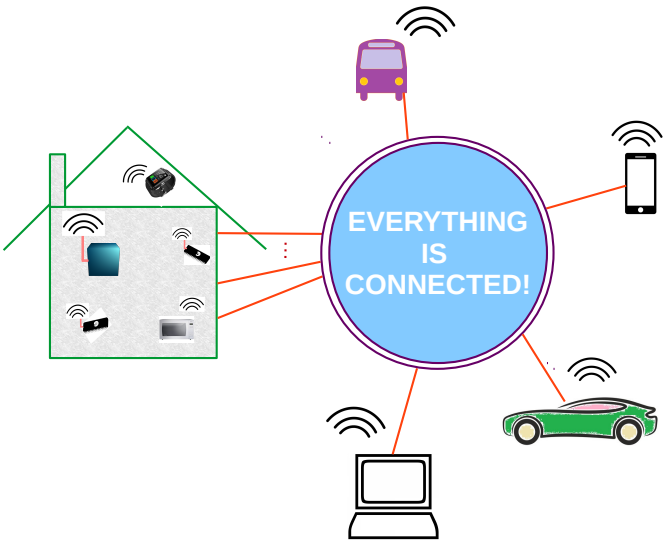

Fig. 1. The Internet of Things (IoT) where every smart device and apparatus are connected to the internet.

Hardware security [2] using nanoelectronic technologies shows promise for area and energy efficient implementations for the 'things' in 'Internet of Things'. A physically unclonable function or PUF [3] is one such system where the security protocol is laid out completely at the hardware level. Memristors, especially metal-oxide memristors or ReRAMs [4] are nanoscale devices that can be used to implement PUF architectures. Memristors are non-volatile, CMOS-compatible and offers high retention time, endurance and multi-level resistances. Different memristor based PUF circuits and systems are proposed in [5]–[9]. Rose *et. al.* extended the idea of his initially proposed absolute write-time based memristor PUF [5] and presented a PUF built from a dense crossbar array of memristors named as XbarPUF [7]. This XbarPUF

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0301. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

is expected to be an improved design compared to earlier versions. Later in this article, the basic functionality, security aspects, and overhead analysis of an XbarPUF is presented from the viewpoint of IoT.

Another useful security primitive for IoT is memory integrity checking that leverage the data dependency of sneak path currents in resistive nanoscale crossbar memory architectures. Usually, memory integrity checking is performed using computationally heavy crypto primitives such as hash functions or Message Authentication Codes (MAC). This becomes a challenge to incorporate those primitives in IoT devices due to limited availability of resources and strict timing requirements. Researchers have started working toward more lightweight methods for integrity checking which might be applicable to embedded systems and IoT applications [10] [11]. However, with existing technology, it is still a challenge to fit these primitives in highly resource constrained IoT devices. An integrity checking method based on nanoelectronic components has been found promising in this regard due to significantly low implementation overhead in terms of area, power and delay [12]. In this article, such an integrity checking method based on in-memory computation of nanoelectronic memory is presented as one of the promising opportunities of nanoelectronics in IoT devices.

II. IoT design requirements and challenges

The internet of things broadly defines the networking of embedded computing systems, often including small processing units, sensors, actuators, and networking systems. One example of an IoT device is the Nest thermostat [13] which provides remote access and control for users, machine learning capabilities to adapt to user needs and other features that work to make for increased convenience in everyday life. In fact, the "Things" in IoT are often basic, everyday devices and components that have not historically included computational elements, let alone networking capabilities. Beginning with supply chain management systems and including wearable health monitoring devices, IoT has emerged as a major challenge in terms of network loads (billions to trillions of interconnected devices) and also the low power requirements for such simple IoT devices.

As deployable embedded systems, IoT devices are vulnerable to a variety of security threats, including side-channel attacks (e.g. power analysis), device piracy and counterfeiting, tampering, malicious memory modifications, and many threats typical of any networked computer system. Several of these threats apply specifically to systems that exist "in the field" where authorized operators have limited physical access but the systems are often exposed to unauthorized access. For example, infrastructure monitoring systems, whether for smart grid applications or other civil engineering tasks, often come with little in the way of security safe guards. The reason security features are often lacking in such deployable IoT devices is a direct result of the limited resources (area, power, etc.) available to operate these devices. Thus, new approaches to designing IoT devices are needed that provide ample security capabilities within a very small area footprint and with very low power consumption.

III. Opportunities with Nanoelectronics

The basic building block of modern integrated circuits (IC), the silicon transistor, has been regularly shrinking following the predicted trends of Moore's Law, from a few micrometers in the 1960s to tens of nanometers today. This scaling behavior has led to most of the performance and energy efficiency advantages we often take for granted in modern electronic systems. However, the trend of Moore's Law scaling is now seriously under threat due to scaling limits for bulk-silicon technologies. In order to continue the miniaturization of basic circuit elements, and thus continue to enjoy subsequent performance benefits, researchers have been investigating nanoscale technology alternatives.

