Metamorphic Framework for Key Management and Authentication in Resource-Constrained Wireless Networks

Raghav V. Sampangi, and Srinivas Sampalli
(Corresponding author: Raghav V. Sampangi)

Faculty of Computer Science, Dalhousie University
6050 University Ave, PO Box 15000, Halifax, NS B3H 4R2, Canada
(Email: raghav.vs@ieee.org)

(Received Mar. 1, 2016; revised and accepted May 12 & June 10, 2016)

Abstract

The advent of the Internet of Things (IoT) has prioritized development in unique identification and sensing technologies, which facilitate IoT's automated and intelligent vision. Data security is critical to the success of such applications, more so with the communication over a wireless channel. However, IoT devices are resource limited and lack the ability to perform sophisticated computations without impacting their longevity requirements, or increasing the cost. This encourages creation of 'lightweight' security solutions for such low resource devices. We propose a new reconfigurable (or, metamorphic) framework for key management and authentication in this paper. Our framework deploys multiple lightweight algorithms and chooses one of them for each message exchange. We evaluate our work using assessment of key sequences, hardware resource utilization assessment and security analysis.

Keywords: Authentication, key management, reconfigurable security framework, resource-constrained wireless networks

1 Introduction

Several present day applications, including those in the military and healthcare, are built on the foundation of emerging wireless technologies. In particular, they employ radio frequency identification (RFID) and wireless body area networks (WBAN) as they facilitate unique identification and remote health monitoring. These technologies help with remote monitoring of military personnel, asset tracking in hostile territories and remote health monitoring of patients, among other applications. Advances in these technologies and in the capabilities of backend monitoring technologies such as cloud servers have further supported their use as central entities of the automated vision of the Internet of Things (IoT).

In RFID systems, electronic circuits called RFID tags store a unique identifier that helps uniquely identify any object, while details about such objects are stored in the backend server. These tags could be passive (without an on-chip power source) or active (with a power source). While the former are energized by the electromagnetic signals transmitted by an RFID reader (or interrogator) and respond to the reader queries, the latter can either respond to queries by the reader or initiate communications by themselves. A third category of tags, called semi-passive tags, have an on-chip power source, but still require the reader to initiate communication [12]. The lack of an on-chip power source imposes restrictions on the amount and type of computations that a passive tag can perform, making it resource-constrained. In WBAN systems, the on-body sensors that record data communicate with the hospital monitoring station through an on-body WBAN hub, and typically through a mobile device configured to be a personal server [17]. WBAN sensors could be required to stay on an individual for a long duration of time, depending on the application (especially in remote healthcare). This would require optimization techniques to ensure that the sensors can function for a longer time without frequent maintenance. One way to accomplish this is to configure the sensor nodes to 'sleep' and be woken up by the hub prior to communication [15]. This is similar to the function of passive RFID tags, and is also the main reason why WBAN sensors remain resource-constrained.

The restrictions on computational abilities of these devices necessitates a trade-off between cost, security and available resources. This further implies that data security solutions that can be deployed might be limited in their sophistication. Data security and privacy are critical in such applications, since the data can be uniquely identified with a specific individual [33]. Furthermore, with mode of communication being wireless, the communication can be 'snooped on'. A straightforward approach is to deploy cryptographic algorithms that are customized for application in low resource applications [20]. However, with most of the algorithms published and available in the public domain, the unpredictability and hence the security of an approach comes down to the strength of its keys.

We set out to determine *whether it was possible to create a framework that would constitute multiple algorithms for key management and authentication, and would enable dynamic choice of one of the deployed algorithms for each message communication – all this without any explicit communication phase for key exchange and agreement of the algorithm being used*. We wanted to explore the possibility of deploying such a set up in the context of resource-constrained wireless networks. In this paper, we discuss our metamorphic framework to accomplish such a functionality. Our approach (Section 3) is designed to support multiple algorithms for key management (and authentication), while facilitating context sensitive and deterministic choices of one of the available algorithms to accomplish several security goals. Although our framework is designed to be generic, applicable to all symmetric cryptosystems especially in resource-constrained wireless networks, we discuss a use case for the framework. The use case (Section 4) considers deploying two algorithms in the framework, namely — GeM2 key management and authentication mechanism based on gene mutation and transfer (Protocol B in [34]), and Butterfly1 (Butterfly key generation and encryption scheme [35]). To evaluate the framework with this use case, we consider key sequence assessment, hardware complexity assessment and security analysis (discussed in Section 5).

2 Related Work

The term *resource-constrained wireless networks* encompasses a wide variety of technologies and applications that are disparate, and have varying requirements. Such requirements could include resource (memory and computational ability) requirements, data storage requirements, but have a common security requirement. Since much of the resource-constrained wireless networks such as RFID systems, WBAN systems, Vehicular Ad-hoc Networks (VANETs) and the like, are created as autonomous systems to facilitate one independent activity in our lives, they are all in some way related to peoples' personal data. This places an emphasis on protecting data being communicated in such systems, thereby preserving the privacy of the individual(s) in question. We discuss some of the existing work in two resource-constrained systems, namely, RFID systems and WBAN applications.

Conventional systems rely on the following broad techniques for security — either on using shared secret keys and complex substitution/permutation functions as with symmetric cryptosystems, or on longer key-pairs and complex mathematical functions as with asymmetric cryptosystems [28]. However, restrictions in resource-constrained wireless networks limits the size of keys that can be used and the type of operations that can be performed, while ensuring security, longevity and keeping costs low. Each fundamental element of security, thus, needs to be customized and adapted for application in resource-constrained wireless networks. In this section, we discuss some of the mechanisms to accomplish key management and authentication resource-constrained wireless networks.

