INVITED PAPER

Public Physical Unclonable Functions

This paper surveys the time-bounded or public PUFs, including their design and security evaluation and the new protocols that they can support.

By Miodrag Potkonjak, *Member IEEE*, and Vishwa Goudar, *Student Member IEEE*

ABSTRACT | A physical unclonable function (PUF) is an integrated circuit (IC) that serves as a hardware security primitive due to its complexity and the unpredictability between its outputs and the applied inputs. PUFs have received a great deal of research interest and significant commercial activity. Public PUFs (PPUFs) address the crucial PUF limitation of being a secret-key technology. To some extent, the first generation of PPUFs are similar to SIMulation Possible, but Laborious (SIMPL) systems and one-time hardware pads, and employ the time gap between direct execution and simulation. The second PPUF generation employs both process variation and device aging which results in matched devices that are excessively difficult to replicate. The third generation leaves the analog domain and employs reconfigurability and device aging to produce digital PPUFs. We survey representative PPUF architectures, related public protocols and trusted information flows, and related testing issues. We conclude by identifying the most important, challenging, and open PPUF-related problems.

KEYWORDS | Cryptographic protocols; physical unclonable function (PUF); public PUF (PPUF)

I. INTRODUCTION

The initial impetus for modern cryptography was provided by the first public-key primitives and protocols in 1976. Using mathematical and algorithmic mechanisms, several primitives (building blocks) and numerous protocols that target a variety of classic cryptographic tasks have been proposed, implemented, and widely used. The initial emphasis was on secure communication between computers that are placed in physically secured locations.

Classic cryptographic mechanisms and protocols are among the most surprising and elegant algorithms within the wide spectrum of computer science tasks. Although their mathematical correctness is still not proved, it is widely considered that they are secure. However, it has also been demonstrated that classic cryptographic systems are easily compromised using side-channel techniques and physical attacks.

More recently, a new type of security primitive, the physical unclonable functions (PUFs), has attracted a great deal of attention. A PUF is a multiple-input–multiple-output function that has hard-to-predict dependency between the outputs and the inputs. While the initial proposal used an optical mesoscopic system for demonstration, the tremendous growth in interest in PUFs is due to its standard semiconductor integrated circuit (IC) implementation. The uniqueness of identical PUF design is provided by currently ubiquitous process variation. Several PUF architectures (e.g., arbiter based, ring oscillator, and SRAM) have been proposed, implemented, and analyzed. The initial security protocol was secret key in which one party collects a set of challenge–response pairs before releasing the PUF to another party. The authentication of the second party can now be done by the first party by issuing a challenge. Only the entity with the PUF can respond to an unknown challenge fast.

A PUF is a low-cost, small-area, and power-efficient security primitive that has good resiliency against side-channel and physical attacks. It has also been demonstrated that PUFs can be used for the creation of a variety of security protocols. However, PUFs remain a secret-key primitive which is an essential constraint for their use in many applications.

The introduction of public physical unclonable functions (PPUFs) was motivated by a need to create an ultrafast, ultralow-power public-key security primitive. The definition of a PPUF is a multiple-input–multiple-output system that is much faster to execute than it is to simulate, and whose security no longer relies on the secrecy of its physical parameters as PUFs do. Instead, PPUFs derive their security from large, preferably exponential, discrepancies between the time required for

Manuscript received August 26, 2013; revised June 9, 2014; accepted June 14, 2014. Date of publication July 15, 2014; date of of current version July 18, 2014. This work was supported in part by the National Science Foundation (NSF) under Awards CNS-0958369, CNS-1059435, and CCF-0926127.
The authors are with the Computer Science Department, University of California Los Angeles, Los Angeles, CA 90095 USA (e-mail: miodrag@cs.ucla.edu).

Digital Object Identifier: 10.1109/JPROC.2014.2331553

0018-9219 © 2014 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/ redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

response calculation from the private key and the time required to simulate the correct response from the public key. This enables PPUFs to underlie several public-key protocols and provide all the novel facilities that this entails, including easier key management and distribution, document signing, data exchange with untrusted parties, etc. Further, PPUFs can outperform classical public-key security primitives in terms of delay, energy, and throughput requirements owing to simpler and computationally less expensive design, and improve upon their resilience to side-channel and physical attacks [1].

Our goal is not just to present the use of PPUFs in a wide spectrum of security protocols and applications ranging from cryptography to trust and privacy, but also to describe their use in other applications such as intellectual property protection of software and hardware, synthesis of secure architectures and systems, and other emerging applications. We emphasize three dimensions: security primitives, protocols, and underlining security and design paradigms.

II. PPUF ARCHITECTURES

A. Public Physical Unclonable Function

The first PPUF design used a simple structure of XOR gates which had an expected exponential amount of glitching relative to the depth of circuitry [2]. Fig. 1 illustrates PPUF operation with an example of a simple PPUF circuit and the variation in gate delay characteristics caused by manufacturing processes variation. Assuming a stable circuit output after the input vector "01" is presented, introducing a new input vector "10" at time $t=0$ produces transient behavior: At $t=8.8$ ps ($t=9.3$ ps) the output at gate B (A) switches to "0", and then to "1" at $t = 11.2$ ps ($t = 10.1$ ps). Similar transient transitions occur at each row of the circuit, with number of such transitions doubling at each row and ultimately producing an exponential amount of glitching at the circuit's output relative to its depth.

More generally, a PPUF was constructed as a rectangular array of gates, w gates wide and h gates deep. If the expected number of output transitions per input transition for each gate is B, the simulation cost of the PPUF was shown to be $wB^h/2$. Therefore, the PPUF structure exploits the gap between implementation and simulation, which is an exponential function of the logic depth of the circuitry.

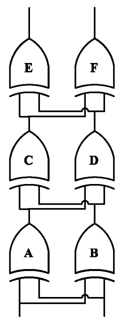
It is important to mention that there is another security primitive, SIMulation Possible, but Laborious (SIMPL), which also exploits the execution–simulation gap (ESG) [3]. Initially, the application range of SIMPL was restricted to authentication and required a relatively small ESG. However, Rührmair *et al.* have developed a variety of security protocols that exploit exponential ESGs [4]–[6]. Very recently, Horstmeyer *et al.* proposed hardware one-time pads that can serve as communication PUFs (cPUFs) [7]. Keys are extracted by optically probing random and unique optical structures using specially designed lasers. The keys are then shared using a theoretically perfect security procedure.

These first-generation PPUFs have several potential drawbacks including the need for ultra-accurate time measurements and detection of ultrashort glitches. While time can be measured with attosecond accuracy, the required instruments are very expensive. Furthermore, in order to achieve an acceptable advantage over attackers, both sides require computational efforts that are at least several seconds. Therefore, they are slower than the classical cryptography public-key cryptography protocols. These drawbacks were eliminated in more recent PPUF designs.

B. Differential PPUF

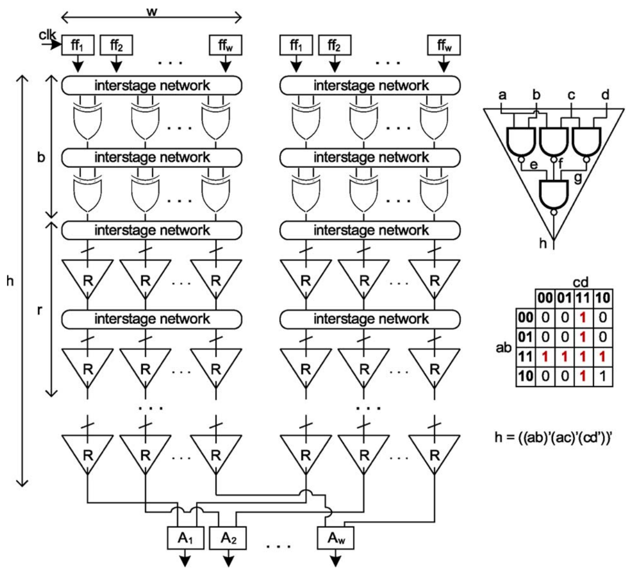
The need for ultra-accurate timing and for detecting glitching was eliminated using a differential PPUF (dPPUF) [8]. Here, an input vector is simultaneously presented to two nominally equal circuits whose delay characteristics differ due to process variation. As the signals propagate through each circuit, they race against one another toward an arbiter which locks its output value to that of the first signal that arrives. Ultimately, it is the differential timing between the frontier signals of the two circuits that defines dPPUF operation rather than the absolute timing of a transient output vector in a PPUF circuit, and this eliminated the need for ultra-accurate timing.

The dPPUF design was presented in the context of an authentication protocol, wherein a verifier issues a challenge input vector to the user. The user, who is in physical possession of the dPPUF, executes the challenge to produce the response within a single clock cycle and presents this response to the verifier. To authenticate the user, the verifier compares this to a simulated dPPUF response to the challenge. Similar to a basic PPUF, the dPPUF-based protocol takes advantage of the ESG between the time it would take an attacker to simulate the response, and the time the user takes to execute the challenge on a dPPUF IC.