A. The Nanoscale Memristor

One nanoelectronic alternative is the memristor. The term 'memristor' (memory resistor) was first coined by Chua in 1971 [14]. Later, in 2008, nanoscale metal-oxide resistive switching devices were demonstrated by HP Labs to exhibit behavior predicted by Chua almost 40 years earlier [15]. The memristor is sometimes considered as the fourth fundamental circuit element after the resistor, capacitor and inductor. Physically, a memristor is a two-terminal device with a resistance that depends on the "past history" of applied voltage. In most applications, memristors are modeled as binary (or multi-level) resistors i.e. there is a high resistance state (HRS) and a low resistance state (LRS) where the resistance states can be changed based on the magnitude and direction of an applied voltage. Since the systems are considered at a behavioral level of operation, we use the term 'memristor' generally to refer to any of a variety of two-terminal resistive switching devices with memory. Although, a variety of combination of materials may display memristor behavior, metal-oxide memristors (HfO_x, TiO_x, TaO_x etc.) are compatible with existing CMOS fabrication technology and are, thus being explored extensively by researchers worldwide.

The versatile nature of nanoscale memristor technology and compatibility with conventional, silicon-based technologies motivate several potential applications. Memristors are non-volatile, which means they can retain their state even if the power supply is removed. Due to their non-volatility, these devices can be thought of as a ultra-high density successor to persistent memory systems such as Flash technology. Furthermore, memristors have beeen developed with exhibit access times comparable to DRAM—leading to possibilities for high-speed persistent memory. An interesting, related example of such technology now emerging in industry is the Intel/Micron 3D XPoint technology [16]. Given the many potential advantages of nanoscale resistive switching technologies (e.g. memristors), its no wonder these technologies now appear to be on the horizon for commercial applications.

B. Security for Nano-Enabled IoT

A basic premise of the ideas discussed here is that memristor-based technology, specifically ultra-dense non-volatile memory, has a place in emerging IoT devices. The

rationale is that memristor memory both utilizes a small amount of on-chip area and also consumes very little power, both important requirements for IoT systems. Furthermore, for systems that are only powered when needed, consider smart cards or wearable technologies, persistent memory provides a useful means for maintaining system state information.

As these nanoscale technologies continue to emerge, relevant security concerns must also be taken into consideration. Here we focus on two areas of security that are critically important for IoT: (1) device authentication to mitigate design piracy and (2) integrity checking to mitigate the threat of malicious memory modification. In both application examples, memristor memory arrays are leveraged to implement "compute in memory" style security primitives that address these challenges. Specifically, much of the computation is performed using the inherent behavior of memristors. This leads to a dual-use for these systems where they can normally be used simply as memory but can switch over to performing important security functions when needed. Since the same hardware is leveraged for two sets of operations, further savings in area can be saved for resource limited IoT device designs.

IV. Resource Efficient Physical Unclonable Functions (PUF)

A. PUF Overview

Typically an IC manufacturer produces many copies of the same functional IC. However, due to unavoidable variations arising from manufacturing processes, there will exist local timing and power mismatch among these different copies of ICs. These tiny process variations are outside the control of manufacturers and it is therefore almost impossible to reproduce the exact same copy of an IC chip. Physical unclonable functions or PUFs [17] leverage these tiny differences among identical IC implementations into useful hardware-specific signatures or keys. Therefore, a PUF would have these three basic characteristics:

1) *Unclonable*: Hard to make a physical clone or hard to make a mathematical model that simulates the behavior of the physical structure.
2) *Uncontrollable*: PUFs utilize the process variation of a set of device with identical nominal values. Even the manufacturer can't control these tiniest variations.
3) *Unpredictable*: The response of a PUF depends on complex internal hardware structure and is, therefore, random and very hard to predict.

Most current security systems and cryptographic protocols depend on the secrecy of private keys. However, problems exist for storing a key in digital memory as memory can potentially be read externally by an adversary. In a PUF-based security protocol, there is no need for storing a secret key as the key is a function of the hardware itself (i.e. generated by the PUF). The generated key would also be non-volatile, such that a continuous power supply is not needed.

A PUF can be modeled as a challenge-response system. The hardware generates a response based on its internal process variations and also an input challenge provided by an authorized user. This concept is shown in Fig. 2.

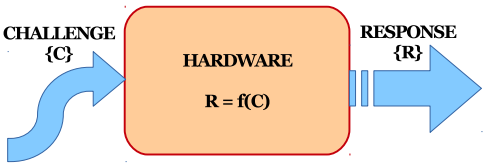

Fig. 2. Challenge-Response System

Many different types of PUFs have been proposed in literature. Arbiter PUF (APUF) [19] and ring-oscillator PUF (RO-PUF) [20] could be the most studied among those. Both of them are delay-based PUFs i.e. depend on delay variation arising from process variations across unique implementations of an IC. Arbiter PUF utilizes variations in the propagation path delay of a chain of identically laid out switch elements [19]. RO-PUF depends on variations in frequencies among identically implemented ring oscillators where the variation in frequencies arise from delay difference among inverters inside a ring oscillator. Other PUFs include SRAM PUF, digital PUF, glitch PUF, memristor PUF etc.

