OCCRA: Overt-Covert Challenge-Response Authentication Using Device-Centric Primitives

Omar Al Ibrahim and Suku Nair, HACNET Labs, Southern Methodist University

Designing scalable protocols for authentication is necessary for large-scale systems. Today, an emerging technology called, Physical Unclonable Functions (PUFs), has become very promising in this prospect. Considerable research has been devoted to improve the physical properties of PUF, yet little research has contributed to the design of PUF-based authentication protocols. In this paper, we propose a novel protocol, namely Overt-Covert Challenge-Response Authentication (OCCRA), to enable authentication and key-sharing using primitives such as PUFs. OCCRA is oracle-based and device-centric. No programmable parameters or digital storage is necessary on the edge device. On the system-end, OCCRA offers scalability, transparent ownership transfer, flexible provisioning and other benefits.

Categories and Subject Descriptors: B.7.0 [**Integrated Circuits**]: General

General Terms: Security, Design, Algorithms

Additional Key Words and Phrases: Physical Unclonable Functions (PUFs), challenge-response authentication, key sharing

1. INTRODUCTION

The increasing demand of scalable authentication has heightened the need for new security primitives and approaches. In recent years, the emergence and rapid proliferation of pervasive computing devices including mobile phones, sensors and smart cards, has created a need for more efficient authentication schemes. These computing devices are becoming an integral part of how people interact and how data is processed. Since the mid seventies, cryptography has been the basis for numerous security applications, ranging from ciphers, hash functions, to asymmetric key distribution. However, there are two major drawbacks of classical cryptographic techniques. First, many of these techniques, in particular public key protocols, are difficult to establish and maintain due to the need for complex key management infrastructures. Second, these techniques are often broken, not necessarily through algorithmic weaknesses, but because of memory leakage and side-channel attacks on the edge device.

As a result, ongoing research has been devoted to investigate device-centric primitives that elute these drawbacks. Recently, a promising technology, called Physical Unclonable Functions (PUFs)[4][15][19][29], has started to come forward for semiconductor devices. PUFs provide a method for volatile key generation and challenge-response authentication based on the random physical properties of semi-conductors. These random properties are known to be reproducible but arguably difficult to clone. In addition to their presumably unclonable artifacts, PUFs are considered tamper-evident and resilient against physical attacks[4][19][29].

Several authentication protocols have been proposed for PUFs [6][9][13][29]. However, these protocols generally require heavy databases on the back-end system which

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

need to be safeguarded against breach attacks. Furthermore, in order to provide a secure refresh of keys from PUFs, some of these protocols also rely on a pre-shared master key stored in non-volatile memory on the device, which could be easily extracted using invasive techniques. These limitations serve as major obstacles for the adoption of PUFs. In this paper, we address some of these limitations and propose a novel authentication protocol which will make PUF-based security more scalable and efficient. This protocol, called Overt-Covert Challenge-Response Authentication (OCCRA), constructs an embedded sequence of challenge-response pairs to enable authentication without additional hardware investment on key-distribution. Furthermore, OCCRA generates the sequence using an oracle-based mechanism that refreshes the system state and thus enables scalability.

In the following, we summarize the advantages of OCCRA:

— *Low-cost oracle-based authentication*: In the OCCRA protocol, we strictly confine to the capabilities of PUF to acquire and use PUF responses for authentication. No expensive encryption or hash algorithm is needed to protect against replay attacks. Since PUFs are efficient for implementation using a few number of gates, our solution is considered low-cost.
— *Device-centric deployments*: Pre-configuring devices is a difficult task in many large-scale systems. This is especially true when the keys are statically programmed in the nodes or when they are computed based on some required randomness criteria or security property (e.g. public-private key pairs). Using OCCRA, device manufacturers will be able to mass produce and distribute keys with the desired properties using standard fabrication layout and without pre-computation. Because the values derived from PUF only depends on the physical characteristics, edge devices are ready for deployment once they are manufactured.
— *Ownership transfer*: The owner of the device will be able to acquire his own set of values, and does not need to consult the manufacturer for a master key. Since PUF has exponentially-many values (w.r.t. the input length), the choice of challenge-response pairs is transparent from the manufacturer.
— *System scalability*: In our protocol, we reduce the storage complexity on the backend system by provisioning challenge-response pairs before and after deployment. Unlike one-time pads, the backend system does not need to pre-store a large set of challenge-response pairs in the database. Instead, the pairs are procured during the operation of the system.
— *Multi-user provisions*: In a shared environment, users would like to authenticate edge devices without the intervention of other parties in the system. Using OCCRA, every user is capable of extracting a set of challenge-response pairs from PUF that is independent of other users. The fact that multiple users are able to provision mutually exclusive set of values, without additional complexity on the device, provides non-repudiation and allows for multiple trust domains to be defined within a system.
— *Breach recovery*: In a security system, an insider breach of the secret keys is sufficient to compromise the system. Recovering from these attacks is infamously difficult because it requires reconfiguration of the nodes in the deployment environment. In contrast, OCCRA offers recovery from breach attacks by refreshing the backend system state using a new set of challenge-response pairs from PUF.
— *Volatile key generation*: PUF is used to derive all the challenge-response pairs needed using the random characteristics of the physical micro-structure. For this reason, OCCRA does not require any digital storage on the device. Since the pairs are physically stored in PUF, the pairs cannot be extracted using invasive attacks.

The organization of this paper is as follows: Section 2 gives an overview on PUF technology and various authentication schemes. The details of OCCRA are presented in

Section 3. Section 4 covers security and performance analysis as well as prototype implementation. Applications of OCCRA are enumerated in Section 5. Finally, we conclude with remarks and future work.

2. BACKGROUND

In this section, we motivate the discussions with a short overview on client authentication, PUF technology and PUF-based authentication schemes.

2.1. Client Authentication

Various schemes have been proposed for client authentication as surveyed by [31]. The most primitive form of client authentication is the use of static passwords. Static passwords are widely used in application domains where the user needs access to personal computers (PCs), mobile devices, and network resources such as intranets and local-area networks. In contrast to static passwords, in dynamic passwords (or one-time passwords) the client and server share a sequence of randomly generated codes as secrets which are stored in a scratch list. With this method, the client presents a code to the server every time it wants access to a system resource. Knowing the codes, the server authenticates the client and the used code is removed from the scratch list.

For more data sensitive applications such as banking and e-commerce, certificate-based authentication are used to bind digital signatures to the client's identity and to enable secure key-exchange for encryption. In this approach, the certificate contains the client's public key, which is signed by one or more certificate authorities (CAs) that the server trusts. The server presents a randomly chosen challenge and the client signs it with its private key. Although the authentication itself is rather simple, it is somewhat difficult to establish and maintain since the server also maintains a list of revoked client certificates (CRLs) in case, for example, a client's private key is compromised and must be invalidated.

Alternatively, Kerberos [22] provides third-party authentication in distributed client-server environments without a public-key infrastructure. It ensures access control and private communications between systems over a network and is effective in open, distributed environments where users have unique identities which can be verified to gain access to network services contained within their permissions profile.

As a complement to Kerberos, biometrics [12][25], smart-cards [24], and token devices are sometimes used to provide a second authentication factor. Smart cards consist of small embedded microprocessors with non-volatile memory and cryptographic functionalities. Typically, to use a smart card for authentication and access control, a user presents the card to a reader and enters the PIN associated with that card to unlock its services. The smart card, in this case, presents 'something you have' factor of security. Token devices are used in a similar fashion to prove one's identity (for example to a bank) but can also be used to generate cryptographic keys for electronic verification.

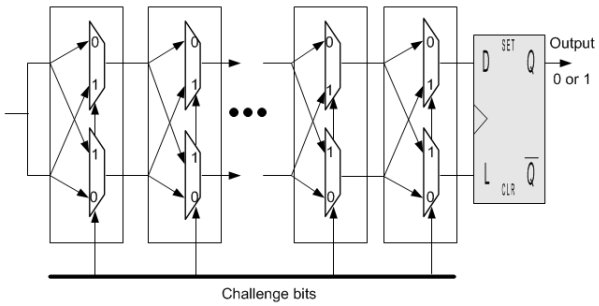
2.2. PUF Technology

Physical Unclonable Functions (PUFs) have emerged to enhance security by providing digital signatures from device manufacturing variations. Many researchers argue that these variations are infeasible to predict or replicate [4][19][29]. PUF outputs are obtained by sending a stimulus, which acts as a challenge to the circuit, to obtain a corresponding bitwise value that represents the response. Every PUF circuit holds a set of challenge-response pairs that distinguishes it from other PUF circuits, yet all of these circuits could be mass manufactured with the same process. In addition, PUF is considered an expedient tool for key generation because the responses are stored in the physical medium and not in the digital form. Hence, PUFs are considered resilient against leakage attacks [29].