Pseudorandom number generators (PRNGs) are a popular choice for cryptographic algorithms for key generation in resource-constrained wireless networks. This is primarily due to their ability to generate unique sequences with different seeds. They can also generate sequences with large periods without repeating sequences [4]. Such algorithms have also been considered for deployment in RFID applications for key sequence generation [25, 26, 29]. Their work ranges from including non-linear filter functions to ensure dispersion of bits in the pseudorandom sequence to varying feedback polynomials to generate pseudorandom sequences. It must be noted however, that PRNG-based techniques help accomplish only key generation and management, requiring them to be combined with other techniques such as hash-based or trusted-third party-based approaches for authentication.

Hash and key-ed hash algorithms are typically used with key generation systems, such as the ones discussed above, to accomplish authentication. One such approach [10] suggests the use of the new SHA-3 algorithm (Keccak algorithm) [3]. In their work, a combination of pseudorandom numbers, encryption keys and the RFID tag ID are used to compute message digests, and the encrypted key is updated on successful mutual authentication. Hashing algorithms are also employed by Hakeem et al. [13] for authentication in their proposal, where they use timestamps for key generation. Their work relies on two separate timestamps, one each generated at the server/reader and the tag. Their work also employs a linear feedback shift register (LFSR) to update keys. The first part of the protocol depends on each entity authenticating the other based on the difference in timestamps between the previous acknowledged timestamp and the current timestamp, and the XOR value of this timestamp difference with the secret tag key, k_t. Tag authentication by the server involves the tag sending a hash of its ID and the upper half of the secret key, K. Key updates at the server and the tag involves updating two secret keys and the timestamp, where the keys are updated using the previous values as seeds to the LFSR, while the current timestamp becomes the new stored timestamp value at the tag.

Hashes, especially keyed hashes, in particular are popular ways of accomplishing authentication in symmetric algorithms. Dong et al.'s work on RFID authentication [10] employs the new SHA-3 standard (Keccak algorithm) [3] to compute the message digests using a concatenation of pseudorandom numbers, keys and the tag ID. Pseudorandom numbers are updated with each communication and are sent in the open, along with the hash containing an internally updated key. The key is updated on successful mutual authentication of the entities. The authors discuss various cases of operation, accounting for loss of tag acknowledgement messages and de-synchronization attempts. Hashing algorithms are also employed by Hakeem et al. [13] for authentication in their proposal, where they use timestamps for key generation.

Shi et al.'s work [36] exploits physical characteristics for security and (one-way) authentication in WBANs. Rather than having the sensors depending on cryptography for authentication, their work, BANA, considers using physical layer characteristics unique to the sensors; specifically, the variation in received signal strength (RSS) in the communication channel. The WBAN controller unit authenticates the on-body sensor nodes based on expected variations in received signal strengths of their individual responses and based on a threshold on the response time. The authors claim that attackers would experience "larger fluctuations due to multipath effect and Doppler spread than on-body sensors", making it a feasible authentication scheme. Mutual authentication among sensors or between sensors and the controller unit does seem to impact the limited resources, especially sensor battery life, in the long run, since BANA expects all sensors to compute the average RSS variations and authenticate other entities. This is mainly because authentication is an independent functionality in these sensors, which are required to include separate deployments of key management and encryption algorithms. Although the design of BANA is innovative in using physical channel characteristics for authentication, the need for separate implementation for key management implementation imposes an additional overhead on the resource-constrained sensors.

Message digests, digital signatures and third party certificates are common forms of accomplishing authentication among communicating entities. A different approach to accomplish this is a 'certificateless' manner, proposed by Liu et al. [23]. Their approach uses a trusted entity called the public key generator (PKG) that generates partial public-secret key pairs for each entity on the WBAN. Entities further request the PKG to generate the corresponding partial secret key using the entity's ID as the partial public key. The certificateless signature includes a message hash, the result of exponentiation operation applied on the public key of the PKG and the signer, its partial secret key and a random integer. This serves as a mechanism to accomplish authentication and message integrity verification. However, it is unclear whether this scheme is designed to authenticate entities on each update. This is because mutual authentication using public key infrastructure, regardless of how secure it is, will place an increased load on the already resource-constrained entity, whether it is a personal server or a sensor.

In a different take to key agreement and refresh, Zhu et al. [48] present a scheme that employs linked key updates, encrypted using the XTEA cipher (extended TEA [46]). In their work, keys are divided into 32-bit blocks and are updated block-wise on successful authentication. One thing to note is that their algorithm is prone to de-synchronization attacks, since the tag update is contingent on server authentication based on the server response, $m2$. If an adversary were to block $m2$ and transmit an unrelated $m2'$, the tag would not be able to authenticate the server, causing it to roll back its key update, thereby disrupting future communication.

Our discussion up to this point has focused on individual algorithms and combinations thereof to accomplish key management and authentication in RFID systems and WBAN applications. We next discuss algorithm frameworks, or a collection of algorithms used to accomplish a single or several security goals. A multi-algorithm encryption framework for active RFID tags has been proposed by Zhou et al. [47], an improvement of which is a generic optimized proposal for reconfigurable security co-processor work by Li et al. [21]. In their work, the control and data logic module chooses one of four encryption algorithms, namely AES, DES / 3DES (Data Encryption Standard), RSA (Rivest-Shamir-Adleman) public key cryptosystem, and ECC (Elliptic Curve Cryptography)-based cryptosystem. Their work is deployed in an FPGA (Field Programable Gate Array)-based active RFID tag, where the design allows for reconfigurability and customization. The control / data logic module chooses the applicable encryption algorithm, in addition to the appropriate memory module, initializes the FPGA-based execution unit and performs the encryption. Their use of FPGA-based design is based on the reconfigurability rationale of the work by Jones et al. [18]. Jones et al. argue that a silicon-based implementation is not suggested for the design to be (re-)configurable. However, when we consider low resource devices such as passive RFID tags, one does not have any other option than implementing the custom security algorithms on silicon chips. Reconfigurability in such cases, can be accomplished by using hardware switches that can route data to the appropriate 'path' of the chosen algorithm for processing. This is the rationale we adopt in designing our framework for security.