Gate	Input 1 Delay (ps)	Input 2 Delay (ps)
A	9.3	10.1
B	11.2	8.8
C	11.1	9.0
D	7.8	7.1
E	8.6	9.5
F	12.4	9.6

Fig. 1. *Simple PPUF [2].*



	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	1

Fig. 2. ***dPPUF architecture of height h and width w [8]. A challenge vector propagates through repeated stages of b booster (XOR) and r represser (R) cells (right), with partly maximally interacting and partly random interstage networks. Signals from two nominally identical but physically unique PPUFs race to arbiters (A_i) to produce the response.***

Assuming p is the probability of the more likely output value (0 or 1) on each output line of a dPPUF, and q is the probability that a randomly chosen response will be successfully authenticated, the size of the output vector n required to achieve q is $log(q)/log(p)$. Therefore, to maximize the proof of strength (q) on a dPPUF of given size, it is necessary to keep p as close to 0.5 as possible. In other words, output predictability dictates dPPUF's security.

Fig. 2 illustrates the dPPUF architecture proposed by Potkonjak *et al.* It uses the concepts of confusion and diffusion to reduce predictability. It is composed of alternating layers of booster cells that amplify the extent of output switching, and represser cells that unpredictably repress frontier signal transitions which would otherwise lock the arbiters. These concepts of repression and boosting are crucial to the creation of frontier signals that are highly unpredictable. Hence, when using a dPPUF, it is sufficient to detect a winner in a race between two signals using an arbiter.

Potkonjak *et al.* also studied the output predictability of the dPPUF architecture by measuring the output probabilities for each of its output bits over many input vectors [Fig. 3(a)]. The red dashed lines depict the ideal case where $P(O_i = 1) = 0.5$ for all i. Similarly, the predictability for a single input vector across 1000 different dPPUF instances was also simulated [Fig. 3(b)]. The study concluded that the few suboptimal outputs should be excluded from verification in order to maximize dPPUF security. Finally, Fig. 3(c) illustrates the extent of correlation between output bits. If an attacker can successfully predict multiple outputs by simulating only a few, this affords him a proportional advantage in simulation time. Low correlations were observed between a large majority of output bit pairs.

Although dPPUFs eliminate many of the problems of the first-generation PPUF, they still require that at least one of the communicating parties invest significant resources and time in simulation.

C. Matched PPUF

Matched PPUFs (mPPUFs) [9] are the first security primitive that enable the execution of a number of cryptographic and other protocols in a single clock cycle. The idea is simple and represents a complete departure from ESG: two identical PPUFs are created in such a way that creation of a third identical PPUF is essentially impossible. mPPUFs are created using both process

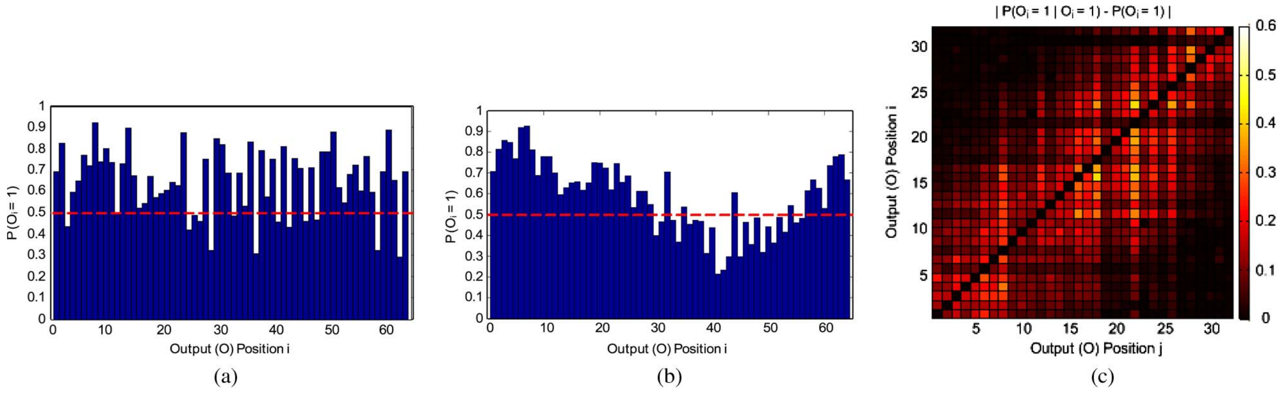


Fig. 3. ***For w = 64 and h = 10, probability that an output bit will equal to 1 (a) for one dPPUF and (b) for 1000 different dPPUF instances given the same input. (c) Correlation between output bits O_i and other output bits O_j [8].***

variation and device aging. Essentially, each party ages a large number of gates in their PPUFs in order to create identical paths through the circuit. Parts that cannot be matched are removed from the PPUFs using either power gating or software mechanisms. Since the PPUF delay characteristics are public, no storage or sharing of secret keys is necessary and the diverse applicability of public-key security may be availed. Further, by obviating the need for simulation, this security is achieved in a single-cycle, low-energy operation.

In [9], Meguerdichian and Potkonjak leveraged a controlled and reversible device aging technique (e.g., negative-bias temperature instability (NBTI)-induced transistor slowdown [10]) to age a subset of the gates of each communicating party and disable the unmatchable gates, thus producing perfectly matched PPUFs. Under the assumption that gate delays are randomly distributed between 0 and 1, and that device aging can increase a gate's delay by up to 0.5, Meguerdichian and Potkonjak showed that 25% of gates would not be matchable and an attacker making use of the publicly available delay characteristics to match his PPUF to that of the communicating parties would succeed in doing so for only 58% of his matched gates. On the other hand, the infeasibility of simulation-based attacks is apparent.

Fig. 4 illustrates the mPPUF architecture which is similar to the dPPUF architecture (Fig. 2). It is also similar to the dPPUF in that its output vector depends on the frontier signal which, in the case of an mPPUF, races against the clock. This architectural detail has been borrowed to enable a continued independence from ultra-accurate timing requirements. The mPPUF architecture also uses boosters and repressers toward minimizing predictability. While boosters increase switching frequency, the role of repressers is to decrease the switching frequency in an unpredictable manner in order to repress frontier signal transitions and thereby increase simulation complexity and unpredictability. For a represser such as a k-input NAND gate, the unpredictability of its output depends on the probability distribution of its input. As it turns out, the effect of multiple stages of identical repressers is a decrease in the randomness of input between consecutive represser stages, in a manner that increases switching frequency in a more predictable manner rather than decreasing it in an unpredictable one. It was shown that alternating among different classes of repressers between stages mitigates this effect [9]. Consequently. the application of repressers was refined to the use of different classes of repressers at different stages. The final stage of the mPPUF architecture is a terminator cell, such as a k-input OR gate, that increases the stability of the inputs to the arbiters.

D. Quantized PPUF

Quantized PPUFs (qPPUFs) eliminated the last mPPUF bottleneck—the need that a user uses a different PPUF in communication with different parties [11]. Since mPPUFs require the delays of corresponding PPUF gates to be closely matched, the number of matchable gates reduces exponentially as the number of communicating parties increases. qPPUFs inherit their architectures from mPPUFs (Fig. 4). However, they differ in that each user's PPUF is aged independently of all other users. Matching then boils down to identifying and disabling those gates corresponding to the pair of communicating PPUFs which cannot be matched.

In order to accomplish matching with k different parties, a user disables k different sets of components on his PPUF. Note that he separately communicates with each of the other users, enabling and disabling the corresponding set of PPUF components in turn. Since aging is accomplished independently for each PPUF, switching communication to a different user in this manner can be accomplished in real time. Consequently, this new mPPUF enables a user to use a single PPUF to communicate with an arbitrary number of parties in which each of them requires only a single cycle computation.

qPPUFs achieve independent aging by quantizing the delay profile of each component class to a limited set of

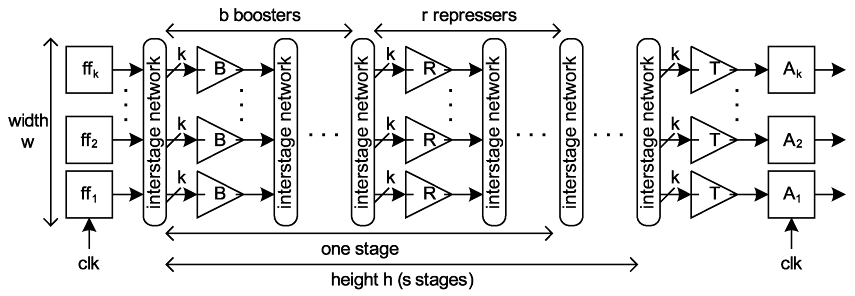


Fig. 4. ***mPPUF architecture of width w and height h [9], consisting of s stages of b booster cells (B) and r represser cells (R), with a final level of terminator cells (T) to enhance stability. Signals from input flip flops (ff) race against the clock to arbiters (A), which output 0 if the signal transitions before the clock and 1 otherwise.***

values. A user then ages each component to coincide with an achievable quantum, thereby eliminating the need for coordinated aging. If a component cannot be aged to any of the preselected quanta, it is disabled. And if a component can be aged to multiple quanta, one is chosen randomly. Fig. 5 depicts such a quantization for a sample PPUF component with a Gaussian delay profile (mean $= 1$, standard deviation $= 0.1$).