B. PUF applications

There are two primary applications for a PUF:

1) Authentication, and
2) Secret key generation.

1) Authentication: A chip can be verified by applying a challenge to a PUF and comparing the generated response with stored response. However, a single challenge-response pair or CRP should not be enough for authentication because an adversary could somehow manage to read a PUF's response during a chip's authentication phase. That's why researchers have described a two stage authentication method using a PUF [20]. In the first phase, named 'enrollment', the trusted party applies randomly selected challenges and obtains the responses. The challenge response pairs generated in the enrollment phase are saved in a database. To verify the authenticity of an IC, the trusted party would choose any challenge that has been recorded during the enrollment phase and match against the stored response. To protect against a sniffing attack, when a challenge-response set is used for an authentication, it is deleted from the database and never used again, thus ensuring a secure and unique way of authentication.

2) Secret key generation: Cryptographic security protocols require secret keys for their implementations. It has been demonstrated in the literature that an adversary can extract digitally stored values (including secret keys) from a chip [21]. Thus, there has been a need for a more secure way of storing keys and PUFs come into play because a PUF key can be produced on demand and does not need to be stored digitally. Therefore, when a PUF is not in use, an external attacker can't extract that key. Instead of directly using a PUF response, it could also be used as a seed to a random number generator to generate random keys that satisfy cryptographic algorithms.

C. XbarPUF: An Example Nanoelectronic PUF

Following the near-impasse of Moore's law [22] for the past few years, researchers have been looking toward CMOS-alternative technologies to continue the progress gained from

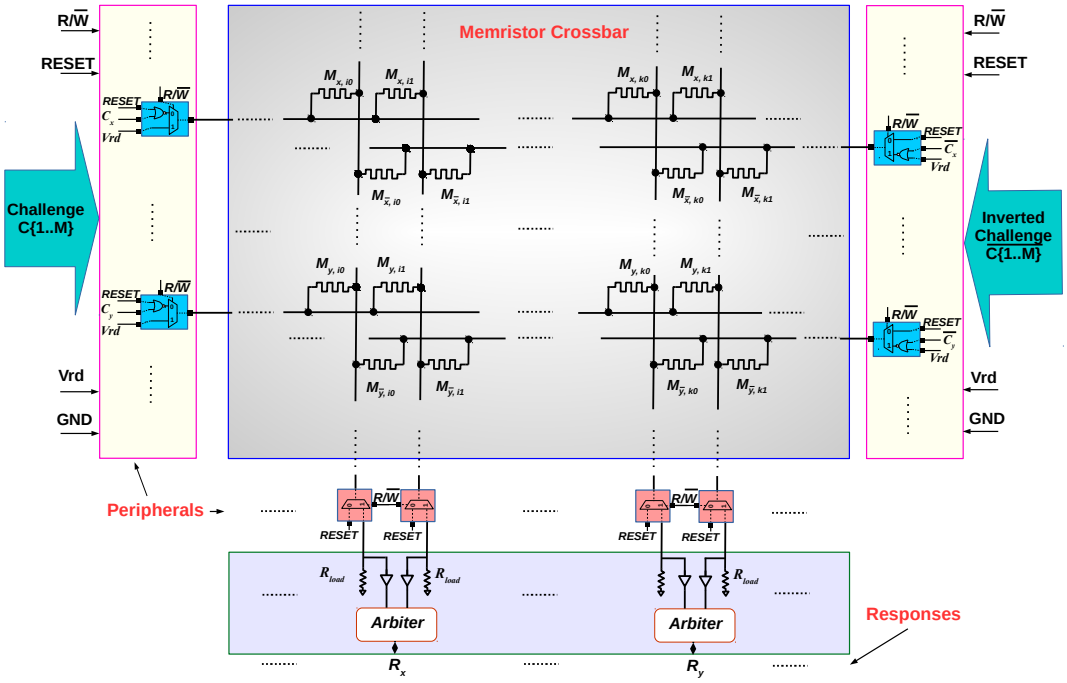

Fig. 3. Schematic of a crossbar-memristive PUF (XbarPUF)) [9], [18].

TABLE I
SECURITY PERFORMANCE OF SIMULATED A 32×2 XORED XBARPUF [9]

	Uniqueness(%)	Bit-aliasing (%)	Uniformity (%)	Reliability (%)
XbarPUF	50.40	51.50	56.50	98.40
Ideal	50	50	50	100

scaling. Emerging nanotechnologies offer some promises in this regard. Lightweight PUFs that leverage nanoscale components for lower area utilization and improved energy efficiency are being investigated.