The specified ISO/IEC 29167-1:2014 standard for RFID security [16] and IEEE 802.15.6 for WBANs [14] provide means for manufacturers of devices conforming to these standards to deploy multiple encryption algorithms on the devices as part of the respective security suites. From the available algorithms, the entities can select one for use for a particular session, during security association. When agreeing on the algorithm to be used, however, the choice is typically communicated in plaintext, available for an eavesdropper to learn about the system states. This reduces the overall uncertainty associated with the system. The approach we adopt, however, involves a random choice of one of the available algorithms, based on a previously agreed and synchronized timestamp, which increases the overall unpredictability and thus, the security of the system.

An appropriate mechanism to accomplish adaptive

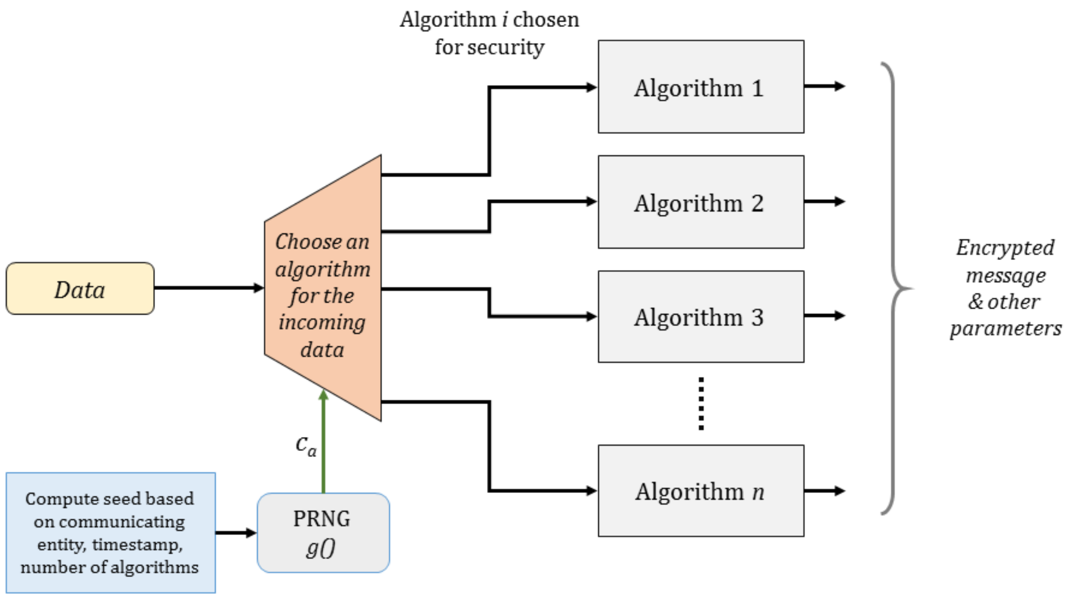

Algorithm *i* chosen for security
Data
Choose an algorithm for the incoming data
c_a
Compute seed based on communicating entity, timestamp, number of algorithms
PRNG g()
Algorithm 1
Algorithm 2
Algorithm 3
Algorithm *n*
Encrypted message & other parameters

Figure 1: Overview of the multi-algorithm framework

computations to yield achieve high security, thus, is to use reconfigurable computing. Although reconfigurable computing is not new, with its applications being found in multimedia and other embedded systems [2, 24, 40], and even for securing the Internet (reconfigurable cryptography solutions for IPSec-based architectures [8]), we present its use in key management and mutual authentication. Our approach depends on the communicating entity and the timestamp, making it minimally context sensitive, and enables resource-constrained devices to dynamically choose one of the available algorithms for key management and authentication. Since it presents the perception of changing its structure with its dynamic algorithm choice, we refer to it as a *metamorphic* framework. We draw inspiration for our work from the functioning of a chameleon, which changes its color based on its surroundings. In the sections that follow, we discuss the proposed metamorphic framework, followed by presenting a use case for the same.

3 Proposed Metamorphic Framework for Key Generation and Authentication

Our work proposes a reconfigurable security framework for resource-constrained wireless networks with the main objective of accomplishing multiple security goals with simple logical operations. The central concept here is a mechanism for choosing one of the deployed algorithms for key generation and authentication, based on the contextual information. This approach is inspired by the functioning of a chameleon that changes its color based on the color of its surroundings.

Let us consider a conventional scenario with a system having one pre-defined algorithm for each aspect of security. Most systems use such an architecture and this works when all parameters other than encryption keys are pre-defined. In such cases, the uncertainty of the system operation remains limited. However, algorithms such as IPSec [11] are considerably better in the security than the former, with the entities choosing one of the pre-agreed algorithms with a security association phase just as the communication session begins. We derive motivation for our proposed framework from this aspect of being able to change algorithms and dynamically so, however, we remove the need for an explicit security association phase where the entities would agree on one of the available algorithms.