To match their PPUFs, one user from each user pair transmits its delay quanta assignments for each component. This is followed by a reply message identifying the components that could not be matched and must be disabled for that user pair. An n-to-n public-key communication can follow, wherein each of n users in a group achieves secure pairwise public-key communication with other members of the group, using a single PPUF while incurring little overhead when switching between communication partners, akin to the one described in [11].

A major drawback of quantization was identified to be a drastic reduction in matching probability. For the delay distribution in Fig. 5 and with two quanta, Meguerdichian and Potkonjak showed that the matching probability is about 0.35. This implies a similar reduction in the effective size and security of this qPPUF. While one solution is to increase the size of the PPUF, a more effective alternative was shown to be one where each PPUF component is replicated to increase its probability of getting matched.

Fig. 6 shows the change in matching probability as the number of replicas and the number of quanta are varied under the same delay distribution as in Fig. 5. Increasing the number of quanta improves the matching probability up to a point, after which the quantization intervals begin to overlap, thereby reducing the matching probability. This reduction arises from the randomized assignment of a delay quantum to a component when it can satisfy multiple quanta. In contrast, increasing the number of component replicas monotonically improves matching probability.

Finally, mathematical and statistical models of gate delay, threshold voltage, effective channel length, and aging were applied to study the behavior of qPPUFs [12]–[15]. Via simulation, Meguerdichian *et al.* also showed that the output predictability of a qPPUF is near-optimal for a vast majority of its output bits ($w = 128$, $b = 2$, $r = 1$, $s = 7$, eight replicas, two quanta). Further studies showed that the probability that an attacker could match a single component of his PPUF to already matched components of two users was greater than 0.8 but less than 1. However, it was argued that with just 2000 PPUF components, the probability that an attacker fully matches the qPPUF was on the order of 10^{-92}, despite a probability of matching a single PPUF component equal to 0.9.

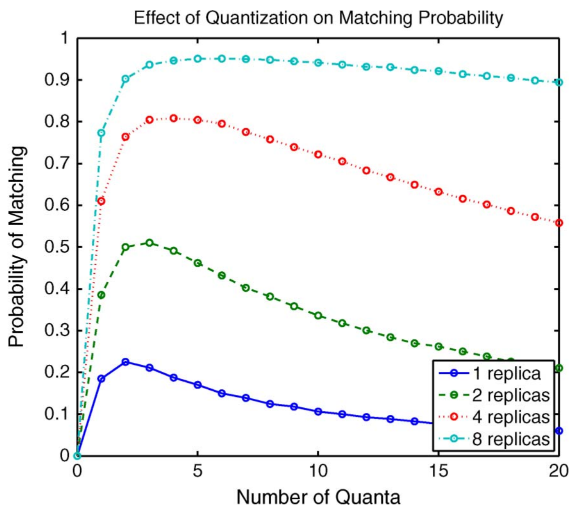


Fig. 6. ***Effect of varying quantization and component replication on matching probability [11].***

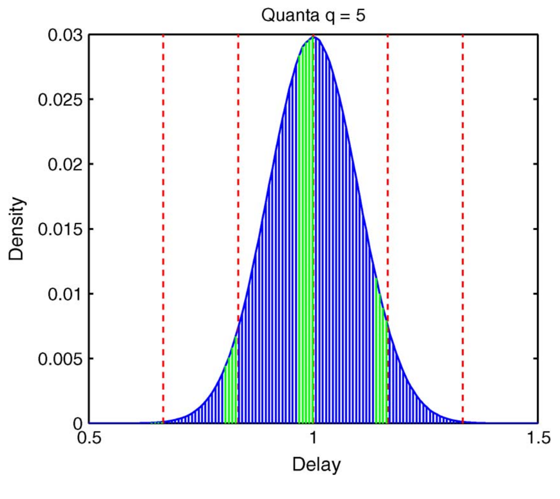


Fig. 5. ***qPPUF quantization example, showing possible quanta (red dashed) and gate delays that can be aged to match some quantum (shaded) for five quanta. Those gates with delay in the green shaded regions can age to two quanta (one is chosen at random) [11].***

E. Digital PPUF

Digital PPUFs aim to provide low-energy public-key security that is independent of operational and environmental conditions, as well as, natural device aging [16]–[19]. This was achieved by avoiding an analog mechanism as the basis of the underlying security primitive. A pair of randomly generated digital bimodal functions (DBFs) is used instead. One of these functions f_{compact} can be computed quickly while its companion f_{expand} is unacceptably computationally intensive. In other words, digital PPUFs reverted to the use of an ESG to gain a disproportionate computational advantage over an attacker.

The following equation

$$\text{Input: } a_i, i \in \{0, 1, 2 \ldots 11\} \tag{1a}$$

$$\text{Output: } c_j, j \in \{0, 1, 2, 3\} \tag{1b}$$

$$b_j, j \in \{0, 1, 2, 3\} \tag{1c}$$

$$\begin{cases} b_0 = a_0 a_1 a_2 + \bar{a}_1 a_3 \\ b_1 = a_4 \bar{a}_5 a_6 + \bar{a}_4 \bar{a}_7 + a_5 a_7 \\ b_2 = a_8 \bar{a}_9 + a_9 \bar{a}_{10} \bar{a}_{11} \\ b_3 = \bar{a}_0 \bar{a}_4 a_8 + a_0 \bar{a}_{10} \end{cases} \tag{1d}$$

$$\begin{cases} c_0 = b_0 b_1 b_2 + \bar{b}_1 b_3 \\ c_1 = b_1 \bar{b}_0 b_3 + \bar{b}_1 \bar{b}_3 + b_0 b_2 \\ c_2 = b_1 \bar{b}_3 + b_2 \bar{b}_0 \bar{b}_1 \\ c_3 = \bar{b}_0 \bar{b}_2 b_3 + b_1 \bar{b}_3 \end{cases} \tag{1e}$$

$$\begin{cases} c_0 = a_0 a_1 a_2 a_4 \bar{a}_5 a_6 a_8 \bar{a}_9 + a_0 a_1 a_2 a_5 a_7 a_8 \bar{a}_9 \\ \quad + a_0 \bar{a}_{10} a_4 \bar{a}_5 \bar{a}_6 + a_0 a_1 \bar{a}_{10} \bar{a}_{11} a_2 a_9 + a_0 \bar{a}_{10} a_4 a_5 \bar{a}_7 \\ \quad + a_0 a_1 a_2 \bar{a}_4 \bar{a}_7 a_8 \bar{a}_9 + a_0 \bar{a}_{10} \bar{a}_4 \bar{a}_5 a_7 + \bar{a}_0 \bar{a}_4 \bar{a}_5 a_7 a_8 \\ \quad + \bar{a}_1 a_3 a_4 \bar{a}_5 a_6 a_8 \bar{a}_9 + \bar{a}_1 a_3 a_5 a_7 a_8 \bar{a}_9 \\ \quad + \bar{a}_1 a_3 \bar{a}_4 \bar{a}_7 a_8 \bar{a}_9 + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 a_4 \bar{a}_5 a_6 a_9 \\ \quad + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 a_5 a_7 a_9 + \bar{a}_1 \bar{a}_{10} \bar{a}_{11} a_3 \bar{a}_4 \bar{a}_7 a_9 \\ c_1 = \cdots \\ c_2 = \cdots \\ c_3 = \cdots \end{cases} \tag{1f}$$

presents a sample digital PPUF where (1d) and (1e) comprise f_{compact}. f_{expand} is derived by substituting (1d) into (1e) and reducing to a minimal sum of products [see (1f)] or product of sums. Note that, in the general case, it is required that the number of inputs equals the number of output functions, i.e., the cardinality of i equals to that of j. Here, each set of subfunctions b_i is identical to the subfunctions c_i. In general, a digital PPUF can require several such iterations of the output subfunctions to arrive at the DBF f_{expand}. It is also required that the number of unique inputs to each output subfunction be bounded by some k to enable field-programmable gate array (FPGA) configuration and synthesis of the subfunctions in f_{compact}.

While f_{compact} and f_{expand} are equivalent, and f_{expand} can be derived from f_{compact} via iterative substitution and reduction, the inherent difficulty in functional decomposition makes it impossible to derive f_{compact} from f_{expand}. Further, Xu *et al.* [16] show that the number of product terms in f_{expand} exponentially increase with the number of primary inputs a_i and the number of iterations of the output subfunctions, whereas the number of terms in f_{compact} only increases linearly. Herein lies the ESG setting f_{compact} as the private key and f_{expand}, or some nontrivial subset thereof, as the public key makes it infeasible for an attacker armed with the public key to compute the DBF output. Xu *et al.* [16] discussed this infeasibility and showed that with 20 iterations and ten inputs, the simulation time of f_{expand} is close to seven orders of magnitude slower that the execution time of f_{compact}. Based on these parameters, it was also shown that defeating the public-key communication scheme described below will take an attacker about 283 years.