The memristive crossbar PUF or XbarPUF was first presented in [7] and later extended in [8], [9], [18]. The primary building block of the XbarPUF is a dense crossbar array of memristors. A circuit diagram of an XbarPUF is shown in Fig. 3. XbarPUF is based on the write-time memristive PUF first described in [5]. The XbarPUF's primary entropy source is the minimum time it takes for a crossbar column of memristors to SET during a write operation. Additionally, the XbarPUF is somewhat a reimagining of the arbiter PUF (APUF) in the sense that pairs of memristive circuits "race" toward a switching event, analogous to racing delay paths. Once the output of one memristive circuit (a column in the XbarPUF array) reaches a certain threshold, an arbiter selects the "winner" thus determining the corresponding response bit. More details on the operation of the XbarPUF is provided in [7].

The key security performance metrics of the XbarPUF are listed in Table I [9]. It is clear from the table that the proposed XbarPUF displays near-ideal performance, though this result is based on simulation and the performance of actual implemented XbarPUF is yet to be determined. The acceptable values of uniqueness, bit-aliasing, and uniformity indicate the strength of producing unique responses across multiple devices, multiple input and multiple output sets. Reliability of Table I was evaluated against varying temperature for the range of 10^0C to 100^0C. As can be expected for XbarPUF, reliability doesn't degrade with changing temperature because relative delay differences between physically adjacent crossbar columns are used to evaluate the XbarPUF response and temperature should affect delays from both columns in a similar manner and thus having no relative effect. Aging of memristor devices may degrade the values of these security metrics presented in Table I for memristors with low OFF/ON ratio (typ. $<$10), as discussed in [9]. However, modern metal-oxide ReRAMs or memristors have high OFF/ON ratio, high retention time and, therefore, their slow aging rate should not degrade the OFF/ON ratio much over time and thus should not affect the performance of the PUF circuit during its lifetime.

PUFs are also found to be vulnerable to machine learning based modeling attacks [18], [23], [24] and the basic XbarPUF is not out of that scope. Detailed analysis of an XbarPUF and its variations are provided in [18]. It was found that although basic XbarPUF structure is prone to modeling attack like Arbiter PUF (APUF), by tweaking this XbarPUF circuit, only around 50% accuracy using machine learning model may be achieved which is equivalent to predicting the outcome of

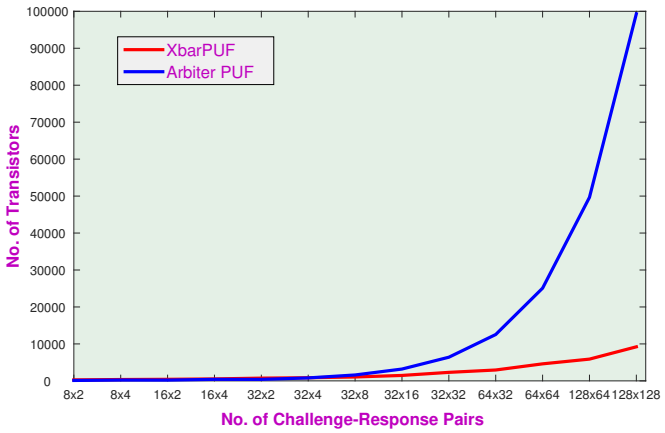

Fig. 4. Plot of area (transistor count) vs. number of challenge and response bits of a CMOS PUF (arbiter PUF) vs a memristor PUF (XbarPUF) [7].

a coin toss. Table II summaries this results in brief.

TABLE II
REVIEW OF RESULTS FOR MODELING ATTACK ON PUFS FOR 5000 CRPS AS TRAINING DATA

	APUF	**XbarPUF** [18]	**Modified XbarPUF** (xor+swap) [18]
LR	>99% [23]	99%	49%
SVM	95% [24]	96.8%	57.9%

The XbarPUF does not have the dependency on a precise write-time, unlike its predecessor [5]. Additionally, the XbarPUF has comparable performance to that of the APUF, while decreasing the transistor count (Fig. 4). A comparison between the XbarPUF and the most popular CMOS PUF or the APUF in terms of transistor count can be seen in Fig. 4. The APUF transistor count is quadratic with the number of challenge-response bits while that of the memristive PUF design is only linear.

The energy consumption of a memristor is static and thus can be quite high for some IoT devices. This is a concern for these devices. However, they can also operate on a low supply voltage, especially when the memristor's threshold voltage is low and thus can achieve very low power operation [18].

Typically, a XbarPUF would need at least twice the switching time (10ns) of a memristor to generate a response. The sensing circuitry needs negligible time compared to the memristor switching time, but it can start to dominate the total response generation time at very low voltage applications. Therefore, a tradeoff between energy and performance has to be made while integrating them with IoT devices.