Figure 1 illustrates the overview of our proposed framework. Imagine that a system has N encryption schemes, each being a composite of algorithms to accomplish key management, encryption and authentication. Central to our framework is a mechanism to choose one of the available algorithms automatically and in a synchronous manner. We refer to this as the *algorithm choice logic*. This logic uses a unique combination of the ID (identifier) of the resource-constrained entity, the initial deploy-time timestamp (t_0) and an incrementing integer number, r_{ac}, in the range $0...(n-1)$, to determine which of the N schemes will be chosen to generate keys for data encryption and generate authentication parameters for a particular message transfer. The integer r_{ac} has a modulus of N, i.e. it 'wraps around' on reaching N (Equation (1)).

$$r_{ac} = (r_{ac} + 1) \bmod N. \tag{1}$$

The 'choice' aspect of the algorithm choice logic is accomplished by a pseudorandom number generator (PRNG), $g()$, that uses a combination of the ID, t_0 and r_{ac} as the seed. This seed, $seed_{ac}$, is generated as summarized by Equation (2). As the seed is always changing

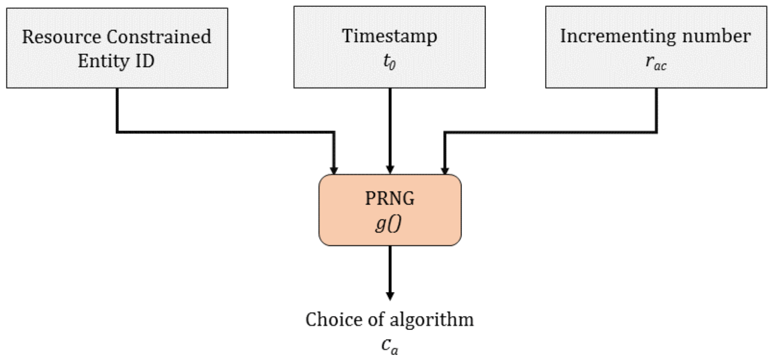

Resource Constrained Entity ID
Timestamp t_0
Incrementing number r_{ac}
PRNG $g()$
Choice of algorithm C_a

Figure 2: Illustration of algorithm choice process

in this approach, our approach ensures that the PRNG generates a new pseudorandom choice of the algorithm

$$seed_{ac} = ID \oplus t_0 \oplus r_{ac}. \tag{2}$$

Here, ID is the identification number associated with the resource-constrained entity (e.g. RFID tag or WBAN sensor); t_0 is the deploy-time timestamp; and, r_{ac} is the incrementing integer number, whose computation is explained by Equation (1). $\oplus$ represents the Exclusive-OR (XOR) operation.

The chosen algorithm, C_a, is determined by generating a pseudorandom number (PRN) using $seed_{ac}$ as the seed for $g()$. The number generated has a modulus of N.

$$C_a = g(seed_{ac}) \bmod N. \tag{3}$$

The deploy-time timestamp, t_0, is stored on the resource-constrained entity just prior to deploying it in its application environment. This is among the first pre-shared attributes, along with the ID and the initial encryption parameters associated with each algorithm. Furthermore, this will be updated during the course of operation as discussed later in this paper. This will ensure that the system state remains unpredictable to an observer. We employ the same timestamp in the algorithm chooser logic to re-use the stored and synchronized data, and to capitalize on the added uncertainty it provides. This specific combination of numbers, i.e. the timestamp, t_0, the incrementing number, r_{ac}, and the ID, changes continuously owing to increments in r_{ac} and at random with changes to t_0. This ensures that the N algorithms have a fair chance in being chosen for a specific encryption cycle. The operation of the seed generation and the corresponding PRN generation are illustrated by Figure 2. The function of the algorithm choice logic in Figure 1 is realized using C_a as the *select* input to choose one of the N algorithms that will be used to generate keys for message encryption and generate parameters for authentication.

When transmitting messages, our framework verifies the length of the message to be transmitted (M_{TI}). The length of the transmitted message is always considered to be the length (in bits, λ) of the longest message among all the algorithms. This ensures that messages are of consistent length and ensures uncertainty of the chosen algorithm.

If one entity cannot authenticate the other, it will start an internal counter to keep track of the number of erroneous messages or failed authentication attempts. If the next message results in a successful authentication, the internal counter is reset to 0, and communication proceeds as directed by the chosen algorithm. In case the counter reaches 3, flags the communication as erroneous and acts as required by the system implementation.

Our framework can be tweaked to include more (or reduced) choice in algorithms, depending on the application needs and the extent of constraints on the available resources. Thus, our framework has an implicit support to scalability, with minimal changes necessary to accommodate more algorithms in the framework. The changes would be in updating the algorithm chooser logic, specifically by updating N and a possible change to the circuit to extract C_a using modulo operation. Algorithm 1 summarizes the working of our framework algorithm (for a case when number of algorithms, $N = 3$).

4 Use Case

In this section, we discuss a use case for the framework, considering two previously published algorithms for key management and authentication [34, 35] as the constituent algorithms of the framework. We discuss customizing the framework for this use case and present a protocol of operation (that can be generalized for other use cases as well).