A digital PPUF can be distributed and enlisted for public-key communication from A to B in the following manner. B selects a sizeable subset of f_{expand} and transmits it to A. Then, A chooses to store a random and manageable subset of the set of terms transmitted by the B as B's public key. Let the number of terms in this public key be l. For each such term f_i, A randomly selects its sum-of-products or product of-sums representation. Further, it selects a random term in the selected representation. Next, it generates the input vector p_i that would make $f_i(p_i) = r_i = 1$ if a product term was selected, or $f_i(p_i) = r_i = 0$ if a sum term was selected. Note that it is sufficient to select p_i as the input that would set the selected random term from f_i to 0 or 1, depending on the functional representation chosen, and this can be accomplished in linear time. Then, A concatenates each p_i for $i \in \{1, \ldots, l\}$ to construct a vector P_1. Similarly, vector R_1 is generated from the r_i's. This process is repeated N times to generate N vectors P_j and R_j for $j \in \{1, \ldots, N\}$. N is a customizable parameter that further boosts the ESG and increases security while also increasing the energy costs and execution time.

Finally, A encrypts its message with the series of vectors R_j and broadcasts the encrypted message along with the vectors P_j. B, who is in possession of the private key f_{compact}, quickly computes vectors R_j from P_j and may then decrypt the message. An attacker overhearing the communication between A and B must, on average, compute half of the potentially exponentially number of terms transmitted by B to successfully decrypt the message, making this proposition infeasible.

Fig. 7 illustrates the sequential logic cluster (SLC) architecture that is designed to compute f_{compact} for a given input vector. It can also be used to derive the subset of f_{expand} that forms the basis of the public key. The input vector undergoes random shuffling before entering w k-input lookup tables (LUTs) that each applies the combinational logic for the corresponding output subfunction of f_{compact}. The output of each iteration is stored in flip-flops at the end of computation before being fed back into the input stream for the next iteration. With this architecture, a large number of unrepeated SLCs can be produced simply by altering the random shuffling or the contents of the SRAM cells in the LUTs.

The performance of digital PPUFs was analyzed in terms of the optimality of output prediction [Fig. 8(a)], and the extent to which the value of each output bit depends on that of the other output bits [Fig. 8(b)]. It was found that output prediction was near-optimal for all bits

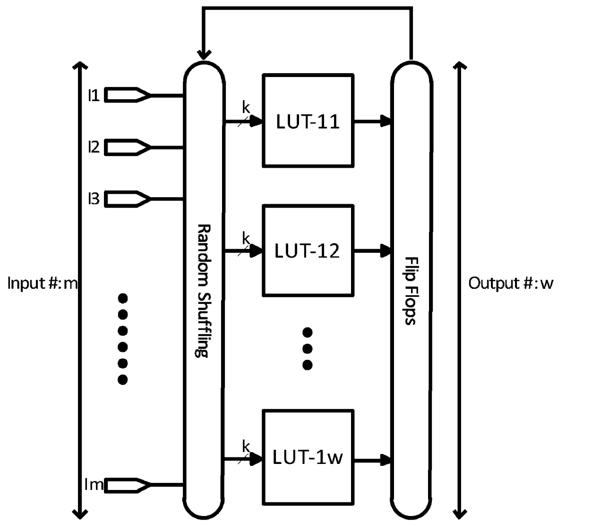


Fig. 7. ***Architecture of a sequential logic cluster [16].***

in the average case, and that a majority of output bit pairs were near-independent (conditional probability close to 0.5). It was also shown that the energy costs of the SLC implementation-based decryption is on par with that of hardware implementations of AES and three orders of magnitude better than that of RSA. A major drawback is the overhead involved in the repeated communication of the large subset terms from f_{expand} that form the basis of the public key. However, technological trends show communication speeds outperforming storage speeds [20], which translates to a widening gap between requirement of an attacker who must store this large subset of terms, and that of the communication parties who must merely share them.

F. Nano-PPUF

Nano-PPUFs provide a faster, low-energy and more secure alternative to device aging, to overcome the burden of simulating the entire PPUF at one of the communicating parties (e.g., basic and differential PPUFs) [21]. Nanotechnology components are inherently faster and use lower power than current complementary metal–oxide–semiconductor (CMOS) technologies. Further, they naturally exhibit nonlinear circuit characteristics, in addition to the randomness that is inherent to their synthesis. This yields an ESG that is larger than in circuits composed of linear components. Nanotechnology components also express the unique characteristic of bidirectional signal propagation: Since input signals can be applied at any end of the circuit, this exponentially increases the input and output spaces with respect to a comparable CMOS PPUF. Together, these characteristics make nano-PPUFs more secure than conventional PPUFs.

Fig. 9 illustrates an example nano-PPUF. It is composed of a grid of nano-PPUF cells, with adjacent cells connected over matching pins. Each cell is modeled as a geometric random network generated during the synthesis of III–V nanowires. Nodes within the network are uniformly distributed and connected based on threshold distances. Each node exhibits nonlinear current–voltage characteristics, which gives rise to the nonlinear circuit characteristics of nano-PPUF cells and grids. The bidirectional signal propagation property of these circuits allows for inputs to be applied at any of the pins at the boundary of each cell. Therefore, nano-PPUF inputs are defined as a combination of input values (the voltages applied at the inputs) and input sets (the set of pins at which the voltages are applied). The rest of the boundary pins define nano-PPUF outputs, and are similarly composed of output values and output sets.

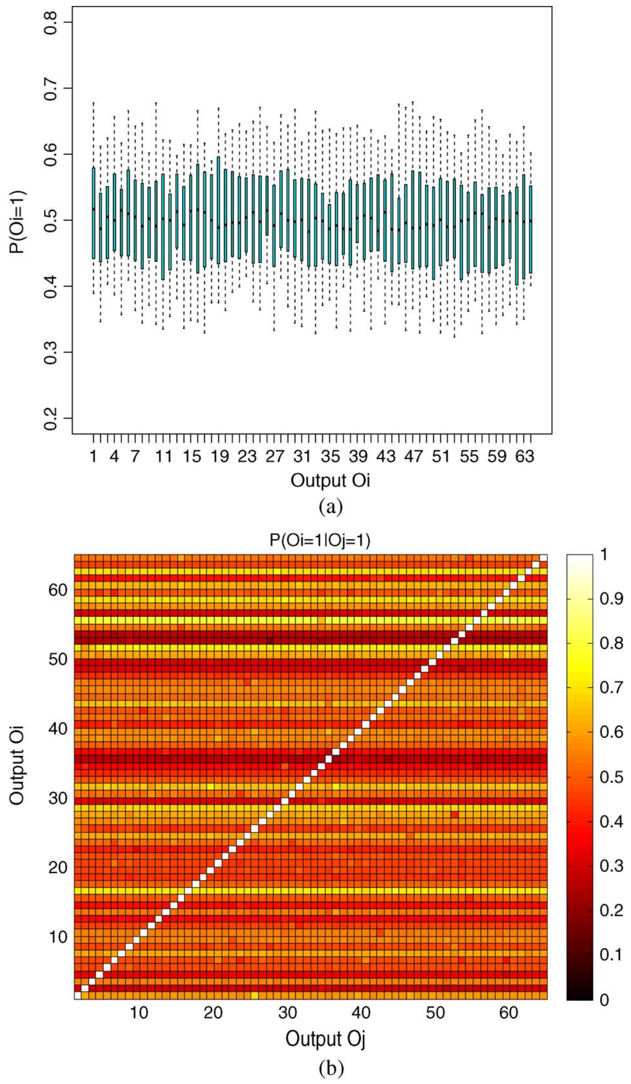


Fig. 8. ***For 64 primary inputs, 32 iterations, and 1 000 000 input vectors: (a) probability that an output bit is equal to 1, and (b) conditional probabilities between output bits O_i and other output bits O_j [16].***

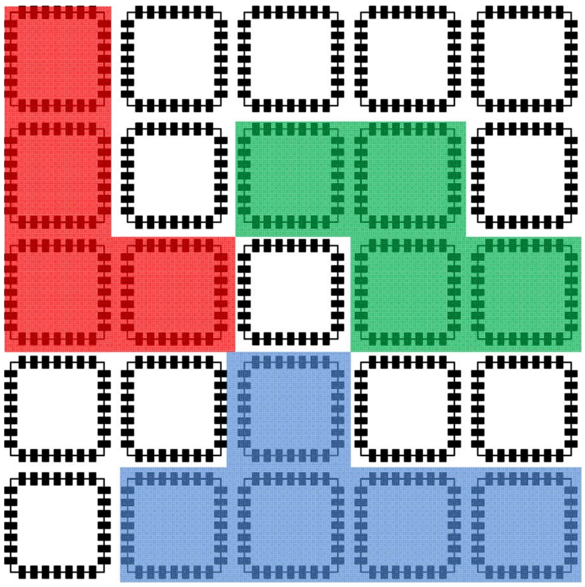


Fig. 9. *Example of a 5 × 5 nano-PPUF grid. Adjacent nano-PPUF cells are connected via adjacent pins. Example polyomino partitions of size 4, 4, and 5 are shaded [21].*

To enable runtime definition of input and output sets, each pin at the boundary of a cell is multiplexed to facilitate input or output. This is what allows for arbitrary definitions of inputs and results in the exponential growth of the input/output spaces. Specifically, Wendt and Potkonjak [21] define a polyomino partitioning strategy where the set of cells accepting inputs are adjacent and connected, and formed by joining one or more equally sized squares edge to edge (see Fig. 9). It is shown that this strategy ensures an input space that is exponential in the number of cells.