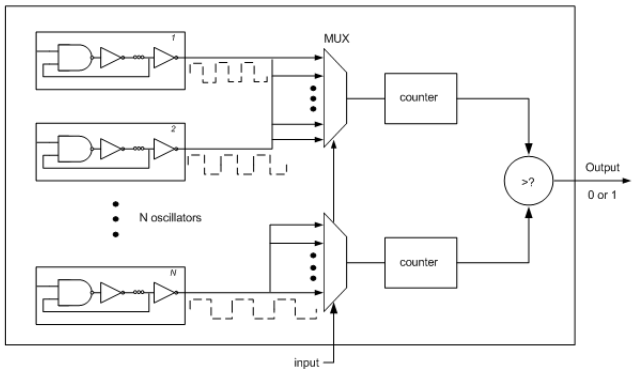
2.2.1. PUF Types. PUFs are also known to be efficient for implementation since they can be easily manufactured with a few numbers of gates. Today, there exist different types of PUFs. In this subsection, we highlight a few of these variants.

— *Silicon Arbiter PUFs*: Silicon Arbiter PUFs [19][29] use delay elements and an arbiter circuit to create a race condition between two lines (Figure 1a). This type of PUFs is available in the market and has been produced for RFID and FPGAs. Silicon Arbiter PUFs utilize a series of multiplexors that are connected and switched by the input bits. The delay in each wire and multiplexor is unique per device. In this approach, two signals contend by traversing the wires at different speeds. An arbiter is placed at the end to determine the winning signal and the set of multiplexors and arbiter are replicated to satisfy the required output length.
— *Ring-Oscillator PUFs*: Ring-Oscillator PUFs [29] are also silicon-based but use the frequency delays of multiple laid-out loops, or *ring oscillators*, to generate the response (Figure 1b). Because of variations in the manufacturing process, each ring oscillator will have a slightly different frequency. To generate the output bits, a set of oscillator pairs is selected and the frequencies are compared. The number of bits that can be generated will depend on the number of uncorrelated comparisons. In contrast to Arbiter PUFs, Ring-Oscillator PUFs are simple to fabricate and do not require accurate circuit layout.
— *Coating PUFs*: Coating PUFs [4] are produced by covering a chip with a protective semi-conductor coating doped with free ion particles at random locations. Below this layer, a layer of aluminum sensors measures the local capacitance of the coating layer to generate a digital output. Coating PUFs produce a single genetic value from sensor measurements, as opposed to multiple values as in configurable types of PUFs. Nevertheless, Coating PUFs provide an efficient method to generate secret keys. However, they require explicit randomness procedures in order to be useful since the generated outputs from these types of PUFs are not random enough for cryptographic applications.
— *Butterfly PUFs*: Butterfly PUFs [14] take advantage of the cross-coupling between two latches. Though Butterfly PUFs are based on wire delays, this type of PUFs is different than the race-based solutions. In Butterfly PUFs, cross-coupling is used to momentarily produce an unstable state in a circuit which creates a negotiation by the latches. The latches are wired to take a set of inputs. The clock signal is always set. The preset value of the first latch and the clear value of the second latch are always reset. The other pins are connected to an excite input, which at first is set, and after some time is dropped to start the negotiation. The final state of the cross-coupling is used to generate the response.

There is a promise in emerging silicon-based PUFs including Arbiter PUFs and Ring-Oscillator PUFs. These types of PUFs are highly attractive for several reasons. First, silicon-based PUFs only use intrinsic randomness from the manufacturing process. Second, they can be efficiently mass-manufactured, and third, silicon-based PUFs are configurable, which means that they can provide a large number of challenge-response pairs for every circuit. Such characteristics are valuable in providing a device-centric model for key distribution. On the other hand, Coating PUFs provide a single digital output with some disordered characteristics and require explicit enhancements to introduce randomness. Nevertheless, Coating PUFs can be used as a seed to a cryptographic hash function or a random number generator, to emulate a configurable type of PUFs.



a) Silicon Arbiter PUF



b) Ring Oscillator PUF

Fig. 1: PUF Types

2.2.2. PUF Output Characteristics. One of the important factors that affect the behavior of PUFs is *intra-chip variation*, an inherent noise factor caused by extreme changes in voltage and temperature. Intra-chip variation may alter the nominal gate parameters of PUFs. Nevertheless, several experiments demonstrated stable PUF output characteristics [4][10][19][20][29]. For instance, the maximum intra-chip variation of a Ring Oscillator PUF is 3 to 4 bit flips out of 128 bits, which is approximately 0.48 percent of the data bits. Similarly, the intra-chip variation of an Arbiter PUF roughly varies from 3.7 percent and 4.8 percent respectively for room and extreme temperatures, and up to 9 percent under voltage variation. When synthesizing Arbiter PUF circuits, gates need to be placed close to each other to minimize skews in the layout. To further reduce output noise, error-correction techniques have been proposed to stabilize the fluctuations.

In [29], Suh and Devadas proposed to compute a syndrome of the data bits for each PUF output, which are then used to reconstruct the output later at the device. To keep hardware costs marginal, the authors suggested to store the syndrome off-chip to some server. Since the syndrome is likely to be a public value, there would be some entropy loss in the regenerated values. To counter for the entropy loss in error-correction codes, Gaujardo and a group of researchers from Philips investigated the application of fuzzy extractors[8]. In their experimentations[10], they developed a Coating PUF embedded with a fuzzy extractor component to generate a uniformly random yet fixed key that is used as a seed for an encryption module. In [20], Mandel and Devadas proposed a

syndrome coding scheme to limit the entropy loss by generating pointers to the output sequence of a Ring-Oscillator PUF.

Several works have exposed weaknesses in current PUF implementations. In [17][18][19], a linear model was derived for delay-based PUFs using certain characteristics about the inter-switch and intra-switch delay variations. Earlier works on PUF modeling attacks also described successful attacks on standard Arbiter PUFs and Feed-Forward Arbiters with one loop. In a very recent paper [27], Ruhrmair et. al showed how modeling attacks can be achieved on several classical PUF implementations including Ring-Oscillator PUFs and XOR Arbiter PUFs. These attacks were based upon existing machine learning techniques. The authors described a centralized algorithm using logistic regression and evolution strategies to impersonate various types of PUFs. In their analysis, they showed a model for PUF that would behave almost indistinguishably to a true PUF given a scalable subset of challenge-response pairs.

Notwithstanding these limitations, research is ongoing to improve the physical properties of PUFs. With future technological trends in circuit design and fabrication layout, new generations of PUFs will have increased output stability and resilience against modeling attacks.

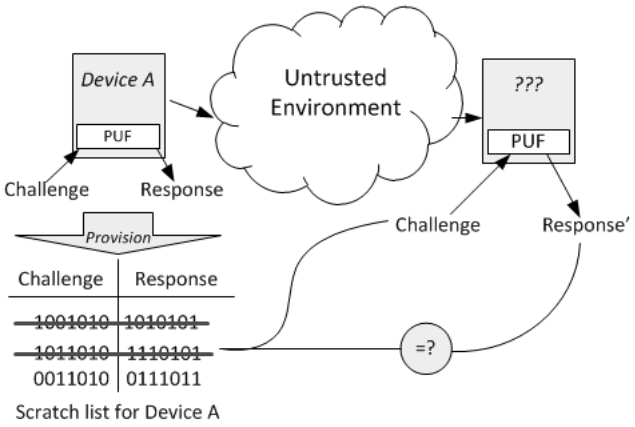


Fig. 2: One-Time Pad Authentication for PUF

2.3. PUF-Based Authentication Schemes

Several authentication protocols based on PUFs have been explored. For instance, Suh and Devadas [29] investigated a solution using one time-pad authentication (Figure 2). In their scheme, the back-end system is initially preloaded with a large number of challenge-response pairs for each device. To authenticate, the trusted party selects a challenge from one of the pre-stored pairs and sends it to the device. The device obtains the response from PUF given the challenge as input, and transmits the response. If the response matches the recorded one within some acceptable threshold then the device is accepted. Though the scheme is simple to implement, it requires extensive database storage to hold the challenge-response pairs. Furthermore, there is a limit on the number of times the system can authenticate a particular device depending on the size of the scratch list.

In [7], Bolotnyy and Robins proposed a PUF-based protocol that generates a sequence of signatures to authenticate RFID. These signatures are derived through a recursive application of PUF by itself multiple times, much like hash chains [16]. Though their protocol provides some unlinkability features, it requires synchronization between the reader and the tags. On the contrary, Kulseng et al. [13] proposed a probing scheme for PUF-based RFID using Linear Feedback Shift Registers (LFSRs) to cover code PUF responses and not expose them to eavesdropping attacks. However, this approach requires static parameterization of the LFSR in the device.