4.1 GeM2: Key Management and Authentication based on Gene Mutation and Transfer

GeM2 is a key management and authentication algorithm inspired by the mechanism of gene mutation and transfer in living organisms [34]. This features keys linked in a manner to the ‘parent-child’ relationship in organisms. Key update in GeM2 proceeds as follows — entities are

Algorithm 1 Algorithm choice & message transmission

```
1: BEGIN
2: Input: $rac \Leftarrow r_{ac}$, $t0 \Leftarrow t_0$, and $nAlg \Leftarrow 3$
3:
4: $rac \Leftarrow (rac + 1)\ mod\ nAlg$ //Generate seed.
5:
6: $seedAC \Leftarrow rac \oplus t0 \oplus ID$
7:
8: $CA \Leftarrow g(\ seedAC\ )\ mod\ nAlg$ //Choose algorithm.
9:
10: if $CA = 0$ then
11:    Algorithm chosen = algorithm 1
12:    Perform any other related actions
13: else if $CA = 1$ then
14:    Algorithm chosen = algorithm 2
15:    Perform any other related actions
16: else if $CA = 2$ then
17:    Algorithm chosen = algorithm 3
18:    Perform any other related actions
19: end if
20:
21: if $\lambda_{MTI} < \lambda_n$ then
22:    $MTX \Leftarrow MTI \parallel g(i)$
23: else if $\lambda_{MTI} = \lambda_n$ then
24:    $MTX \Leftarrow MTI$
25: else if $\lambda_{MTI} > \lambda_n$ then
26:    MTX consists of chunks of length, $\lambda_{MTI}$
27: end if
28:
29: if $authentication = failed$ then
30:    $AuthFailCounter{+}{+}$
31:    if $AuthFailCounter = 3$ then
32:       Flag error
33:    end if
34: else if $authentication = success$ then
35:    $transmit\_message\ (\ MTX\ )$
36: end if
37: END
```

initialized with an initial key and seeds for the PRNG. For each new key generation, a new seed is first computed using the linear recurrence formula [41], $seed_i = seed_{i-1} + seed_{i-2}$, i.e. by summing the previously used (or initially stored) seeds. Using this seed, a pseudorandom number ($numX$) is generated. A *mutation* pattern is then generated as follows — the '1' bits of the parent key (or initial key for the first key generation) are first 'preserved' by inverting the parent key, and performing an AND operation on this inverted parent and $numX$. This mutation pattern is then imposed on the parent by the XOR ($\oplus$) operation, i.e. $new\_key = parent\_key \oplus mutation$. With PRNG seeds being updated with each successful (acknowledged) authentication, $numX$ also used to generate the authentication-synchronization parameter (asv_i), computed as a hash of $numX$ XOR-ed with a special pattern called $pattern_{asv}$.

To add uncertainty to an otherwise straightforward key generation mechanism, GeM2 introduces random choices to update keys. The first random choice is at the beginning of key generation, where it checks whether to use the current parent key or update it (referred to as evolution of the parent). This is based on a random choice between 0 (continue with key generation) and 1 (parent evolution). Following this, a state identification parameter (g) is checked to see if it is equal to $genLimit$. $genLimit$ limits the number of 'child' keys for a parent to the specified number. If $genLimit = 5$, it means that a parent key can generate up to 5 keys before being forced to evolve. Of course, the uncertainty is in the combination of these two parameter checks, which implies that a parent key can either generate no child key or up to a maximum of $genLimit$ child keys. Each time a parent evolves, g is reset to 0, while the current parent number is identified by another state identifier (p). The parameters p and g for a given entity pair initialized with a common set of values, will specify the current state of the system, enabling only authorized entities to be able to determine the appropriate parameters given a specific state, (p_i, g_j). This facilitates key generation, as well as mutual authentication.

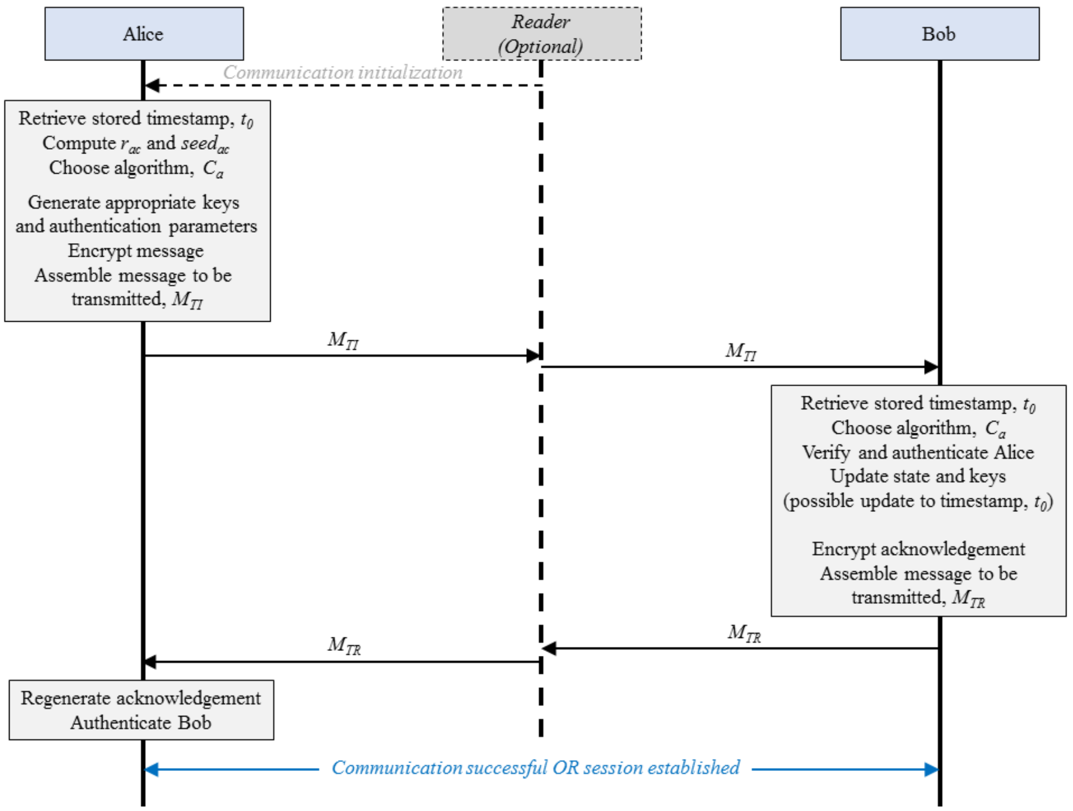
4.2 Butterfly1: Butterfly Key Management and Encryption