Two protocols are described as applications for nano-PPUFs: authentication and public-key communication. Authentication works as follows: The verifier issues a challenge to the user as an input to the user's nano-PPUF (input set and values). The user executes the challenge on his PPUF and presents the response to the verifier. The verifier simulates one or a few partitions of the user's nano-PPUF to verify that the input and outputs along their boundaries converge to the response provided. An attacker cannot know which partitions the verifier may simulate and must simulate the response over the entire nano-PPUF circuit, which is an infeasible task if the circuit is large enough.

Public-key communication is achieved as follows: The sender simulates a challenge and response on a partition of the receiver's nano-PPUF, and encrypts his message with the challenge. He transmits the encrypted message along with the response and the input set of the challenge to the receiver. The receiver searches for the input values corresponding to the challenge until the response values are matched. Once the entire challenge is recovered, the message is decrypted. On average, an attacker must simulate a search over half the set of all input values for the given input set, which is exponential in the size of the input set, making the attack infeasible.

Wendt and Potkonjak assessed the security of nano-PPUFs via SPICE circuit simulations. The study showed low correlation between input values and outputs. It was observed that even if such a correlation was discovered, given that the number of possible input sets is exponential in the size of the circuit, this would make it difficult to compute and store all sets of relationships for future lookup. In contrast, simulations showed strong correlations between a small set of output pairs (Fig. 10).

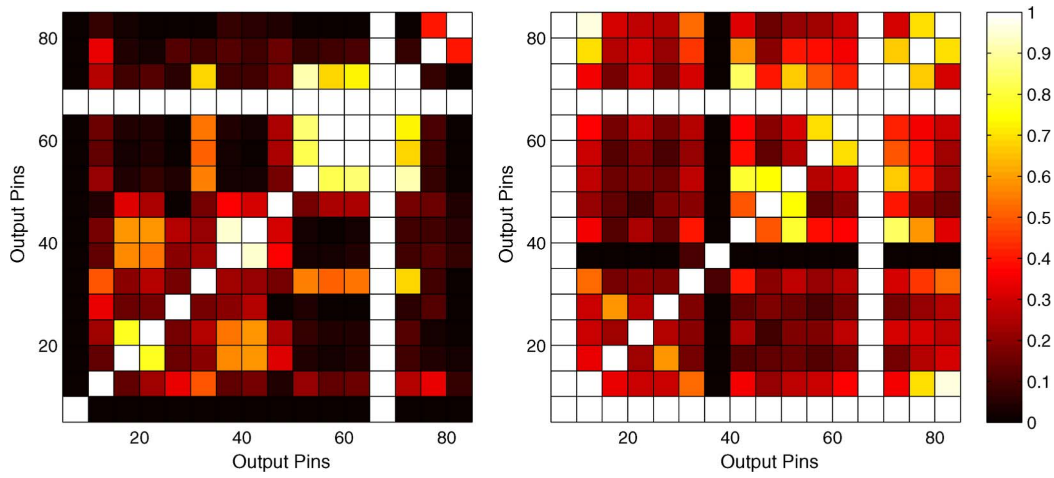


Fig. 10. *R^2 correlations between a subset of nano-PPUF output pins. Both figures depict the same nano-PPUF with different input sets. Pin 65 has perfect correlation because it returns a constant value due to process variation at network synthesis [21].*

However, the correlated pairs changed when the input set changed. Consequently, Wendt and Potkonjak again posited that it would be infeasible to compute and store an exponential number of such correlations for each input set. Finally, simulation attacks were observed to be infeasible due to the exponential relationship between the circuit size and simulation time.

III. PPUF PROTOCOLS

In this section, we sample from a wide spectrum of security, privacy, and trust protocols that employ PPUFs as a hardware primitive. Our emphasis is more on demonstrating new paradigms and strategies that facilitate the creation of protocols than on providing comprehensive coverage. The most important observation is that conceptually entirely new paradigms with respect to classical and quantum cryptography are required and created. What is common to all of them is that they are often surprisingly simple, and that all of them are intrinsically resilient against physical and side-channel attacks. For each of the representative protocols, we present their procedures when gap-based or matched PPUFs are used. Here, the most important observation is that matched PPUF-based protocols are invariably ultrafast and energy efficient: they rarely require more clock cycles and their PPUFs often have only a few hundred gates.

A. Public-Key Cryptography

First, we consider gap-based public-key communication protocols and follow that with a few modifications mainly related to the use of public-key infrastructure protocols presented by Beckmann and Potkonjak [2]. As with classical cryptography procedures, all PPUF-based public-key cryptography protocols assume the existence of a third trusted-party public-key infrastructure to prevent false impersonation attacks where the attacker pretends that he is one of the legitimate parties. Let us consider the situation where party A wants to send a secret message to party B using a gap-based scheme. Both parties A and B are in possession of their PPUFs whose public characteristics have been deposited with the trusted party T. Recall that for a given input I anybody can use simulation and calculate the correct corresponding output O using long simulations that, at the very least, require times in range of seconds. At the same time, the execution of A's or B's PPUF can be accomplished in the nanosecond range or faster.

The key challenge is to now create protocols that leverage this gap between the execution and simulation times, say of a factor of 10^{12}, into two advantages for each of participating sides over the potential attacker. The advantage of side A that has to decrypt the message from B is obvious and provided directly by the ESG. The advantage of side B arises in the following manner. B selects a large set of numbers S that has on the order of 10^{12} elements, and sends it to A. Set S has a compact representation. For example, it may be 10^{12} numbers starting from a certain specified number. B now randomly selects an element from S and uses it as input I in a simulation to calculate the corresponding output O of A's PPUF. Finally, it uses bitwise XOR of the input I with the actual message AM to create the encrypted message EM and sends it to A together with the output O.

Now A uses its PUF to find the input that corresponds to output O, trying all inputs from set S. Once that input is identified, all that is required is for it to XOR input I and encrypted message EM to obtain the actual message AM. A can do it in a few seconds because it has its fast PPUF. Recall that A has a 10^{12} factor speed advantage over the attacker.

B also has similar advantage because it has to compute only single challenge–response pair as it is the one that selects the input I. We see that, by selecting the size of set S, A, and B can optimize their relative execution times. There are other degrees of freedom that can be used for boosting this advantage over the attacker. For example, each of the legitimate parties may have multiple PPUFs where message sender randomly selects which one is used. Modern ICs have billions of gates, and each PPUF requires no more than a thousand gates.

Note that the presented protocol ensures that the attacker cannot decrypt the encrypted message, but, in this form, it allows impersonation attacks. However, a very small alteration is sufficient to prevent this attack. For instance, A can send a message M to B that is encrypted using B's PPUF and is used for additional XOR in the protocol presented above. This message can only be decrypted using B's PPUF in reasonable time. Finally, we conclude that it may be inferred from the ESG of the presented protocol that the expected time to crack the encryption is more than 100 000 years, and may be further boosted in several ways.

We now consider public-key cryptography for matched PPUFs. For the sake of brevity, we discuss quantized PPUFs. Again, we assume that there is a third trusted-party public-key infrastructure where all parties deposit the delays of each gate in the standard PPUF structure, after all the gates have been aged to the standardized quantization level. If two parties want to communicate with each other, they disable all unmatched gates in their PPUFs and begin their communication. The expected cardinality of the two matched PPUFs is well beyond the feasibility of simulation in any reasonable time on any reasonably sized distributed computing platform.

Now, there are several ways how public-key communication can be organized. In the simplest instance, all that is required is for party B to select an input I and calculate the corresponding output using the PPUF configuration that is matched to that of A. The final step is XORing of the output O and actual message AM into the transmitted message SM that is sent to party A together with input I. Party A also required only two steps. The first is the calculation of output O and the second is XORing of the output O and SM which yields the actual message AM. Note that, in this protocol, the exchange of only a single message which is composed of input I and the encrypted message SM is sufficient. The only required computations consist of one PPUF calculation and a single bitwise XORing. Hence, this is the fastest and lowest energy realization of a public-key protocol.