In a more recent development on PUFs, Beckmann and Potkonjak introduced Public PUFs (PPUFs) [6] to offer public-key sharing using the effects of delay lines on the output transitions. In PPUFs, the gate delay characterization of PUF is used as the public key, and the PUF circuit itself is used as the private key. To enable key-exchange, the sender uses the public key to simulate a particular output transition between a triggered input and a steady state input at some particular time. However, this approach requires accurate time measurement of the gate delays.

3. THE OCCRA PROTOCOL

In this section, we present Overt-Covert Challenge-Response Authentication (OCCRA). Since our contribution is focused on the protocol-level, we assume to have a Strong PUF [4] as a building block for the security protocol. This model of PUFs is based on complex challenge-response behavior and very many challenge-response pairs. The security features of a Strong PUF include the following: i) It must be impossible to physically clone a Strong PUF. ii) A complete brute-force attack on all the challenge-response pairs is computationally infeasible. iii) The responses of a Strong PUF should be completely random for a random selection of the challenge values, which also means it should be resilient against modeling attacks. In addition to these assumptions, we also consider a PUF that generates determinstic outputs. Thus, we base our protocol on reliable types of PUFs.

Table I: OCCRA protocol symbols and notations

Variable	Description
(o_i, r_{o_i})	Overt challenge-response pair
(c_j, r_{c_j})	Covert challenge-response pair
O	Overt set: consists of overt pairs as elements
C	Covert set: consists of covert pairs as elements
m_{ij}	Mask bits ($m_{ij} = r_{o_i} \oplus c_j$)
$P(.)$	PUF function

Table I introduces the notations/terminologies used to describe the protocol.

3.1. Protocol Stages

OCCRA consists of four stages: enrollment, authentication, refresh and flush cycles. In the following, we describe the details of each of these steps.

3.1.1. Enrollment. In the enrollment stage, the OCCRA protocol starts by registering PUF-based devices with the backend system and acquiring an initial set of challenge-response pairs from each device. All devices are assumed to have two modes of operation: *provision mode* and *secure mode*. In provision mode, the backend reader communicates with a particular device using direct challenge-response interaction with PUF. This mode of operation is used to enroll the devices to the system before they are deployed in the field. The challenge-response pairs are procured under a closed environment and stored in the backend database.

Next, devices are switched to secure mode after they have been enrolled. To protect against oracle attacks [26], provision mode is locked and readers can only use OCCRA protocol messages to communicate with a device. In the operation of the protocol, we assume an attacker has no way of maliciously gaining access to provision mode as it is safeguarded or permanently disabled after setup. This can be achieved by implementing a one-way switch logic which transitions the state of a device from provision mode to secure mode.

The challenge-response pairs provisioned by the backend system are only used to initialize the protocol. In the database, the backend system stores challenge-response pairs of the form (c_i, r_i) indexed by the device id. The number of pairs provisioned is quite small compared to the One-Time Pad Scheme in [29]. Further, the cost for system setup only depends on registering the devices to the system and extracting a small subset of challenge-response pairs from each device. Hence, there is no need to preload a large set of challenge-response pairs for every device.

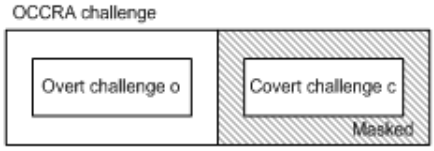


Fig. 3: OCCRA challenge message format

3.1.2. Authentication Stage. Once the challenge-response pairs are provisioned, we move to the authentication stage. Whenever a device is authenticated, it presents a static identifier that is attached to the response. The identifier is used to look up the set of challenge-response pairs that are provisioned for the device in the system database.

In the OCCRA protocol, a challenge is composed of two parts: an overt part and a covert part. The overt part of the challenge (overt challenge for short) is presented as clear text in the body of the message, while the covert part of the challenge (covert challenge) is masked by the response of the overt challenge. Together, the overt and covert parts constitute the new challenge (Figure 3). Both parts (overt and covert) need to be defined in the operation of the protocol, and the new composite challenge is evaluated atomically. The new response is the XOR of the overt and covert responses. Under the assumption of a Strong PUF, an adversary cannot extract the covert challenge from the mask because according to Shannon's theorem [28], an adversary has no advantage in deciphering the XOR if one of the components is random. Thus, an XOR is sufficient to conceal the responses, at the same time, the backend system is able to resolve the output. As a rule, the covert challenge is never presented as an overt challenge in later messages, and an overt-covert challenge-response combination is published only once, but other overt-covert combinations may be used in later authentication.

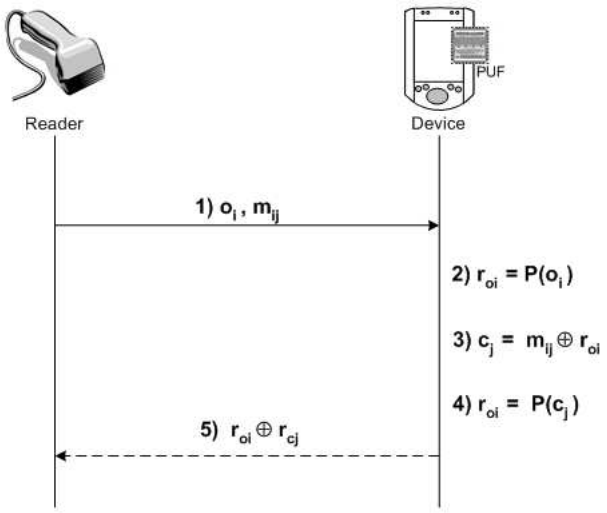


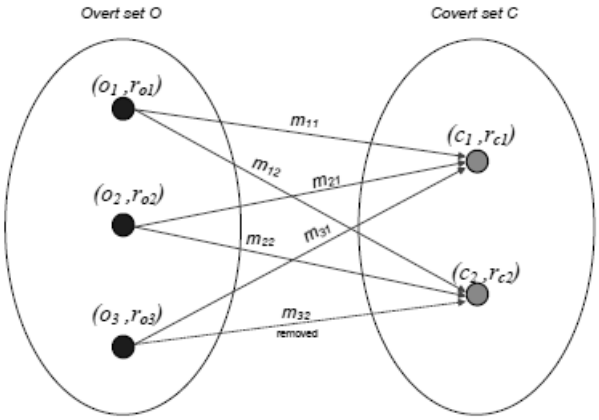
Fig. 4: The OCCRA protocol

As shown in Table I, we denote o_i as the overt challenge and c_j as the covert challenge. Also, we denote *P(.)* as PUF. The reader begins the OCCRA protocol (Figure 4) by sending an overt challenge o_i to the device. The reader also sends the mask $m_{ij} = r_{o_i} \oplus c_j$, containing the covert challenge c_j. Upon receipt, the device extracts c_j from m_{ij} by evaluating PUF with o_i and then XORing the result r_{o_i} with the mask. The device applies PUF again to obtain the response r_{c_j} from the covert challenge. In the final step, the device transmits back the XOR of r_{o_i} and r_{c_j}.

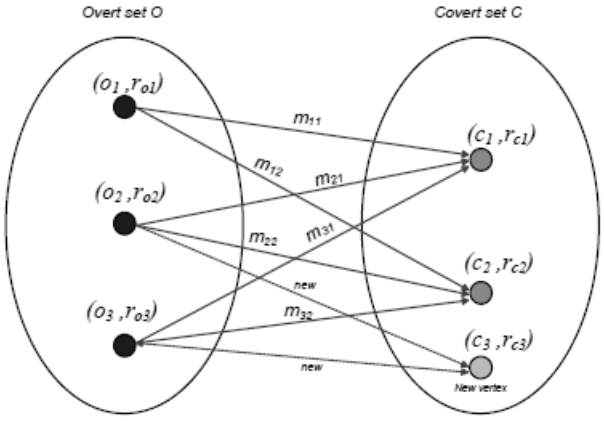
On the backend side, the provisioned set of challenge-response pairs is grouped into two categories: an overt set *O* and a covert set *C*. *O* contains all the overt challenge-response pairs (o_i, r_{o_i}), where as *C* contains all the covert challenge-response pairs (c_j, r_{c_j}). The backend system constructs a new set of challenge-response pairs (Table II) between the two sets *O* and *C*. A graphical representation of the new pairs can be described using a bipartite construction, as shown in Figure 5. The figure illustrates how to create new challenge-response pairs by taking the product of the sets. Using the bipartite construction, the backend system keeps track of the association between the overt and covert pair combinations. The vertices represent the actual challenge-response pairs derived from PUF where as edges represent the new OCCRA challenge-response pairs used in authentication. The weights represent the masks. Edges are removed once they are used but vertices are removed when they are disconnected from the graph. Figure 5 a) illustrates an example for removing an edge between (o_3, r_{o_3}) and (c_2, r_{c_2}).