Butterfly encryption scheme [35], which we refer to as Butterfly1, employs a dynamic PRNG seed update technique that is naïvely based on the concept of Butterfly effect[1] [30]. In this seed update technique, the communicating entity chooses an integer, j at random ($0 \leq j \leq m$, with m being the length of the PRNG seed). The value of j determines the bit in the seed to be inverted. This single bit change will result in a completely new seed, which further results in a completely different set of pseudorandom sequences that are output from the PRNG.

The key management and authentication mechanism in Butterfly1 uses the Butterfly seed update mechanism, in addition to timestamp (t_i) and the updated value of the seed (s_j) to generate two keys for encryption. The first key, K_i, is computed as a pseudorandom number, generated using a seed that consists of an XOR combination of the updated seed and timestamp. The second key, K_T, is referred to as the transfer key and is computed as a pseudorandom number, generated using s_j as the seed.

Butterfly1 uses a multiple enveloping technique for enciphering the data. First, the data is XOR-ed with the updated seed, s_j, considered as the first envelope (resulting in an initial encrypted message, m_i). Following this, m_i is encrypted using K_i using a symmetric key algorithm (XOR is used in [35] due to its involutory property, i.e. it is its own inverse) generating the intermediate ci-

[1] *Butterfly effect* is a concept in chaos theory, defined by Polinar [30] as "hypersensitivity to perturbation". This means that in a non-linear deterministic system, if the initial conditions are changed ever so slightly, there will be drastic changes in the output of a later state.



Alice
Reader (Optional)
Bob
Communication initialization
Retrieve stored timestamp, t_0
Compute r_{ac} and $seed_{ac}$
Choose algorithm, C_a
Generate appropriate keys and authentication parameters
Encrypt message
Assemble message to be transmitted, M_{TI}
M_{TI}
M_{TI}
Retrieve stored timestamp, t_0
Choose algorithm, C_a
Verify and authenticate Alice
Update state and keys
(possible update to timestamp, t_0)
Encrypt acknowledgement
Assemble message to be transmitted, M_{TR}
M_{TR}
M_{TR}
Regenerate acknowledgement
Authenticate Bob
Communication successful OR session established

Figure 3: Framework: Protocol of operation

phertext, c_i. Butterfly1 also uses an XOR combination of the updated seed (s_j), the initial encrypted message (m_i) and the timestamp (t_i) as the seed for a PRNG, using the generated pseudorandom number as the message signature, θ_i. Next, the concatenation of c_i, t_i, the message sequence number (i), and θ_i, is encrypted using K_T (using the same symmetric key algorithm used for the earlier encryption using K_i). This becomes the final envelope over the data. Butterfly1 relies on changing values of the timestamp, the updated seed and hence the keys (and message signatures) to generate continuously changing parameters that are specific to the context (time and synchronized entity).

4.3 Using GeM2 and Butterfly1 in the Proposed Framework

We considered GeM2 and Butterfly1 as an initial attempt to explore the feasibility of our proposed framework. Although they are stand alone security proposals, with completely different rationales and mechanisms of operation, we explore this combination as a starting point to understand the behavior of the framework with such algorithms. This becomes a stepping stone towards exploring the the feasibility of using standard algorithms, Diffie-Hellman [9] for key generation, AES or DES for encryption and combinations thereof, in our framework.

In the algorithm choice logic (discussed in Section 3), the value of n will be set to 2 in the computation of r_{ac} (as defined in Equation (1)). The PRNG seed, $seed_{ac}$, would then depend on the ID of the entity (RFID tag, WBAN sensor, RFID server or WBAN hub), and t_0. The mechanism to update the timestamp t_0 is discussed as part of the protocol of operation. The protocol of operation is as follows for a communication of the i^{th} message between Alice and Bob (Figure 3):

Step 1: Alice retrieves the previously acknowledged and synchronized timestamp, t_0, and increments the stored value of r_{ac}, and computes the algorithm choice, C_a using Equations (1), (2) and (3).

Step 2: The algorithms are chosen as per Equation (4).

$$C_a = \begin{cases} 0 & \text{algorithm chosen} = \text{GeM2} \\ 1 & \text{algorithm chosen} = \text{Butterfly1} \end{cases} \quad (4)$$

Step 3: Alice computes the encryption keys and associated authentication parameters as explained in Sections 4.1 and 4.2 as specified by the chosen algorithm.

Step 4: Determining the length of the message (M_{TI}): In this implementation, the longest message transmitted is by Butterfly1. We consider this length as λ_n. The framework makes a decision on the transmitted message (MTX) based on Equation (5).

$$MTX = \begin{cases} M_{TI} & \text{if } \lambda_{M_{TI}} = \lambda_n \\ M_{TI} \parallel g(i) & \text{if } \lambda_{M_{TI}} < \lambda_n. \end{cases} \quad (5)$$

Here, $g(i)$ is a pseudorandom number generated using i as the seed (i being the sequence number in Butterfly1), only to generate numbers (that have no meaning for the algorithms) for padding the message to be transmitted so as to make it be of the same length as the longest message. In case messages in Butterfly1 (or GeM2) become longer than λ_n, they are broken into separate messages and transmitted using the same mechanism.