A PUF has a huge number of potential applications in the context of internet of things. It should be noted, however, that some concerns do exist. Because a PUF response is a function of complex internal hardware circuitries of a chip, the responses are not 100% consistent. Even this one or a few bit errors could make PUF undesirable for key generation algorithms for cryptographic machines as mentioned earlier. Fortunately, several error correcting schemes do exist to produce reliable output from a PUF to use in cryptographic operations.

The vast majority of IoT devices are deployed in the field as a battery-powered embedded system and are, therefore, resource-constrained in terms of power and also area. Thereby, any security measures employed in these devices must consume negligible amount of power compared to the total power consumption of that device and should also be minimal in size. As can be seen from 4, the XbarPUF utilizes a very small area compared to a CMOS PUF like APUF and thus more suitable for these IoT applications. XbarPUF also consumes a small amount of power as shown in [7], [8].

The area and power advantages of security primitives, specifically PUFs, such as the XbarPUF are not as convincing when considering the need for mechanisms such as error correction. Certainly, any error correction code implemented in hardware could be expensive in terms of area and even software solutions will consume more power than desired. One solution that has been proposed recently is the idea of a weak memristive PUF where error correction is not needed [25]. Of course, the potential advantages gained depend on the required security applications.

V. INTEGRITY CHECKING VIA IN-MEMORY COMPUTATION TECHNOLOGIES

The reliability and security of any classical computing system largely depends on the integrity of data fetched from memory. The memory is often a favorite target for attack by adversaries, posing serious threats to security and reliability of computing systems. Attacks include replacing the original memory content with some malicious piece of data which may perform undesired operation in the computing device. Therefore, integrity of the data read from the memory of a computing device must be verified to make sure that it has not been tampered with by any unauthorized sources.

Crossbar resistive memory has a common feature of sneak path currents which refers to the leakage current through unselected memory cells [26]. This current causes reliability issues for writing to and reading from a selected cell. However, there are modified architectures for memristor based crossbar memory such as 1-Transistor 1-Resistor (1T1R) which mitigate sneak path problems in nano-crossbar memory [27]. The main idea of 1T1R structure is to use a selector device to restrict the current flow only to the selected memory cells. Moreover, the 1T1R structure gives flexibility to enable or disable sneak paths for a particular portion of the memory. This capability is useful for building applications that can leverage the information provided in sneak path currents.

Integrity checking requires generating a tag from the memory data and storing it whenever the memory is updated by legitimate users. Later on, the integrity of data read from the memory can be verified by regenerating the tag and comparing against the stored tag. In traditional systems, this is performed using a completely separate tag generating unit which takes the memory data as input and generates the tag from it. We can leverage sneak path currents in a crossbar memory for generating tags from stored data for integrity checking

purpose. The memory itself is used for tag computation in this technique which saves a significant amount of computational resources and is particularly suitable for computing systems with limited resources such as IoT devices.

A. Sneak Path Tag Generation

With enabled sneak paths, reading from a particular address of crossbar memory does not reflect the data stored only in that address. Rather, it reflects the whole of data stored in the memory. This sneak path based reading can be used as a tag for the stored data. A multi-bit tag can be generated by reading the memory with a single selected row and multiple selected columns.

Due to scalability limitations, an infinitely large crossbar cannot be read while also simultaneously generating tags from it. A larger crossbar causes larger current to flow through the selected columns and the generated tag bits are biased towards a logic high value. Here we consider a scheme where the memory is divided into multiple sections of considerable size and partial tags are generated from each section sequentially. Generated tags from each section are then cumulatively combined using an XOR operation. Each memory section can be enabled (tag generation) or disabled (normal read/write) for sneak paths with corresponding word line select transistors.

It can be inferred that a generated tag is more dependent on the data bits in the selected row than the bits in the unselected row. For security, every memory bit should have an equal effect on the generated tag. With a view to removing the bias of the selected row, a data row is randomly selected using a pseudo random number generator (PRNG) during every tag generation instant. Thus, generated tags are uniformly affected by each data bit. A key is applied by the authorized user to work as a seed for the PRNG and the row line is decoded accordingly. This architecture is illustrated in Fig. 5

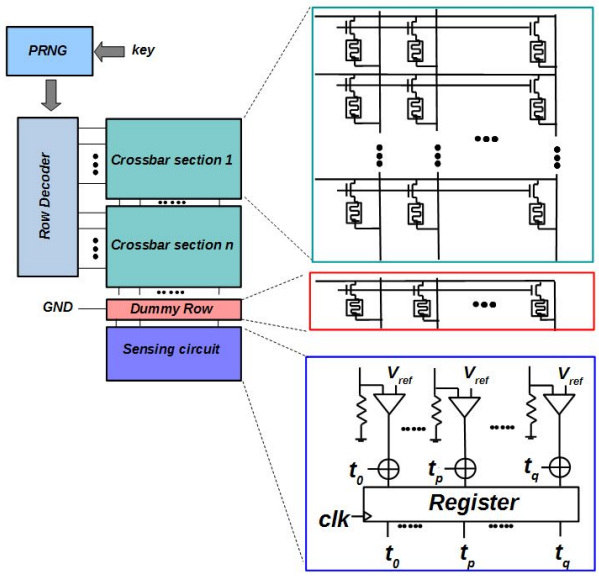

Fig. 5. Tag generation protocol. Tags are generated on individual sections of the crossbar and combined by XOR operation.