In other words, the backend system constructs a new set of challenge-response pairs from the actual pool of challenge-response pairs obtained from PUF. In Table II, we illustrate the example of constructing six new pairs using three overt pairs (o_1, r_{o_1}), (o_2, r_{o_2}), (o_3, r_{o_3}), and two covert pairs (c_1, r_{c_1}), and (c_2, r_{c_2}). Using the newly constructed set, OCCRA uses the generated pairs for authentication. The backend system selects one of these pairs and presents the composed challenge (overt, mask) to the PUF device. Then, the backend system deletes the pair from the set after authenticat-

ing the device. Since we have a bound on the number of pairs in the set, the system eventually exhausts its window for authentication and will need to acquire a new set of challenge-response pairs from PUF.



a) Edge removal for m_{35}



b) Vertex addition for (c_3, r_{c3})

Fig. 5: Bipartite graph to represent challenge-response pairs

3.1.3. Refresh Cycle. One of the unique envisions about OCCRA is the aspect of refreshing the challenge-response pairs. This can be achieved by using the covert challenges to bootstrap more challenge-response pairs from PUF. Given a high-entropic PUF, there are many challenge-response pairs. The backend system randomly selects a new challenge to embed in the mask. The protocol logic remains the same using the interactions described in Figure 4. The newly selected challenge is treated as the covert part and is sent to the device. The device extracts the covert challenge from the response of the overt challenge, evaluates the covert challenge with PUF, and then sends out the XOR of the responses. Upon receipt, the system extracts the covert response from the masked response, and then adds the new challenge-response pair to

Table II: OCCRA challenge-response mappings

Overt Pair	Covert Pair	Overt Challenge	Mask	Final Response
(o_1, r_{o_1})	(c_1, r_{c_1})	o_1	$m_{11} = r_{o_1} \oplus c_1$	$r_{o_1} \oplus r_{c_1}$
(o_1, r_{o_1})	(c_2, r_{c_2})	o_1	$m_{12} = r_{o_1} \oplus c_2$	$r_{o_1} \oplus r_{c_2}$
(o_2, r_{o_2})	(c_1, r_{c_1})	o_2	$m_{21} = r_{o_2} \oplus c_1$	$r_{o_2} \oplus r_{c_1}$
(o_2, r_{o_2})	(c_2, r_{c_2})	o_2	$m_{22} = r_{o_2} \oplus c_2$	$r_{o_2} \oplus r_{c_2}$
(o_3, r_{o_3})	(c_1, r_{c_1})	o_3	$m_{31} = r_{o_3} \oplus c_1$	$r_{o_3} \oplus r_{c_1}$
(o_3, r_{o_3})	(c_2, r_{c_2})	o_3	$m_{32} = r_{o_3} \oplus c_2$	$r_{o_3} \oplus r_{c_2}$

its window. At any given point, the OCCRA protocol refreshes the system state by updating overt-covert mappings used to construct new OCCRA challenge-response pairs. The frequency of the refresh cycle is determined by the size of the window, and is tunable by the system depending on the number of times it encounters a particular device.

To avoid reuse, the backend system removes the overt-covert pair used in the bootstrap. As an example, Figure 5 b) depicts adding a new pair to the system. As shown graphically, a new vertex (c_3, r_{c_3}) is added in the covert set *C* and is connected to all pairs in set *O* except for the one used in the probe. Since o_1 is used to probe for the new mapping, the pair (c_3, r_{c_3}) is connected to all vertices in *O* except for (o_1, r_{o_1}). In this example, two new OCCRA challenge-response pairs are created by masking (c_3, r_{c_3}) with the overt pairs (o_2, r_{o_2}) and (o_3, r_{o_3}). The masks $m_{23} = r_{o_2} \oplus c_3$ and $m_{33} = r_{o_3} \oplus c_3$ are obtained by XOR and graphically represent the new edges of the bipartite construction. The label "new" on the edges in Figure 5 b) indicate the newly added OCCRA challenge-response pairs.

A key aspect of the protocol is the ability to refresh the challenge-response pairs almost indefinitely, as long as there are unused pairs from PUF. Furthermore, the backend system does not require a significant number of a priori shared challenge-response pairs at any point in time. It can simply rely on the refresh cycle to acquire new pairs.

3.1.4. Flush Cycle. A flush cycle is sometimes needed to overcome factors such as intra-chip variation, system crash, or information leakage. In a flush cycle, the backend system completely renews the challenge-response pairs stored in the database. A flush cycle can be achieved by executing multiple refresh cycles. Once a flush cycle is complete, a new set of OCCRA challenge-response pairs is generated. In the event the challenge-response pairs are lost from the backend database, a flush cycle can enable recovery given that the backend system has at least one challenge-response pair. Alternatively, the device could forcibly be reset to provision mode to acquire new pairs from PUF. However, to protect against eavesdropping, provisioning needs to be carried inside a controlled environment.

3.2. Management of the Challenge-Response Pairs

To manage the challenge-response pairs, we define a set of restrictions to construct the OCCRA challenge-response pairs from the provisioned set. These restrictions are enforced on the back-end side and do not require any special logic on the edge device.

—*Restriction 1: Every overt-covert combination is used only once*.
Given any two overt-covert combinations o_i^1 & c_j^1 and o_i^2 & c_j^2 presented for authentication, we must not have both $o_i^1 = o_i^2$ and $c_j^1 = c_j^2$. This is to ensure that every OCCRA challenge-response pair is unique. If this restriction is not satisfied, then an eavesdropper can collect OCCRA protocol messages (o_i, m_{ij}) and $(r_{o_i} \oplus r_{c_j})$ of different overt-covert combinations and replay them back at a later time to fool the backend system.

—*Restriction 2: No covert challenge is used as an overt challenge and vice-versa*.
This restriction simply states that $o_i \neq c_j$, which is necessary to prevent information leakage caused by XOR. Suppose an overt-covert combination o_i^1 & c_j^1 is presented for authentication, then the composed challenge (o_i^1, m_{ij}^1) and the response $r_{o_i}^1 \oplus r_{c_j}^1$ are published. If another overt-covert combination o_i^2 & c_j^2 is published and if $o_i^2 = c_j^1$, then $r_{o_i}^1$ can be leaked out by XORing o_i^2 with m_{ij}^1.

—*Restriction 3: Selecting overt-covert combinations to be used as challenge-response pairs should be completely random and unpredictable by an adversary*.
An adversary can exploit his knowledge of the overt-covert combination used to form a replay strategy. For example, if we have o_1 & c_1, o_1 & c_2, and o_2 & c_1 presented for authentication, then an adversary knows $r_{o_1} \oplus \mathbf{r}_{c_1}$, $r_{o_1} \oplus \mathbf{r}_{c_2}$, and $r_{o_2} \oplus \mathbf{r}_{c_1}$. Hence, the response $r_{o_2} \oplus \mathbf{r}_{c_2}$ can be obtained by XORing the three responses. A more condensed version of the attack is to obtain $r_{c_j} \oplus c_j$ after observing just one authentication by XORing m_{ij} with $r_{o_i} \oplus r_{c_j}$. If an adversary can guess when is the next time the covert challenge c_j is asked, he can replay the correct response. We will show in the analysis that the best design to confront any replay strategy by an adversary is to select the challenges in a given window completely at random.

4. EVALUATING SECURITY AND PERFORMANCE OF OCCRA

In this section, we cover the evaluation of the OCCRA protocol. We first discuss OCCRA's security properties by exploring how the protocol circumvents (or resists) various attacks. Next, we analyze the protocol overhead by comparing with the simple One-Time Pad Scheme proposed by Suh and Devadas [29] and with the refresh scheme proposed by Kulseng et al. [13]. Finally, we present a prototype implementation of OCCRA using an RFID middleware platform.

4.1. Security Analysis

Our objective here is to identify various attacks that may potentially be used against OCCRA and to analyze the protocol for each of these attacks. Before we present the analysis, we introduce a couple of definitions.

Definition 1 (XOR-Expression) - *An XOR-Expression is defined as a finite sequence of XOR operators applied to pairs of bitwise constants or variables.*

An XOR-Expression is nothing more than a Boolean expression which only uses the XOR operator. The *degree* of an XOR-Expression is the number of distinct variables in the expression. For instance, the expression $x \oplus y \oplus z$ has a degree of 3 for some distinct Boolean variable x, y, and z. We call an XOR-Expression to be in a *reduced form* if the form is expressed in the minimum degree. Given this, we can formulate an OCCRA challenge-response message as follows:

Definition 2 (OCCRA Challenge-Response Message) - *An OCCRA Challenge-Response Message is a tuple of XOR-Expressions* $M = (o_i, m_{ij}, r_{o_i} \oplus r_{c_j})$*, derived from the overt and covert pairs* (o_i, r_{o_i}) *and* (c_j, r_{c_j}).