B. Secure Location Authentication

Secure location authentication enables verifier V to provably establish the claimed location of another participating party A [22], [23]. Recall that the verifier can deploy n measurement devices and ask A to send signals to each of them. In 3-D space, it is sufficient to use four measurement devices. The problem has not been solved in the domain of classical cryptography. A scheme where V asks party A to send its digital signature for a specified challenge along with the global positioning system (GPS) signal is not effective because A can provide its secret key to another helping party that serves as its proxy. Interestingly, the secure location problem can be solved using quantum cryptography.

The problem is also easily solved using any of the proposed PPUF designs. And at least in the case of matched PPUFs, its energy expense and time requirements are minimal and correspond to a few computational cycles if only digital components are considered. The idea is to slightly modify the procedure for remote trusted sensing and/or trusted computing. Note that, in these protocols, the verifier's PPUF can load any data to its PPUF, while A can receive its GPS data only through an antenna.

The essential observation is that A's PPUF cannot be replicated and, therefore, only A can correctly respond to V's challenges. A cannot remotely hap its proxy because this would introduce additional communication delays. In addition, A would have to produce PUF results for challenges and the GPS signals, and accurate results are possible only if A is actually at the claimed location or if its PPUF can use data from memory. The latter case would require hardware changes that would invalidate the delays of A's PPUF.

C. Matched PPUF-Based *k*-Anonymity Security Protocol

Privacy has emerged as one of the dominant system constraints and essential application desiderata for numerous important domains. For example, it is particularly important to mobile users (e.g., cybercar applications that provide complete information about locations of cars and their trajectories), and in the financial, medical, and social network domains which may provide sensitive information about users. There have been a number of attempts to solve this problem, for instance, zero knowledge techniques have been proposed to preserve user privacy of authentication at toll booths. While these techniques appear algorithmically sound, their computational requirements are excessive: even the most efficient implementations require days for execution on modern personal computers (PCs). Gap PPUF-based solutions are somewhat faster but still require computational efforts with runtimes in the range of seconds. However, quantized matched PPUFs produce protocols that require only a few clock cycles, i.e., nanoseconds or so.

The essence of the approach is as follows. Let us assume that a toll owner (verifier) has m drivers (users), each equipped with a PPUF with uniform architecture. The PPUFs follow the quantized matched PPUF paradigm. Therefore, the verifier can configure her PPUF in such a way that it has identical input/output relationship with any subset of users. The larger the required overlap of matched gates, the smaller the cardinality of this subset will be. At the extreme, if the necessary overlay is very high, only a single user matches the service provider.

The basic k-anonymity protocol starts with a step where the verifier partitions all users into n groups so that each group has $k1$ drivers. For each group, she finds all gates that are shared among all $k1$ drivers in that group. At the first interaction step, she sends the characteristics of each of the n quantized matched PPUFs to the corresponding drivers, together with a randomly generated challenge.

If a driver is able to calculate the correct response to the challenge corresponding to one of the quantized matched PPUF characteristics, it is a potential sign that he indeed is one of paying subscribers. The chances that an attacker can compute the correct answer can be calculated using the following reasoning. If we denote the probability of matching two corresponding gates by p_g and the number of gates in each PPUF by n_g, then the probability p_{a1} that the attacker matches with a particular group is equal to p_g n_g. Hence, the attacker's chance to match with any one of the n groups is equal to $1-(1-p_{a1})^n$ which, for realistic numbers, is a very small quantity.

This probability can be reduced using the following procedure. The verifier additionally partitions users in the group for which she received the correct responses into two groups so that each group contains at least $k2$ drivers, where $k2$ is smaller than $k1$. These new groups are established in such a way that the number of matched PPUF gates in each new partition is maximized. If the user with the correct response is legitimate, he can still easily calculate the response to verifier's challenge for the new matching. However, the attacker's probability is additionally reduced. Further, this procedure can be iteratively repeated until the probability of a successful attack is satisfactory. The key idea is that the number of matched gates in each group is maximally increased which allows for the reduction in anonymity to be favorably traded against the chances of a successful attack.

D. Trusted Sensing and Computing

The quantitative advantages of PPUF-based cryptographic protocols in terms of metrics, such as time of execution, footprint, and energy, are important. They are

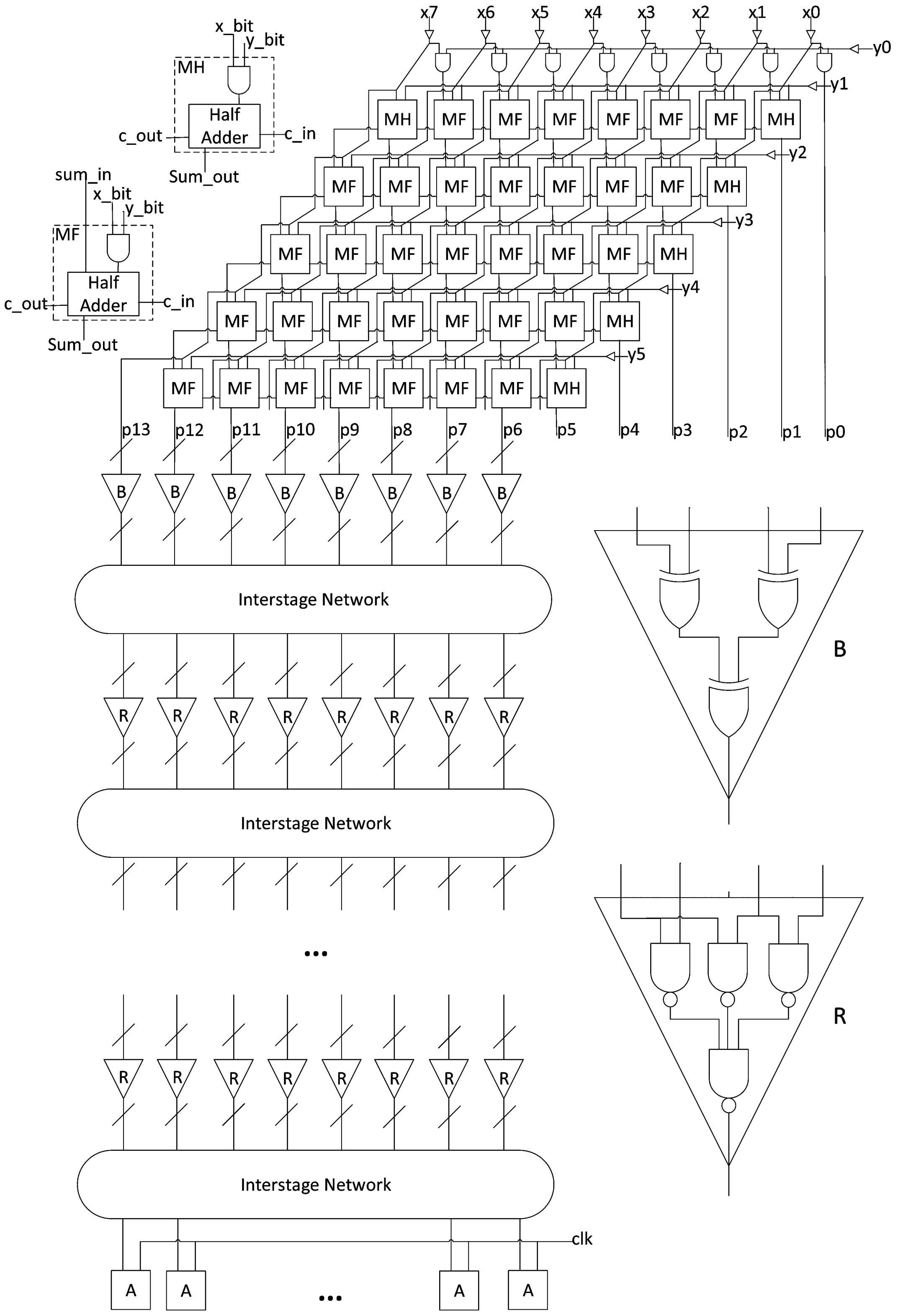


Fig. 11. ***Quantized matched PPUF integrated with multiplier circuit.***

often three or more orders of magnitude more efficient in terms of energy. These advantages are further exaggerated by their resiliency against physical and side-channel attacks. However, these protocols will be serious alternatives to classical cryptography only if and when the security of these protocols is adequately tested, i.e., when numerous researchers launch numerous attacks.

From an application point of view and in a qualitative sense, the benefits of PPUFs are more apparent and practical. These benefits are particularly pronounced for quantized matched PPUFs. Specifically, there are two major advantages of matched PUFs using standardized quantization over other PPUF schemes. One is that it is the first security primitive (building block) that enables execution of public-key protocols in a single clock cycle while providing resiliency against physical and side-channel attacks. The other is that it enables integration with standard logic is such a way that the secure flow of information is guaranteed. This property can be used for the creation of a spectrum of new security, privacy, and trust systems, protocols, and applications. In the rest of this section, we focus on techniques that enable the creation of localized and distributed secure systems that employ quantized matched PPUFs. It is important to emphasize that these techniques can be applied to other types of PPUFs. However, these systems are significantly less efficient, except in the case of digital PPUFs.