Step 5: On receiving the i^{th} message transmitted by Alice, Bob retrieves the timestamp, t_0 and performs the same computations as Alice to generate the keys and authentication parameters. The message integrity verification and entity authentication then proceeds as specified by the chosen algorithm.

Step 6: If Bob can authenticate Alice, Bob may respond with an encrypted acknowledgement/response as specified by the implementation, which could lead to session establishment or conclusion of a message transfer. On successful authentication, Alice and Bob have synchronized states.

Step 7: If Bob cannot authenticate Alice, Bob uses the internal counter to track of the number of erroneous messages or failed authentication attempts. This counter is reset on a successful communication, but Bob will flag the communication as erroneous if the counter reaches 3.

Step 8: Our framework also allows Alice and Bob to agree on the duration of each security association, or the duration for a single long-term communication session. The initial timestamp, t_0, is updated with the last synchronized timestamp at the end of such a security association. In this implementation, the security association period was set to be the choice of Butterfly1 algorithm. This means that whenever the framework chooses Butterfly1, t_0 is updated, i.e. $t_0 = t_i$. Therefore, when encrypting the current message, the processes to generate the key and other parameters would use the previously stored (and acknowledged) value of t_0, and update this value whenever the Butterfly1 algorithm is used. This ensures that there is no fixed duration of a security association, which adds another layer of uncertainty and consequently improves the security of the communication. However, if a system implementation requires a consistent duration for security association, the framework can be updated with minimal changes to set an internal timer that determines when the security association ends, thereby forcing a timestamp update.

Our framework is thus, able to utilize the best possible options from the available algorithms to ensure security. By retaining the length of the transmitted message to be a constant, our framework introduces an additional element of unpredictability to an observer.

5 Evaluation and Results

We evaluate our proposed metamorphic framework using three analyses. Our analyses are in the context of the use case discussed in Section 4, with GeM2 and Butterfly1 as constituent algorithms of the proposed framework. Although our framework proposal is generic, this gives us a way to provide some context to the assessment. Our evaluation of the framework involved three parts:

Key Sequence Evaluation: We implemented our framework (along with GeM2 and Butterfly1) using Java programming language to verify the working of the concept and generation of key sequences for further evaluation. We used the generated key sequences to test similarity between consecutive keys using Sörensen's Similarity Index (SSI) [37]. Using the results so obtained, we compared its performance with the RFID security proposals by Zhu et al. [48] and Dong et al. [10], and with an AES-based key generation approach for WBANs proposed by Liu et al. [22]. Note that we also implemented the proposals by Zhu et al., Dong et al., and Liu et al. using Java to generate key sequences for our assessment.

Hardware Complexity Evaluation: We estimated the approximate resource requirement for implementing the proposed framework (along with GeM2 and Butterfly1) encryption scheme on hardware.

Security Evaluation: We also performed a security assessment using Scyther protocol analyzer [6, 7] and qualitative security analysis to evaluate the security of our proposal.

5.1 Key Sequence Evaluation

We first implemented GeM2 and Butterfly1 using Java, followed by using these in the implementation of the framework. In our implementation of GeM2, the initial key was set to $92EB8D6ECF7F808A705D1A4566991AF0$, the initial seeds to compute the PRNG seed were set to 14930352 and 24157817. For Butterfly1, the PRNG used to choose the value of the variable j at random, which decides the state of the seed (s_j), was initialized to $192BC333250CCCFF$, while the seed (s) itself was initially set to 12345678. We used the Java Random class to introduce random delays (0 and 2 seconds) between consecutive key generations, as an attempt to emulate real-time communication, and used methods in the Random class to extract PRNG sequences. Furthermore, we used the Java method, $System.currentTimeMillis()$ to extract the timestamp. We extracted 10240 key sequences for our assessment.

A strong cryptosystem needs to have a key management mechanism that is strong and able to generate and refresh keys in a manner that can perplex adversaries, for

Table 1: Summary of similarity between keys

Configuration	Average SSI (SSI_{av})
Proposed framework* (K, K_i)	0.3437
GeM2* (K)	0.3040
Butterfly1* (K_i)	0.3809
Liu et al. [22]	0.3826
Zhu et al. [48]	0.4110
Dong et al. [10]	0.3815

$*$: Only keys in parentheses were considered for analysis.

a cryptosystem is accepted to be only as strong as the keys used (Kerchkoff's Principle [39]). This led us to verify how similar consecutive keys generated by each algorithm and the framework are, since GeM2 and Butterfly1 use varying logical operations to accomplish the desired functionality. To evaluate similarity between keys, we considered keys generated by the system and compare pairs of keys. We quantified the similarity between keys using SSI, which is a measure of how similar the various pairs of keys are, i.e. it is the ratio of twice the total similar characters in the two keys to the total size (in characters) of each key. Equation (6) summarizes the computation of SSI.

$$SSI = \frac{2 \times n(A \cap B)}{n(A)\ +\ n(B).} \tag{6}$$

Here, $n(A \cap B)$ represents the number of characters (or, numbers) in the key pair that are same, $n(A)$ and $n(B)$ represent the total number of characters (or, numbers) in each of the keys A and B of the key pair, respectively. The expectation is to have key similarity sufficiently low, with an average SSI value in the vicinity of 0.30, or at least less than 50%. This is to ensure that keys do not appear to have obvious similarity or patterns, which could be exploited by adversaries to derive keys.

Table 1 summarizes the average SSI values for the proposed framework, GeM2, Butterfly1, and the proposals by Liu et al. [22], Zhu et al. [48], and Dong et al. [10].

We can observe that our framework is able to generate keys that are less similar to each other, compared to the other algorithms, while GeM2 performs better than the framework.