We use this construction to analyze the OCCRA protocol against a wide range of attacks. Since OCCRA is an XOR-based protocol, an attacker can explore the algebraic properties of associativity ($x \oplus (y \oplus z) = (x \oplus y) \oplus z$), commutativity ($x \oplus y = y \oplus x$), neutrality ($x \oplus 0 = x$) and nilpotence ($x \oplus x = 0$) to extract the bitwise value.

4.1.1. XOR Leakage Analysis. An XOR-leakage analysis is carried out by using the properties of XOR to extract the values of the variables c_j, r_{c_j}, and r_{o_i} from a set of OCCRA challenge-response messages. These messages can be spoofed over the air during the course of communication between a backend reader and a PUF-based device. If we assume a Strong PUF, then we also assume that the challenge and response values are fairly random. According to Shannon, an attacker has no advantage in deriving the components of an XOR-Expression if one of the components is random. However, this only applies to the information-level. An attacker can resort to XOR leakage analysis hoping to achieve a reduction that leaks out the values of the variables. Here, we present an important theorem about the secrecy of the values in the OCCRA protocol.

Theorem 1. *For any number of OCCRA challenge-response messages,* $n > 0$*, and under Restrictions 1-3, the OCCRA protocol guarantees that the XOR-Expressions in* m_{ij} *or* $r_{oi} \oplus r_{cj}$ *do not reduce to degree 1. Hence, the values of* $\{r_{o_i}, c_j, r_{c_j}\}$ *are secret in the OCCRA protocol*.

Proof. The proof is by mathematical induction on number of messages n. First, notice that the components of an OCCRA message is either $r_{o_i} \oplus c_j$, $r_{o_i} \oplus r_{c_j}$, or their derived component $c_j \oplus r_{c_j}$. As a result, an XOR-Expression in the messages must involve an even number of variables. We first prove the theorem for the case $n = 2$ and consider a four-tuple expression $x \oplus y \oplus z \oplus w$, where the variables x, y, z, and w are selected from the set $\{r_{o_i}, c_j, r_{c_j}\}$. Recall that degree reduction is achieved in an XOR-Expression through the nilpotent property, which is equivalent to equating two variables in an XOR-Expression. As a result, it is sufficient to enumerate all possible ways of equating the four variables x, y, z, w. Table III below lists all possible cases of variable equalities of the four-tuple expression and their associated degrees.

It is clear from the table that a four-tuple XOR-Expression can never reduce to degree one. This proves the theorem for case $n = 2$.Next, we assume the theorem holds true for any XOR-Expression E_n of length $2n$ and degree d where d is even. We also need to prove that the XOR-Expression E_{n+1} of length $2(n + 1)$ is also of even degree.

Now, there are three possible expressions for E_{n+1}.

$$E_{n+1} = E_n \oplus r_{o_i} \oplus c_j \quad (1)$$
$$E_{n+1} = E_n \oplus r_{o_i} \oplus r_{c_j} \quad (2)$$
$$E_{n+1} = E_n \oplus c_j \oplus r_{c_j} \quad (3)$$

If $d = 0$ then clearly E_{n+1} has degree 2. For $d > 0$, since E_n is assumed to be in its reduced form, all variables of E_n are distinct. Therefore, we have three cases to consider.
Case 1: Both of the new variables in E_{n+1} are distinct from variables in E_n, implying degree of E_{n+1} is $d + 2$.

Case 2: Only one of the new variables equals to a variable in E_n. This implies degree of E_{n+1} is d.

Case 3: Each of the new variables equal to some variable in E_n. Hence degree of E_{n+1} is $d - 2$.

In conclusion, degree of E_{n+1} is always even and cannot be one. □

Table III: List of variable equalities of four-tuple XOR-Expressions for $n = 2$

Case #	Description of Variables Equality	Number of Tuples	Expression Degree
1	all four variables equal	$\binom{4}{4} = 1$	zero
2	three variables equal, one distinct	$\binom{4}{3} = 4$	2
3	two distinct pairs of equal variables	$\binom{4}{2} = 6$	zero
4	one pair of equal variables distinct of other variables	$\binom{4}{1} = 4$	2
5	all four variables distinct	$\binom{4}{0} = 1$	4

4.1.2. Modeling Attacks. Modeling attacks on OCCRA presume that an adversary has collected a subset of OCCRA protocol messages, and tries to derive a numerical model from this data using some algorithm to predict the responses to arbitrary challenges. In the OCCRA protocol, an adversary has no direct access to query PUF, but it can eavesdrop on the messages, or actively send out OCCRA challenges and read out their corresponding responses.

Theorem 2. *Under Restrictions 1-3 and assuming Strong PUF, it is impossible for an attacker to model PUF using OCCRA protocol messaging.*

[Note: In a Strong PUF, we require the mapping P (i.e. the PUF) is one-to-one (i.e. injective) and its image values are random.]

Proof. Suppose we are given n_o overt pairs and n_c covert pairs and n OCCRA messages from the history of messages on the variables. We divide the n messages into n_o bins based on the values of o_i, (thus the bins are labeled by the o_i's and referred to as the O-bins). Notice that one or more of the bins may be empty bins and contain no

messages at all. Now to model the mapping P, we need to carry out a supervised pattern analysis on the pairs (c_j, r_{c_j}) or the pairs (o_i, r_{o_i}). Alternatively, we may carry out the analysis on the pairs $(r_{o_i} \oplus c_j, r_{o_i} \oplus r_{c_j})$ which are published in the messages. We notice that since Restriction 2 is imposed, each bin involves a single o_i, and thus the c_j's are all distinct for messages within a bin. Hence, no leakage of information on c_j is done when restricted to messages within a bin. Moreover, the mapping P is injective and thus the r_{c_j}'s are all distinct when restricted to messages within a bin. This implies no leakage of information is done on r_{c_j} as well. Consequently, the pairs (c_j, r_{c_j}) are purely random and therefore there is no correlation among the pairs (c_j, r_{c_j}). As a result, there is no correlation among the pairs $(r_{o_i} \oplus c_j, r_{o_i} \oplus r_{c_j})$ as well because these pairs differ from the previous pairs by only an offset, and correlation is independent of offsets. Finally, since any modeling technique exploits the correlation among the pairs to learn P, this implies P cannot be modeled using messages within a bin.

Next, we need to show that P cannot be modeled from messages between the bins. To this end, we redistribute the n messages into n_c new bins based on the variables c_j and refer to them as the C-bins. Similar to the O-bins, each of the C-bins involves a single c_j and there is no leakage of information on r_{o_i} or r_{c_j} when restricted to messages within a bin. Two alternative sets of pairs within a bin are candidates for modeling P, namely $(o_i, r_{o_i} \oplus c_j)$ and $(o_i, r_{o_i} \oplus r_{c_j})$. Since the second argument of pairs from both sets is purely random and no leakage of information is done on them, this implies there is no correlation among the pairs of each set when restricted to messages within a bin. Hence, P cannot be modeled based on either set of the pairs.

It remains to show there is no correlation among pairs which involve distinct o_i's and distinct c_j's. But, this case is clearly less structured than the previous two cases and thus P cannot be modeled using these pairs as well. This concludes the proof of the theorem. □

4.1.3. Replay Attacks. If M_1 and M_2 are two OCCRA transmitted messages involving the same covert variable c_j then it is easy to show that the derived component $c_j \oplus r_{c_j}$ from the two messages are identical. This implies the system has undergone some information leakage. An adversary may take advantage of this information and XOR a future transmitted message with the derived component to replay the response of a transmitted message that involves the same covert variable c_j. Throughout this paper, we will assume that the leaked information of the system is in the form of a key that depends only on the value of a covert variable c_j, e.g. the derived component $c_j \oplus r_{c_j}$. The adversary will use the key to intrude the system once a future message is transmitted. To eliminate this source of information leakage completely from the system may seem difficult for now, but it is possible to mitigate the leakage and minimize the likelihood of using the information to replay future transmitted messages. This implies the need to design the system in such way that the amount of information leaked on the choice of the c_j's is minimal, and hence the probability that the adversary correctly guessing the right variable c_j used by the reader in transmitting a future message is minimal.