Fig. 11 shows one way how we can integrate a standardized quantized PPUF with a multiplier. A part of the multiplier is used as part of the aged PPUF. The integration is realized in such a way that a subset of gates of the multiplier is simultaneously used for the computation of multiplications, and as part of the security primitive PPUF. As in Fig. 11, this may be achieved by treating the output of this subset as input to a quantized matched PPUF—multiple stages of booster (B) and repressor (R) cells that race against the clock to the arbiters (A). Therefore, if such a multiplier is used for computation, it automatically also participates in producing the PPUF output. The PPUF output, therefore, depends on the computation that is conducted by the multiplier and provides proof that a particular computation was indeed performed by the multiplier. Of course, an arbitrary arithmetic or logic unit may be used instead of the multiplier.

Similarly, instead of the computational or logic modules of processor, memory, or application-specific ICs, we can overlap PPUF circuitry with the circuitry of a sensor or a radio receiver to fulfill trusted sensing [24], [25] and communication requirements. Yet another option is to integrate an aged matched PPUF with clock circuitry. The key observation is that the attacker cannot alter any part of the system that is integrated with a matched PPUF without also altering the PPUF's properties. This is so because the delay characteristics of the pertinent PPUF are impacted by the driving loads of the integration gates.

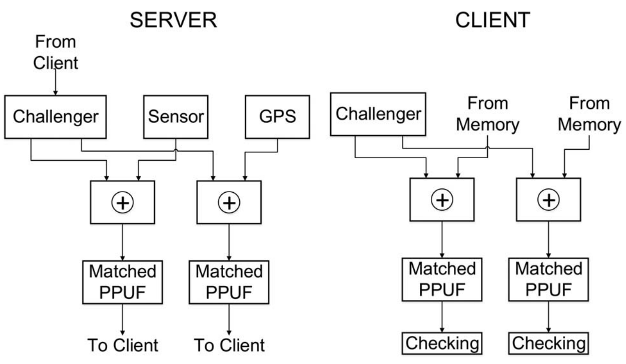


Fig. 12. ***Trusted sensing system and flow.***

Using integrated matched PPUFs, one can create a variety of security, privacy, and trust protocols. We now discuss two such protocols. The first is for remote trusted sensing, where the goal is to design a distributed system that consist of two devices where the first receives trusted data from the second. The data are trusted in the sense that the first device can check that the received data are from the second device, and that it is collected at a specific time and a specific location. This is accomplished in the following way.

The distributed hardware platform is shown in Fig. 12. Both devices client (C) and server (S) have identical core architectures. However, while device S receives inputs to its matched PPUF from the GPS or sensors, device C is designed in such a way that its PPUF inputs can be independently controlled from regular memory. The GPS signal contains data that indicate the location and time of sampling (data collection) at device S. The server sends both original data collected from sensors and GPS as well as their corresponding matched PPUF output. The client can easily check the correctness of the data by comparing it with the values of its PPUF output for the claimed actual data. The protocol can be further improved to prevent replay attacks, if the data that are served to the inputs of the matched PPUFs are combined using XORing with a set of challenges that are supplied by the client.

In addition to the trusted remote sensing, the proposed scheme can also be utilized for remote trusted computation. Again, we have two participating parties: client (C) and server (S). It is assumed that the client has more restricted energy and computational resources, while the server provides services to the clients using its plentiful resources. For example, side C may be a smartphone and side S may be a data center. The client can check that the results or intermediate values calculated by the server are indeed calculated by the server at claimed times using a variant of the protocol for remote sensing. Hence, the client can probabilistically check the results and the required time by the data center. This checking can be done either in real time or offline, depending on the application and resources available at the client.

IV. TESTING PPUFs

In this section, we briefly discuss aspects related to testing PPUFs. We place equal emphasis on establishing relevant testing issues, the state-of-the-art testing techniques, and current performance of various types of PUFs and PPUFs.

Testing is an essential step in qualifying ICs for the actual usage. It has two main aspects. The first is manufacturing, where the goal is to check if the IC implementation was completed without faults. The second is functional testing, where the correctness of design is evaluated. Similarly, it is crucial that the design and implementation of PPUFs are properly evaluated before their deployment.

It is easy to see that IC manufacturing testing corresponds to gate and transistor level characterization (GLC) where delay, leakage, or some other device metrics are analyzed. GLC is a well-studied topic [26]. In order to address functional testing, one has to first define security attacks that are relevant to PPUFs. Currently, it appears that prediction and emulation attacks are the most effective. In prediction attacks, the attacker uses known challenge–response pairs to guess and accurately calculate the response for challenges of interest. In emulation attacks, the attacker tries to create or identify identical, or at least similar, PPUFs to facilitate subsequent prediction attacks.

Three specific types of prediction attacks have been undertaken with remarkable success on PUFs. The first is the use of linear algebra techniques, and in particular linear programming, to quickly create a PUF model [27]. The second type employs machine learning methods to build statistical models that accurately predict the response for random challenges with high probability. Finally, recently, it has been shown that side-channel-based techniques are exceptionally effective in reverse-engineering PUFs.

There are also a number of statistical metrics for predicting the expected level of effectiveness of any prediction attack. For example, it is highly desirable that correlation between any input and any output, or between any two outputs, be close to 0.5. It is also important that the dependency of any output on a small set of inputs and a small set of outputs is such that effective prediction is not possible. These statements should stay unchanged even if a set of intermediate signals are included in the set in order to avoid prediction attacks involving the calculation of intermediate signals and subsequently use them for the prediction of one or more outputs.

It is often exceptionally important that prediction is prevented for similar inputs, in particular for two inputs that have bitwise Hamming distance 1. This phenomenon is named the avalanche effect and is often studied in classical cryptography. It is well documented that early types of PUFs and PPUFs often had very weak avalanche effect properties. While ideally the Hamming distance of the outputs should follow a uniform distribution, it often follows Gaussian distributions with a small variance. However, more recent PPUFs have significantly improved their avalanche effect distributions. One potential explanation is that some classes of PUFs and PPUFs are not sufficiently nonlinear and do not have a high enough share of mixing signals, which are two properties identified by Shannon as essential for cryptographic systems.

PPUFs can also be analyzed from a hardware point of view in terms of its predictability. For instance, one can study the extent to which two PPUFs are similar in terms of having a large percentage of identical corresponding gates that have identical responses to the same set of challenges. It would be natural to label the extreme situation where two PUFs do not match only on a single gate, a hardware avalanche effect. Preliminary simulation experiments show that it is much more difficult to find high-quality PPUF structures with respect to the hardware avalanche effect. At the same time, they show that more nonlinear PPUFs perform significantly better. We expect that the study of other hardware-based analyses of prediction and/or emulation capabilities will emerge as important criteria in the design of PUF and PPUF structures.

It is important to note that PPUF selection greatly depends on process variation and its assumed model. This dependency is complex. Too little variability makes measurements difficult. On the other hand, too much variability may also be detrimental. For instance, if one gate has a much higher delay than any other gate, it makes the PPUF an easy target for prediction attacks.

One important PPUF application is its use as a synchronized random number generator (RNG), i.e., the creation of two or more hardware RNGs with identical properties, so that no additional such RNG can be created. These synchronized RNGs can be used for a plethora of security and trust applications. Recently, Xu and Potkonjak [28] have demonstrated that such devices can be easily built using digital PPUFs. For example, the first digital PPUF design easily passed the complete NIST randomness test.

V. OPEN PROBLEMS

We finish our PPUF coverage by stating and explaining the rationale behind the most important pending challenges and opportunities. These challenges are separated to two research and development lines: PUF-related challenges and application-protocol-related problems.

A. PUF-Related Problems

1) Stability: For PPUF-based security primitives to see widespread use, their behavior must be stable under a wide range of operating and environmental conditions, which will endow the security primitive with the property of reliability and enhance their commercial viability. However, the first- and second-generation PPUF designs can be highly sensitive to power supply noise and drops, thermal variations, and use-related device aging. These parameters can affect gate delays in ways that are complex and difficult to model. For example, this can quickly introduce inconsistencies in the output of matched PPUFs. Similarly, unintentional glitching may trigger arbiters before signals from the inputs actually arrive at the arbiters. It is, therefore, important to introduce techniques that can maintain the stability of the PPUF behavior despite variable operating and environmental conditions, in a manner that does not compromise fundamental PPUF properties such as the ESG, rapid execution times, low computational overhead, etc. Digital PPUFs are a step in this direction.