TABLE III
ANALYSES OF AVALANCHE EFFECT FOR TAG GENERATION PROTOCOL WITHOUT AND WITH A RANDOMIZED DUMMY ROW OF MEMORY CELL. SIMULATED MEMORY SIZE IS 64×64, TAG SIZE IS 16 BITS, NUMBER OF SECTIONS IN THE MEMORY IS 4.

Change in data bits(%)	Change in tag bits(%)	
	Without dummy row	with dummy row
0.1	0.7375	47.0425
0.5	0.9450	45.3675
1	0.9975	46.6375
5	1.1050	47.3175

Described tag generation method lacks the avalanche property. The avalanche property is one of the desirable properties of any crypto algorithm which ensures any tiny change made in the input causes a significant change in the outputs. Ideally, half of the output bits should be changed randomly due to even small changes in the input. To overcome this, a row of dummy bits are randomly written during every write to memory. With this technique, even though only a few bits are changed apparently in the memory, additional row of dummy bits also add to this. The idea of dummy row associated with the discussed tag generation method is illustrated in Fig. 5.

B. Security

The end goal of an adversary for circumventing the integrity checking is to generate a valid tag on a tampered data because integrity checking protocol only checks if the data matches with the tag. We discuss the security of the described integrity checking method from three attack perspectives targeting to achieve the same end goal.

1) Attack1: An attacker might try to manipulate a small number of data bits in order to minimize the changes to the tag bits. In other words, if the system lacks the avalanche effect an attacker has a greater success probability of forging data by generating a valid tag for it. For example, if q out of N number of tag bits are changed on an average due to a small change in the data, the number of possible combinations to be attempted to forge a tag with manipulated data would be N_{C_q}. Possible combinations for this attack would be maximum if half of the total tag bits are changed randomly due to any small change in data, i.e. $q = N/2$ [28].

Table III shows the change in tag bits for the discussed protocol with and without dummy row idea. The avalanche effect has been shown in terms of hamming distance between tag bits due to a change in a small percentage of input data bits. Results shown in Table III indicate that the protocol with dummy row has a hamming distance close to the ideal value of 50% while the same metric is very low for the protocol without the dummy row. Diffusion is another property which indicates how likely each tag bit is to be flipped due to a change in the memory and can be used to analyze the security against such predictability of tag bits. Analyzing the diffusion property of the discussed integrity checking protocol it is found that all bits nearly have 50% probability of changing due to small change in the data. This is shown in Fig. 6 where each bar indicates the average probability of each individual tag bit being flipped due to a small change in data.

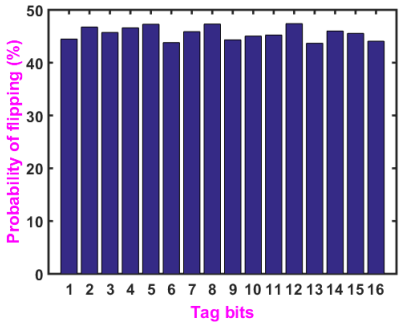

Fig. 6. Probability of flipping each bit due to small (0.1%) change made in the data for a 32×32 crossbar memory with tag size of 16 bits.

2) Attack2: If an attacker knows the key, i.e. which row to select for tag generation, he can easily forge stored data with a valid tag. However, the key is applied by the authorized user as the seed of a PRNG before every tag generation and an attacker should not have that key. If there are N row lines in the crossbar, the probability of being successful by randomly choosing a key which selects a row line for tag generation in each section is $\frac{1}{N}$. If the memory is divided into k sections, the probability of successfully selecting the correct row lines during a complete tag generation phase is $\frac{1}{N^k}$. Thus, the same memory with a greater number of sections poses more complexity for a random key guess attack. However, there is an optimization between the required security and performance as generating tags by dividing memory into more sections requires more overhead. This optimization task is left as a future improvement for this work.

3) Attack3: Another attack is a general collision based attack where the adversary randomly manipulates data in order to find another data that has the same tag as the original data had. The Collision rate is defined as the minimum number of trials needed on an average to find a collision. Collision rate for an ideal hash function having n bits is $2^{-n/2}$ [29]. By analyzing the collision rate for this integrity checking method, it is found that similar security level in terms of collision rate can be achieved with approximately 20% increase in the tag bits. However, this overhead in tag bits is negligible as the proposed method uses in memory computation for generating tags instead of a completely separate hardware.