One interpretation of minimum leakage of information from the system is that the history of the messages (which are aliases for variables c_j's) contribute least information to the adversary. This implies that the probability distribution of the messages (or their aliases the c_j's), over a finite number of messages, is uniform. In theory, the c_j's form an infinite sequence of stochastic variables of which only a finite number n_c of them are available for selection at any given time, constituting a window to the infinite sequence of the c_j's. We can model the refresh cycle of the system such that once any c_j in the window is used up by the transmitted messages, a new window is opened consisting of the next n_c covert challenges in the sequence. We thus can

think of the refresh cycle as a moving window along the sequence of c_j variables. More formally, let c_1, c_2, c_3, $\cdots$ be a sequence of challenges generated according to the refresh cycle of the OCCRA protocol, where each set of n_c consecutive challenges form a window to the sequence. Hence, the entries of the k^{th} window are the challenges $c_{(k-1)n_c+1}, c_{(k-1)n_c+2}, \cdots, c_{kn_c}$.

Another important ingredient of the protocol is the choice of the c_j's. Recall that the c_j's are selected completely at random within any given window. i.e. such that the probability of selecting any of the n_c covert challenges in a window equal to $1/n_c$. We will show in the theorem below that under the above conditions for selecting the windows and selecting the c_j's within a window, the sequence of c_j's form a Poisson process and that the probability of intrusion on the part of an adversary is a Poisson probability. It is that a simple yet powerful model such as the Poisson probability model can capture the random nature of the transmitted messages. We will first prove this result and then comment on the implications of the model on the security of the OCCRA protocol.

Theorem 3. *Under Restrictions 1-3 and assuming Strong PUF, the sequence of covert challenges c_1, c_2, ... etc. forms a Poisson process whose distribution depends only on the number of covert challenges n_c in each refresh cycle. The probability distribution of the number of messages $N(c_j)$ transmitted using the covert challenge c_j has the truncated Poisson probability mass function:*

$$P(N(c_j) = k) = \frac{(e^{-1/n_c}(1/n_c)^k)}{k!(1 - P(N(c_j) <= n_o))}, \text{ for } k = 0, 1, \ldots, n_o.$$

Proof. We will be deriving the probability distribution of any given number of messages $N(c_j)$ transmitted using the variable c_j. We will first show that the sequence c_1, c_2,... forms a spatial Poisson process satisfying the following conditions [5].

(1) For finite collection of c_{j_1}, c_{j_2}, ...,$c_{j_{n_c}}$ the number of messages $N(c_{j_1})$, $N(c_{j_2})$, ... , $N(c_{j_{n_c}})$ are independent random variables obeying the restriction $N(c_{j_1} \cup \ldots \cup c_{j_{n_c}}) = N(c_{j_1}) + N(c_{j_2}) + \ldots + N(c_{j_{n_c}})$.
(2) Probability distribution of each $N(c_{j_k})$ depends on c_{j_k} only through its frequency n_{j_k}, (i.e. number of overt pairs left in c_{j_k}'s bin for a given window).
(3) There exists a parameter $\lambda > 0$ such that $P(N(c_{j_k}) >= 1) = \lambda n_{j_k} + o(n_{j_k})$
(4) The probability of c_{j_k} points overlapping is zero

We will verify that the above conditions are all satisfied using the OCCRA protocol. First, notice that since the c_j's are selected completely at random, each window defines a finite collection of random variables. Thus, for k^{th} window, the number of messages involving variable c_{j_k} and denoted by $N(c_{j_k})$ is also a random variable which is independent of any other $N(c_{j_h})$ involving a different variable c_{j_h}. Thus, condition 1 is established. Now, the probability of $N(c_{j_k})$ equals the number of overt challenges n_{j_k} available for forming a message with variable c_{j_k} divided by the total number of all such n_{j_k}'s from variables in the window. Hence the second condition is satisfied.

As for the third condition, we can proceed as follows. Were it not for Restriction 1 of OCCRA protocol, which excludes messages involving repeated overt challenges with same covert challenge, the probability of $N(c_{j_k})$ would equal to the cardinality of the overt set, $|O|$, divided by the product $|O||C|$ of cardinalities of overt and covert sets which equals to $1/n_c$. However, with Restriction 1 imposed, it is easy to show that the probability equals to $1/n_c$ plus a term with complexity of order little *o*. This establishes the third condition with $\lambda = 1/\,|O||C| = 1/n_o n_c$. Finally, the last condition is

self evident since the c_{j_k} points cannot overlap and hence the probability of overlapping equals to zero.

Having established the conditions of the spatial Poisson process, we can thus conclude that $N(c_j)$ has the Poisson distribution with mean $1/n_c$ and with probability mass function

$$P(N(c_j) = k) = \frac{(e^{-1/n_c}(1/n_c)^k)}{k!} \text{ for } k = 0, 1, \ldots, n_o$$

Notice that the above function is not a probability mass function. Since we have truncated the Poisson mass function at the value of $n = |O|$, the function does not add up to one. The truncation is imposed by the ceiling n_o on number of messages that can be sent for any given c_j. To adjust for the lost probabilities, we need to divide the probability mass function by a constant to obtain the truncated Poisson probability mass function.

$$P(N(c_j) = k) = \frac{(e^{-1/n_c}(1/n_c)^k)}{k!(1 - P(N(c_j) <= n_o))}, \text{ for } k = 0, 1, \ldots, n_o.$$

This completes the proof of the theorem. □

An immediate and interesting implication of the theorem is the following. It is well known that the Poisson distribution (and its truncated version) has a property known as memoryless property which is defined as $P(X > m + n|X > m) = P(X > n)$ for any n, m in $0, 1, 2, \ldots etc$. This implies an intruder cannot improve his/her chances of intrusion from past history of messages. This result seems in line with our observation that the probability distribution of the c_j's within any window is uniform (which implies no clue can be gained on the c_j's from the history of messages compiled over a window), and from the observation that different windows involve different c_j's (which implies history building of the messages should start over every time a new window opens). Clearly, this property gives assurances over the intruder benefiting from past history of messages provided large enough history has been built.

4.2. Performance Analysis

Figures 6 and 7 show a comparison between OCCRA and traditional PUF-based authentication schemes. The comparison is based on the number of challenge-response pairs generated with respect to the storage factor and the number of refresh cycles. As shown, two benchmarks were used in the comparison: the One-Time Pad protocol by Suh and Devadas [29] and the refresh protocol by Kulseng et al. [13]. As shown in Figure 6, the number of challenge-response pairs for OCCRA for a single window of authentication is quadratically more than One-Time Pads because OCCRA considers every overt-covert combination as a new pair. The authentication space of OCCRA is also compared with the refresh protocol over the course of the system operation (Figure 7). Here, we show the number of refresh cycles.

The protocol proposed by Kulseng et. al [13] uses every provisioned challenge-response pair only once whereas OCCRA expands the set of pairs by constructing new pairs from the provisioned set. Our plots consider the maximum number of pairs generated per window using a bipartite-graph that produced maximum number of edges. It can be easily shown, using simple calculus (e.g. [30]) , that the maximum number of edges is obtained by distributing the provisioned pairs evenly between the overt and the covert sets.

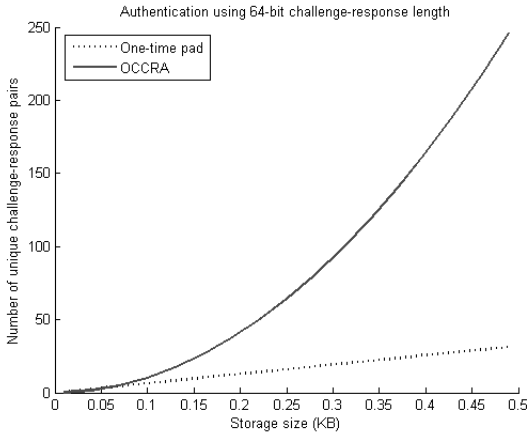


Fig. 6: Comparison between OCCRA with a single window and One-Time Pad by Suh and Devadas

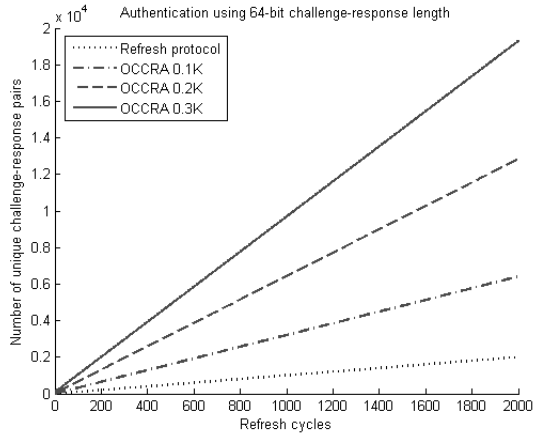


Fig. 7: Comparison between OCCRA with various window sizes and the refresh protocol by Kulseng

4.3. Prototype Implementation

We developed a prototype to demonstrate the operation of OCCRA based on GlobeRanger's iMotion Platform [2] which provides a computing environment for devices such as RFID and sensors. The prototype consists of:

(1) A database that holds an initial provision of challenge-response pairs and the OCCRA challenge-response pairs.
(2) Logic to generate the OCCRA challenge-response pairs from the provisioned challenge-response pairs.
(3) Logic to send a challenge to a tag and receive/verify the response via an RFID reader.
(4) Logic to remove a pad whose overt-covert pair has been used.