2) Secrecy and Predictability: A strong security characteristic of any security primitive is the predictability of its output given some input. The obvious approach here is to expose this characteristic via a set of security tests that prove unpredictability in the behavior of the security primitive. We can identify at least two such tests. 1) The distribution of the output bits should be scrutinized for patterns. For example, the entropy in PPUF output can be tested via compression techniques with theoretical guarantees such as the context-tree-weighting method [29]. The National Institute of Standards and Technology (NIST) battery of statistical tests may also be applied to check for nonrandomness. 2) The correlation between PPUF input and output should be examined. For example, the strict avalanche criterion tests this condition by checking if, when a single input bit is flipped, the output bits flip with probability 0.5. Such tests will be crucial in determining whether an attacker can gain an advantage merely by statistically modeling the input/output relationship in a computationally tractable manner.

B. Application-Protocol-Related Problems

1) Aging-Based Protocols: Oblivious transfer protocols underlie an important class of cryptographic applications wherein a sender (S) would like to send a message to a receiver (R) such that R receives the message with probability 0.5, but S cannot ascertain whether R has received the message. For example, such a protocol has been shown to lend itself to commital-protocol-based secure multiparty communication schemes and zero knowledge proofs [30]. A simple-matching-based PPUF protocol may be devised to serve as an oblivious transfer protocol as follows: S communicates two sets of gates G_1 and G_2 that it may age. S and R independently pick one of the two sets and age the corresponding set of gates on their PUFs. The PUFs are now matched with probability 0.5. S generates a key based on the output of its PPUF which it uses to encrypt the message, and transfers it to R. Clearly, R can decrypt the message with probability 0.5. However, the problem with such aging-based security protocol designs is that device aging is a unidirectional operation and, after a few iterations, a gate may no longer be aged, effectively destroying the PPUF, or limiting its effectiveness, within just a few iterations. A solution to this problem has the potential of vastly expanding the range of protocols that PPUF-based primitives can support.

2) Suceptibility to Side-Channel Attacks: In Section III-D, we have discussed the application of PPUFs to trusted remote sensing and computing. However, in this context, PPUFs are extremely vulnerable making them susceptible to side-channel attacks. The attacker may have complete access to the PUF-integrated chip, including access to signals at the pins of the IC, revealing timing and power information among other things regarding the operations occurring on the IC. Simply encrypting all data before it is sent off chip (e.g., transfers to memory, disk, network, etc.) may not be sufficient for this purpose. Clearly, an important open problem in this context is the execution of computation and communication on PUF-integrated ICs while ensuring that no information regarding these secure operations is revealed via side-channels such as signal timing, system power usage, electromagnetic radiation, etc.

VI. CONCLUSION

We have surveyed a recently emerging hardware security primitive: PPUFs. PPUFs support creation of secure communication, storage, sensing, and computing, as well as protocols for preserving privacy and ensuring trust. While currently the two largest PPUF families are gap-based and matching-based PPUFs, new families such as digital PPUFs have been proposed very recently. We have analyzed the properties of these PPUF families, discussed techniques for testing and evaluating PPUFs, and presented representative security protocols. The main quantitative advantages of PPUFs include energy efficiency, high throughput low latency, and a small footprint. The main qualitative advantages are their flexibility in the creation of new classes of security protocols, and their permanent integration with sensing and computing systems to enable trustable flow of information. ■

REFERENCES

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL: CRC Press, 1996.

[2] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Proc. Int. Workshop Inf. Hiding*, 2009, pp. 206–220.

[3] U. Rührmair *et al.*, "Towards electrical, integrated implementations of SIMPL systems," *in Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, ser. Lecture Notes in Computer Science, vol. 6033, Berlin, Germany: Springer-Verlag, 2010, pp. 277–292.

[4] U. Rührmair, "SIMPL systems: On a public key variant of physical unclonable functions," 2009. [Online]. Available: http://eprint.iacr.org/2009/255.pdf

[5] U. Rührmair and M. van Dijk, "On the practical use of physical unclonable functions

in oblivious transfer and bit commitment protocols," *J. Cryptogr. Eng.*, vol. 3, no. 1, pp. 17–28, 2013.

[6] U. Rührmair, "SIMPL systems as a keyless cryptographic and security primitive," *Cryptography and Security*. Berlin, Germany: Springer-Verlag, 2012, pp. 329–354.

[7] R. Horstmeyer, B. Judkewitz, I. Vellekoop, and C. Yang, "Physical key-protected one-time pad," 2013. [Online]. Available: http://arxiv.org/abs/1305.3886

[8] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public physically unclonable functions: Architecture and applications," in *Proc. Design Autom. Conf.*, 2011, pp. 242–247.

[9] S. Meguerdichian and M. Potkonjak, "Matched public PUF: Ultra low energy security platform," in *Proc. Int. Symp. Low Power Electron. Design*, 2011, pp. 45–50.

[10] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, "A comprehensive model for PMOS NBTI degradation: Recent progress," *Microelectron. Reliab.*, vol. 47, no. 6, pp. 853–862, 2007.

[11] S. Meguerdichian and M. Potkonjak, "Using standardized quantization for multi-party PPUF matching: Foundations and applications," in *Proc. Int. Conf. Comput.-Aided Design*, 2012, pp. 577–584.

[12] B. Cline, K. Chopra, D. Blaauw, and Y. Cao, "Analysis and modeling of CD variation for statistical static timing," in *Proc. Int. Conf. Comput.-Aided Design*, 2006, pp. 60–66.

[13] D. Markovic, C. Wang, L. Alarcon, T.-T. Liu, and J. Rabaey, "Ultra low-power design in near-threshold region," *Proc. IEEE*, vol. 98, no. 2, pp. 237–252, Feb. 2010.

[14] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm MOSFET's: A 3-D atomistic simulation study," *IEEE Trans. Electron Devices*, vol. 45, no. 12, pp. 2505–2513, Dec. 1998.

[15] S. Chakravarthi, A. Krishnan, V. Reddy, C. Machala, and S. Krishnan, "A comprehensive framework for predictive modeling of negative bias temperature instability," in *Proc. IEEE Int. Reliab. Phys. Symp.*, 2004, pp. 273–282.

[16] T. Xu, J. Wendt, and M. Potkonjak, "Digital bimodal function: An ultra-low energy security primitive," in *Proc. Int. Symp. Low Power Electron. Design*, 2013, pp. 292–297.

[17] T. Xu and M. Potkonjak, "Lightweight digital hardware random number generators," in *Proc. IEEE SENSORS*, 2013, DOI: 10.1109/ICSENS.2013.6688562.

[18] T. Xu, W. J. B., and M. Potkonjak, "Matched digital PUFs for low power security in implantable medical devices," in *Proc. Int. Conf. Healthcare Inf.*, Verona, Italy, 2014, to be published.

[19] T. Xu and M. Potkonjak, "Robust and flexible FPGA-based digital PUF," in *Proc. Int. Conf. Field Programmable Logic Appl.*, Munich, Germany, 2014, to be published.

[20] D. A. Patterson, "Latency lags bandwidth," *Commun. ACM*, vol. 47, no. 10, pp. 71–75, 2004.

[21] J. Wendt and M. Potkonjak, "The bidirectional polyomino partitioned PPUF as a hardware security primitive," in *Proc. Global Conf. Signal Inf. Process.*, 2013, pp. 257–260.

[22] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *Proc. IEEE INFOCOM*, 2011, pp. 1889–1897.

[23] D. Liu, P. Ning, and W. K. Du, "Attack-resistant location estimation in sensor networks," in *Proc. Int. Symp. Inf. Process. Sensor Netw.*, 2005, pp. 99–106.

[24] M. Potkonjak, S. Meguerdichian, and J. Wong, "Trusted sensors and remote sensing," in *Proc. IEEE SENSORS*, Nov. 2010, pp. 1104–1107.

[25] J. Wendt and M. Potkonjak, "Nanotechnology-based trusted remote sensing," in *Proc. IEEE SENSORS*, 2011, pp. 1213–1216.

[26] S. Wei, S. Meguerdichian, and M. Potkonjak, "Gate-level characterization: Foundations and hardware security applications," in *Proc. Design Autom. Conf.*, 2010, pp. 222–227.

[27] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE Int. Test Conf.*, 2008, DOI: 10.1109/TEST.2008.4700636.

[28] T. Xu and M. Potkonjak, "Lightweight digital hardware random number generators," in *Proc. IEEE SENSORS*, 2013, DOI: 10.1109/ICSENS.2013.6688562.

[29] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems, "Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 499–503.

[30] J. Kilian, "Founding crytpography on oblivious transfer," in *Proc. 20th Annu. ACM Symp. Theory Comput.*, 1988, pp. 20–31.

ABOUT THE AUTHORS

Miodrag Potkonjak (Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from the University of California Berkeley, Berkeley, CA, USA, in 1991.

He is a Professor with Computer Science Department, University of California Los Angeles (UCLA), Los Angeles, CA, USA. He created first watermarking, fingerprinting, and metering techniques for integrated circuits, as well as first remote trusted sensing and trusted synthesis approaches, compilation using untrusted tools, and public physical unclonable functions.

Vishwa Goudar (Student Member, IEEE) received the Ph.D. degree in computer science from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, in 2013.

His research interests lie in the area of security, fault tolerance, and power management of low-power wearable systems.