C. Reliability

For the proposed integrity checking protocol, reliability can be defined as the ability to generate the same tag for the same data while reading between subsequent write operations. Nano-electronic circuits such as memristor based crossbar memory often suffer from variability issues due to process and environmental variations. Further, different parameters of a memristor show variation from one operating cycle to another. However, this variation has greater impact from one write cycle to another write cycle. Cycle to cycle variation for write operations does not cause a reliability concern for the discussed integrity checking method since during every new write operation a new tag is generated which will be compared with tags regenerated during intermediate read operations.

TABLE IV
COMPARISON OF IMPLEMENTATION OVERHEAD BETWEEN THE INTEGRITY CHECKING METHOD LEVERAGING SNEAK PATH BASED IN-MEMORY COMPUTATION AND THAT PROPOSED BY HONG *et al.* [10].

Overhead	Hong *et al.*	Integrity checking via in-memory computation
Energy (pJ)	53.66	6.34
Delay (No. of clock cycles)	8	4
Transistor count	13312	1538

Moreover, cycle to cycle variation from one write to another write makes the tag generation scheme more indeterministic for an adversary to forge a valid tag on tampered data. Reliability concern is mostly associated with variation in one read to another. This might happen if the read operation is destructive which means it alters the stored resistance level of the memristors. However, a read voltage below the threshold required for swithcing a memristor keeps the switching rate to a negligibly small amount and prevents the alteration of memristance level. Thus, the proposed method works reliably so long as the read voltage is below the switching threshold level. However, other factors such as supply voltage fluctuation, temperature variation can potentially be a reliability concern. This can be mitigated for all practically considered noise limits by improving the noise margin of sensing circuitry at each sampled column of crossbar during tag generation.

D. Performance

An obvious distinction between the discussed integrity checking protocol with other conventional method is the use of in memory computation for tag generation. Performance of this memory integrity checking protocol is compared with a cost effective tag generation (CETD) scheme proposed by Hong *et al.* [10]. Prototypes of both designs designed in the same platform are compared here.

Table IV shows the comparison results. It can be seen that the nanoelectronics based method has a significant improvement over the conventional one we are comparing with. Both energy consumption and transistor counts for the discussed integrity checking method exhibits nearly $10\times$ improvement. The prototype for the nanoelectronics based method requires 50% less clock cycles than the prototype of another method. The primary source of improvement for the described method comes due to the fact that in-memory computing of nanoelectronic crossbar memory facilitates the tag computation simply by a single read operation without a separate tag generating circuit. Absolute data regarding the implementation cost might vary depending on the process and implementation method. However the nanoelectronics based integrity checking method discussed here can always be inferred as a lightweight method compared to other conventional schemes and be used in IoT applications .

VI. CONCLUSIONS AND FUTURE PROSPECTS

The main contribution of this paper is to encourage and enlighten the readers about the security threats posed by ever-increasing IoT devices and how nanoelectronic hardwares

help in that regard. IoT devices are highly resource constrained in terms of size, power consumption and timing. Nanoelectronic technology offers low power, low area and faster implementation of these security protocols and are, therefore, investigated more and more. In this article, we have shown the operation of a PUF, a very promising security protocol with these technologies. Nanoelectronic PUF can be used in different security applications such as secret key generation, hardware authentication. We have also presented nanoelectronic memory technology which can be used both as a standalone memory and tag generator for memory integrity checking. Unauthorized memory modification of IoT devices can be detected with this integrity checking which reduces the possibility of a number of security vulnerabilities. These works make us optimistic about building nanoelectronic alternative of cryptographic security primitives which at present are difficult to incorporate in IoT devices due to large resource requirements. However, there exist some challenges to employ these emerging technologies everywhere as they pose some reliability issues. Because most of the discussed metal-oxide memristors are CMOS-compatible, seamless flow of back-end memristor and front-end CMOS is possible. Moreover, memristor based implementation usually should take less area than their CMOS counterpart. Therefore, the overall manufacturing cost is expected to be similar to CMOS technology, if not lower. Thus silicone area dominates while considering the cost of these devices. However, adopting any new technology takes both time and initial money investment. These challenges are something that need to be overcome by extensive research both in material science and device engineering.

VII. About the authors

Mesbah Uddin (muddin6@vols.utk.edu) is currently doing his Ph.D in Computer Engineering at the University of Tennessee, Knoxville, USA.

Md. Badruddoja Majumder (mmajumde@vols.utk.edu) is currently doing his Ph.D in Electrical Engineering at the University of Tennessee, Knoxville, USA.

Dr. Garrett S. Rose (garose@utk.edu) is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville, USA.

References

[1] O. Arias, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in internet of things and wearable devices," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 1, no. 2, pp. 99–109, April 2015.