(5) Logic to generate a new challenge-response pair using a refresh cycle.
(6) Logic to update the window of OCCRA challenge-response pairs using the newly refreshed pairs.

Table IV: Provisioned challenge-response pairs

Tag ID		
Pair ID	Challenge	Response
CR_1	c_1	r_1
CR_2	c_2	r_2
CR_3	c_3	r_3
CR_4	c_4	r_4
CR_5	c_5	r_5

Table V: Overt-covert challenge-response graph

Tag ID					
Graph Edge	Overt Pair	Mask	Covert Pair	Expected Response	Used
1	CR_1	m_{14}	CR_4	$r_1 \oplus r_4$	True
2	CR_1	m_{15}	CR_5	$r_1 \oplus r_5$	False
3	CR_2	m_{24}	CR_4	$r_2 \oplus r_4$	False
4	CR_2	m_{25}	CR_5	$r_2 \oplus r_5$	False
5	CR_3	m_{34}	CR_4	$r_3 \oplus r_4$	False
6	CR_3	m_{35}	CR_5	$r_3 \oplus r_5$	False

Table 4 shows the schema for storing the initial challenge-response pairs. There is one table for each tag provisioned and the challenge-response information associated with each tag is stored in a database. Table 5 shows the information for the reconstructed pairs represented in Table 4. Each challenge-response pair is used only once. The ”Used” column indicates if a pair has been used.

The iMotion Platform contains the database to store the initial challenge-response pairs and the OCCRA challenge-response pairs for each tag. It also contains the logic sending for the OCCRA protocol and verifying the protocol response. In the messaging, iMotion sends OCCRA protocol messages to the tag via an RFID reader. It communicates with the reader using a reader protocol such as EPCglobal LLRP (Low Level Reader Protocol) [1]. The RFID reader transforms and relays the messages to the tag over an air interface (typically EPCglobal Gen 2 Air Interface).

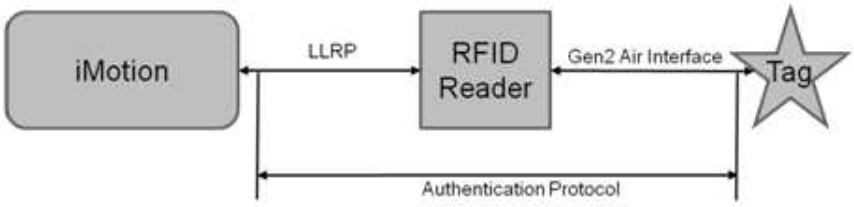


Fig. 8: Generic system view with iMotion

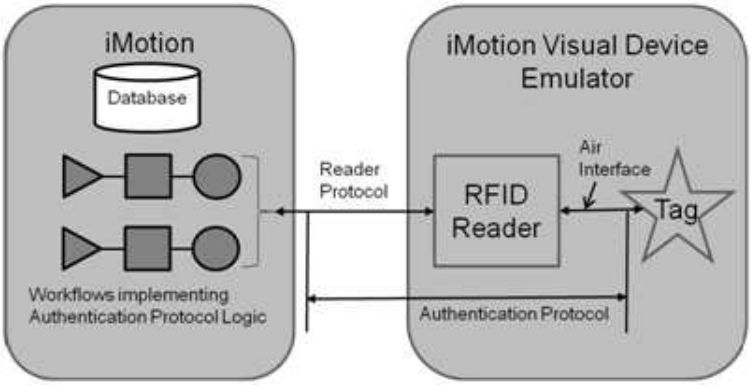


Fig. 9: iMotion demonstration system

OCCRA is transparent to the reader protocol, the air interface protocol and the RFID readers. The reader and air interface write commands are used to send the challenge-response information in OCCRA from iMotion to the tags. Afterwards, iMotion can use the read commands to get the tag's response. The read and write commands can be directed at specific tag addresses or used as tokens within the data to indicate OCCRA protocol messaging. The system used to demonstrate the protocol is shown in Figure 8 and Figure 9. It uses iMotion to implement the database and protocol logic as described above. In addition, it also uses iMotion's Visual Device Emulator (VDE) to simulate the RFID reader and the tags.

One of the lessons learnt from the prototype implementation is to have a dispatchable process for managing the removal and reconstruction of the challenge-response pairs. Furthermore, we note that for a real implementation it is more practical to keep the protocol logic device-centric through bypassing the reader protocols.

5. APPLICATIONS OF OCCRA

As discussed previously, OCCRA provides numerous advantages for authentication and key distribution. In this section, we elaborate on some these advantages and present a number of applications of OCCRA including smart grids, RFID, and biometrics.

5.1. Smart Grids

Smart grid is an emerging technology in power systems which enables monitoring and managing energy consumption. Today, smart grids are subverted to a series of attacks including masquerade, tampering, and denial-of-service [23][21]. The motivation for these attacks is high largely because they can be easily monetized. A compromised meter is used to manipulate energy costs on individual usage, fabricate meter readings on targeted victims, or simply launch large-scale attacks on the grid system, leading to blackouts. Since smart meters today adopt the standard wireless interface for communication and configuration management, this introduces a new dimension of attacks to the grid system. One of the main elements for protecting the smart grid network is to authenticate devices and control messages. Smart meters are physically exposed in residential and business areas, which can be easily tampered with, thereby allowing hackers to gain access to encryption keys and stealing identities to intrude the system.

Alternatively, incorporating OCCRA into the grid system will enable authentication and secure key provisioning without much cost or effort. PUFs can be embedded into smart meters so that grid operators automatically provision key values using control messages. PUF is then locked and OCCRA is enabled. There is no burden to manage the distribution of keys in the grid system because they are obtained from the physical properties of PUFs. This aspect is important especially for smart grids, which typically consist of thousands of nodes. In addition, the keys derived from the smart meters are selected based on the high-entropic PUFs, which consist of a large number of challenge-response pairs that is cumbersome to exhaust. Consequently, provisioned values can be periodically refreshed by the grid system using OCCRA's query interactions. This also implies that ownership transfer is transparent. In other words, grid operators do not need to contact the smart meter vendors for the master keys. Furthermore, PUFs provide a complex physical microstructure for key storage at the smart meters, thereby making them resilient against memory leakage techniques.

5.2. RFID

Another application of OCCRA is to provide counterfeiting resistance for RFID. RFID has various use cases including package validation, supply chain management, and smart cards. Basic RFID tags are vulnerable to counterfeiting mainly because scanning and replicating tags require little money or expertise, thanks to the fully field-programmable tags which are available in the market. Forging attacks on RFID vary from bypassing speedway tolls to penetrating entry systems to a facility. Several demonstrations showed the success of these attacks (e.g. [11]). Though speculative, more ambitious counterfeiters could potentially clone tags attached to medical prescription cases or even aircraft parts.

Many solutions were proposed to address RFID counterfeiting. Mechanisms such as holograms, color-shifting inks, and special printing, have largely shown to be ineffective. While cryptographic RFID tags purports to offer a seemingly strong protection, they are too expensive to be used in large quantities. Traditional item-level RFID are typically designed for mass deployment and disposability, and therefore they are characterized by severe resource limitations. As an alternative to cryptographic RFID, PUF-based RFID serve as a cost-effective solution for such high volumes of inherently resource-constrained tags.

Today, there are several tags that support PUF. For instance, Verayo [3], a leading provider of the technology, has two types of PUF-based RFID: Vera M4H and Vera X512H, which implement a 128-bit Silicon PUF. Incorporating OCCRA with PUF provides an order-of-magnitude reduction than traditional cryptographic RFID with respect to gate equivalence, power consumption, and computational delay. No encryption

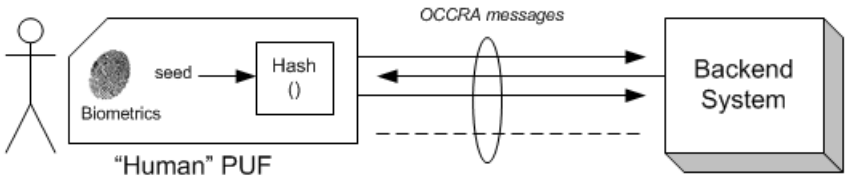


Fig. 10: OCCRA for biometric security and privacy protection

keys are generated or stored in the tag's digital memory. Because of that, OCCRA is suitable for tags with stringent memory constraints, such as EPC C1G2 [1] tags, which only support a 256-bit programmable memory. Furthermore, because many RFID applications naturally require high read rates (e.g. 150 tags/s), strong cryptography is undesirable in these environments. Instead, applying OCCRA with PUF-based RFID only takes a few gate delays, and thus it is a more efficient solution for many of these applications.

5.3. Biometrics and Hash Chains

Today, biometrics [8][12][25] are used as a form of identification and access control that uniquely identifies individuals based upon one or more physical traits such as fingerprints, facial features, and iris patterns. The first time an individual interacts with a biometric system requires an enrollment step to store the biometric information. The biometric information are subsequently captured to determine and verify the client's identity. Because many biometric systems digitally store fingerprinting information, the potential damage of any data breach is enormous. Since biometric information adheres to physical traits, thus identity theft persists once the information is compromised.

Alternatively to avoid such notorious incidents, we suggest that biometric data is fed as a seed into a cryptographic primitive, such as a hash function, to emulate a "Human" PUF. As shown in Figure 10, the system applies OCCRA to interact with an emulated biometric function and authenticates without necessarily being presented with the actual biometric data. The emulation is carried out using a memory-less path that leaves no trace of biometric data in the system. In the event of a breach, OCCRA can refresh the system state using new challenge-response pairs the next time the user interacts with the system. The concept of providing biometric security without compromising the client's identity is interesting and can be applied across various domains with similar security needs.

Departing from PUF-like applications, another application of OCCRA is to securely exchange the seed of a hash chain. Hash chain [16] is a password protection scheme proposed by Lamport which uses successive application of cryptographic hash on a single key to generate a chain of one-time keys. Due to the one-way property of hash, it is computationally intractable for an eavesdropper to generate the next key if the hash chain is used in the reverse order of its generation. However, hash chains are bounded in length, and thus when a chain is exhausted, a new seed needs to be shared between the authenticating parties to build a new chain. If the new seed is transmitted in the clear, then it may be spoofed and thus all subsequent hash computations are generated by the attacker. One way to address this difficulty is to apply OCCRA. Using OCCRA, a new seed is exchanged by sending a covert message with the next hash key. The other party simply obtains the new seed through bitwise XOR.

6. CONCLUSION

In this paper we have developed, OCCRA, a new protocol for scalable authentication and key-sharing using device-centric primitives. Unlike other PUF-based approaches, OCCRA takes advantage of PUF as an oracle to refresh the system state by constructing an embedded sequence of challenge-response pairs. We have outlined key features of OCCRA including: device-centric deployments, ownership transfer, breach recovery, and volatile key generation. We also presented an evaluation of the protocol supported by mathematical analysis and prototype implementation.

Our mathematical results demonstrated that: i) OCCRA preserves the secrecy of covert challenge-response messages, ii) An attacker model cannot be derived for PUF using OCCRA protocol messaging, and iii) A replay strategy based on the history of protocol messages will not improve the attacker's chance of successful intrusion. The formal analysis is based on mathematical induction and statistical modeling using the Poisson probability distribution to represent the probability of guessing the transmited covert challenges in each refresh cycle. The performance analysis demonstrated the scalability of OCCRA in comparison with traditional PUF-based authentication techniques. The middleware-based implementation presented the proof-of-concept.

This paper has been primarily concerned with designing scalable protocols for technologies such as PUFs. Though intra-chip variation and reverse-engineering remain inherent weaknesses of the current PUF technology, continuous academic and industry efforts are ongoing to improve the physical properties of PUFs. As for future work, we would like to explore the application of OCCRA to embedded systems and other PUF-like primitives. Moreover, extending OCCRA to mutual authentication is another interesting topic for research.

REFERENCES

EPC Global. http://www.epcglobalinc.org/home/.

Globeranger iMotion. http://www.globeranger.com/products/iMotion.asp.

Verayo Inc. www.verayo.com.

ASIM, M., GUAJARDO, J., KUMAR, S., AND TUYLS, P. Physical Unclonable Function and Their Application to Vehicle System Security. IEEE Vehicle Technology Conference, pp. 1–5.

BADDELEY, A. Spatial Point Processes and their Applications. In *Lecture Notes in Mathematics* (2007), vol. 1892, pp. 1–75.

BECKMANN, N., AND POTKONJAK, M. Hardware-Based Public-Key Cryptography with Public Physically Unclonable Functions. Information Hiding.

BOLOTNYY, L., AND ROBINS, G. Physically Unclonable Function-Based Security and Privacy in RFID Systems. Fifth Annual IEEE International Conference on Pervasive Computing and Communications, 2007. PerCom '07., pp. 211–220.

DODIS, Y., REYZIN, L., AND SMITH, A. Fuzzy Extractors: How to Generate Strong Keys From Biometrics and Other Noisy Data. Eurocrypt, Springer LNCS 3027, pp. 523–540.

FRIKKEN, K., BLANTON, M., AND ATALLAH, M. *Robust Authentication Using Physically Unclonable Functions*. Springer Berlin / Heidelberg, 2009, pp. 262–277.

GUAJARDO, J., KUMAR, S., SCHRIJEN, G., AND TUYLS, P. Physical Unclonable Functions, FPGAs, and Public Key Crypto for IP Protection. International Symposium on Circuits and Systems (ISCAS), IEEE, pp. 3186–3189.

HAN, D., AND KWON, D. Vulnerability of an RFID Authentication Protocol Conforming to EPC Class 1 Generation 2 Standards. *Computer Standards & Interfaces 31*, 4 (2009), 648–652.

JAIN, A., ROSS, A., AND SALIL, P. An Introduction to Biometric Recognition. *IEEE Transactions on Circuits and Systems for Video Technology 14*, 1 (2004), 4–20.

KULSENG, L., YU, Z., WEI, Y., AND GUAN, Y. Lightweight Secure Search Protocols for Low-Cost RFID Systems. 29th IEEE International Conference on Distributed Computing Systems, pp. 40–48.

KUMAR S., GUAJARDO J., M. R. S. C. T. P. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *Hardware-Oriented Security and Trust (HOST08)*. (2008), pp. pp. 67–70.

KURSAWE, K., SADEGHI, A., SCHELLEKENS, D., SKORIC, B., AND TUYLS, P. Reconfigurable Physical Unclonable Functions - Enabling Technology for Tamper-Resistance Storage. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), IEEE Computer Society, pp. 22–29.

LAMPORT, L. Password Authentication with Insecure Communication. *Communications of the ACM 24*, 11 (1981), 770–772.

LIM, D. Extracting Secret Keys from Integrated Circuits. Master's thesis, MIT, 2004.

MAJZOOBI, M., K. F., AND POTKONJAK, M. Testing techniques for hardware security. In *Proceedings of the International Test Conference (ITC08)* (2008).

MAJZOOBI, M., KOUSHANFAR, F., AND POTKONJAK, M. Techniques For Design and Implementation of Secure Reconfigurable PUFs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS) 2*, 1 (2009).

MANDEL, Y., AND DEVADAS, S. Secure and Robust Error Correction For Physical Unclonable Functions. *IEEE Design and Test* (2010), 48–65.

METKE, A., AND EKL, R. Security Technology For Smart Grid Network. *IEEE Transactions On Smart Grid 1*, 1 (2010).

NEUMAN, B., AND TS'O, T. Kerberos: An Authentication Service for Computer Networks. *IEEE Communication 32*, 9 (1994).

PETERSON, G., AND FRINCKE, D. Smart-Grid Security Issues. IEEE Security & Privacy, pp. 81–85.

RANKL, W., AND EFFING, W. *Smart Card Handbook*. John Wiley & Sons, 1997.

RATHA, N., CONNELL, J., AND BOLLE, R. Enhancing Security and Privacy in Biometrics-Based Authentication Systems. *IBM systems Journal 40*, 614-634 (2001).

RHRMAIR, U., SLTER, J., AND SEHNKE, F. On the Foundations of Physical Unclonable Functions. Cryptology ePrint Archive, Report 2009/277, 2009. `http://eprint.iacr.org/`.

RUHRMAIR, U., SEHNKE, F., SOLTER, J., DEVADAS, S., SCHMIDHUBER, J., AND DROR, G. Modeling Attacks On Physical Unclonable Functions. Proceedings of the 17th ACM Conference on Computer and Communications Security(CCS10).

SHANNON, C. Communication Theory of Secrecy Systems. *Bell System Technical Journal 28*, 4 (1949), 656–715.

SUH, G., AND DEVADAS, S. Physical Unclonable Functions for Device Authentication and Secret Key Generation. ACM IEEE Design Automation Conference, pp. 9–14.

VAPNYARSKII, I. Lagrange Multiplier. *Encyclopaedia of Mathematics*, Springer (2001).

WEIGOLD, T., KRAMP, T., AND MICHAEL, B. Remote Client Authentication. IEEE Security and Privacy, pp. 36–43.