[2] A. Sengupta, "Hardware security of CE devices [hardware matters]," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 130–133, Jan 2017.

[3] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. of the 9th ACM Conf. on Comput. and Commun. Security*, 2002, pp. 148–160.

[4] H. H. Li, Y. Chen, C. Liu, J. P. Strachan, and N. Davila, "Looking ahead for resistive memory technology: A broad perspective on ReRAM technology for future storage and computing," *IEEE Consumer Electronics Magazine*, vol. 6, no. 1, pp. 94–103, Jan 2017.

[5] G. S. Rose, N. McDonald, L. Yan, and B. Wysocki, "A write-time based memristive PUF for hardware security applications," in *Proc. of the IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, November 2013, pp. 830–833.

[6] O. Kavehei, C. Hosung, D. Ranasinghe, and S. Skafidas, "mrPUF: A Memristive Device based Physical Unclonable Function," *ArXiv e-prints*, Feb. 2013.

[7] G. Rose and C. Meade, "Performance analysis of a memristive crossbar PUF design," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2015, pp. 1–6.

[8] M. Uddin, M. B. Majumder, G. S. Rose, K. Beckmann, H. Manem, Z. Alamgir, and N. C. Cady, "Techniques for improved reliability in memristive crossbar PUF circuits," in *IEEE Comp. Society Annual Symp. on VLSI (ISVLSI)*, July 2016.

[9] M. Uddin, M. B. Majumder, K. Beckmann, H. Manem, Z. Alamgir, and N. C. Cady, "Design considerations for memristive crossbar physical unclonable functions," *ACM Journal on Emerging Technologies in Computing (JETC)*, Accepted for publication, 2017.

[10] M. Hong, H. Guo, and S. X. Hu, "A cost-effective tag design for memory data authentication in embedded systems," in *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*. ACM, 2012, pp. 17–26.

[11] T. Liu, H. Guo, S. Parameswaran, and X. S. Hu, "Improving tag generation for memory data authentication in embedded processor systems," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2016, pp. 50–55.

[12] M. B. Majumder, M. Uddin, G. S. Rose, and J. Rajendran, "Sneak path enabled authentication for memristive crossbar memories," in *Hardware-Oriented Security and Trust (AsianHOST), IEEE Asian*. IEEE, 2016, pp. 1–6.

[13] O. Arias, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in internet of things and wearable devices," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 1, no. 2, pp. 99–109, 2015.

[14] L. O. Chua, "Memristor-the missing circuit element," *IEEE Trans. on Circuit Theory*, vol. 18, no. 5, pp. 507–519, September 1971.

[15] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.

[16] P. Clarke, "Intel, Micron launch "bulk-switching" ReRAM," *EETimes*, July 2015.

[17] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proc. IEEE*, vol. 102, pp. 1126–1141, May 2014.

[18] M. Uddin, M. B. Majumder, and G. S. Rose, "Robustness analysis of a memristive crossbar puf against modeling attacks," *IEEE Transactions on Nanotechnology*, vol. PP, no. 99, pp. 1–1, March 2017.

[19] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas, "Design and implementation of the AEGIS single-chip secure processor using physical random functions," in *Proc. of the 32nd Annual Int. Symp. on Comput. Architecture*, 2005, pp. 25–36.

[20] G. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *44th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2007, pp. 9–14.

[21] D. Samyde, S. Skorobogatov, R. Anderson, and J. J. Quisquater, "On a new way to read data from memory," in *Security in Storage Workshop, Proc. First Int. IEEE*, December 2002, pp. 65–69.

[22] G. Moore, "Moores law," *Electronics Magazine*, vol. 38, no. 8, p. 114, 1965.

[23] U. Ruhrmair, J. Solter, F. Sehnke, and X. Xu, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. on Inform. Forensics and Security*, vol. 8, pp. 1876–1891, August 2013.

[24] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm arbiter PUFs: Accurate modeling poses strict bounds on usability," in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2012, pp. 37–42.

[25] W. Che, J. Plusquellic, and S. Bhunia, "A non-volatile memory based physically unclonable function without helper data," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, 2014, pp. 148–153.

[26] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, July 2013, pp. 156–160.

[27] H. Manem and G. S. Rose, "A read-monitored write circuit for 1T1M memristor memories," in *Procs. of IEEE Int. Symp. on Circuits and Syst.*, May 2011, pp. 2938–2941.

[28] C. Estebanez, J. C. Hernandez-Castro, A. Ribagorda, and P. Isasi, "Evolving hash functions by means of genetic programming," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006, pp. 1861–1862.

[29] M. Bellare and T. Kohno, "Hash function balance and its impact on birthday attacks," in *Advances in Cryptology EUROCRYPT 04, Lecture Notes in Computer Science*. Springer-Verlag, 2004, pp. 401–418.