One thing to note is that when used appropriately, the framework is able to combine the best attributes of the algorithms and contribute to making the system improve overall in terms of security. This can be observed with the slightly high similarity in keys generated by Butterfly1, which when combined with GeM2 in the framework, performs better. Furthermore, random choices of algorithms for each communication also ensures that the overall unpredictability remains high. This, in addition to low similarity between keys, improves the security. In our continuing work, we will work to ensure that the unpredictability of keys is not dependent on the best algorithm available in the framework, but that the framework itself will ensure high unpredictability of keys.

5.2 Hardware Complexity Evaluation

To test resource utilization, we developed a behavioral model of our framework using VHDL (Very High Speed Integrated Circuit - VHSIC - Hardware Description Language) [32] and deployed it on the Xilinx Spartan-6 FPGA (Field Programmable Gate Array) SP605 Embedded Kit [42]. Logical operations on an FPGA are accomplished sing configurable logic blocks (CLBs) and programmable interconnects. Spartan-6 has two slices, with each slice composed of "four logic-function generators (or look-up tables, LUTs) and eight storage elements" [43]. We implemented a modified version of the J3Gen PRNG (Melià-Seguí et al. [26]) and SHA-1 message digest algorithm (Rainier [31]). We used these implementations to realize the various functionalities to estimate resource consumption, and our proposal is generic, and designed to work with any encryption, PRNG and message digest algorithms.

Hardware resource estimation helps understand the overall resource utility when algorithms are implemented on application specific integrated circuits (ASICs). This is based on the logic block utilization (i.e. LUTs, Flip-Flops, etc.). As observed in Table 2, the resources consumed by our framework implementation are considerably low than available on the FPGA. The number of logic cells consumed can be used to determine the approximate logic gates for the implementation. A logic cell is a "logical equivalent of a classic four-input LUT and a Flip Flop" [44], resulting in 1.6 logic cells per LUT in Spartan-6 [43], or approximately 6.4 logic cells per slice. Using the formula, $1\ slice \approx 6.4\ logic\ cells$, we estimate the number of logic cells. Furthermore, it has been estimated that $1\ logic\ cell \approx 15\ ASIC\ gates$ [45]. Our framework requires approximately 960 ASIC gates for a key size of 128 bits, where as PRESENT [5] and Grain128 [1], which are also included in the cryptographic suite specifications as part of ISO/IEC 29167-1:2014 [16], require 1570 and 1857 gates, respectively (as reported in their respective publications, results summarized in Table 3). This lets us ascertain that the overhead of our proposals on resource-constrained devices or other devices will be considerably less as compared to other approaches. Furthermore, our gate count estimate is very much within the range of 200 - 3000 gates, which is suggested to be the available gates for security on resource-constrained devices [19, 27].

5.3 Security Evaluation

We performed a security assessment of the protocol of operation (Figure 3) using Scyther protocol analyser [6, 7]. The Scyther tool allows us to perform a formal security analysis of the communication protocol [7], verifying it using the Dolev and Yao adversary model [6, 28], which assumes perfect cryptography, abstract messages and that the adversary has full control over the network. In this section, we discuss the results obtained from this analysis (summarized in Table 4), and also elaborate on how these

Table 2: Logic resource utilization

Device Utilization Summary	
Slice Registers (available:54,576)	17
Slice LUTs (available:27,288)	19
Occupied Slices (available:6,822)	10
MUXCYs used (available:13,644)	16
LUT Flip Flop pairs (/available)	13 (/22)
Bonded IOBs (available:296)	5
BUFG/BUFGMUXs (available:16)	1
Approximate Logic Cell and ASIC Gate Equivalent	
Logic Cells (available:43,661)	64
ASIC Gate Equivalent	960

Table 3: Gate count estimates

HiveSec	PRESENT [5]	Grain128 [1]
960	1570	1857

Table 4: Evaluation of the proposed framework using Scyther

Claim	Result	
	Initiator	Responder
Claim: *Secret*		
Secret Parameters‡	$NAWB^*$	$NAWB$
Claim: *Alive*	$NAV^\dagger$	NAV
Claim: *Weakagree*	NAV	NAV
Claim: *Nisynch*	$NAWB$	$NAWB$
Claim: *Niagree*	$NAWB$	$NAWB$

$NAWB^*$: No attacks, within bounds
$NAV^\dagger$: No attacks, verified
‡: The following parameters were expected to remain secret — all keys in the algorithms timestamp, message signatures, authentication-synchronization vector, and intended message

results impacts how the framework performs with respect to standard security goals [38] — confidentiality, integrity, authentication, non-repudiation (by association) and forward / backward secrecy.

Claim-1: Secrecy. Parameters expected to be secret in each configuration remain a secret, including and primarily the key generation parameters and intended messages. This claim holds true since none of these parameters are exchanged. This further implies that the framework configurations ensure *confidentiality* of all parameters. Furthermore, with the communicated parameters also including message signatures, the entities are able to verify *message integrity*.

Claims-2 and 3: Alive and Weakagree. Scyther validates our claims that the entities are running the same configuration (Weakagree) and all previous message sessions have used the proposed scheme (Alive).

Claim-4: Non-injective Synchronization (Nisynch). Our claim that the initiator and responder states are synchronized in the framework is also verified. Synchronization is dependent on the internal states of the system, which requires that each algorithm states be synchronized, and entities are able to recognize any attempts of de-synchronization. This guarantees protection against *replay* and *de-synchronization* attacks.

Claim-5: Non-injective Agreement (Niagree). The initial deploy-time parameters, such as the initial seeds and timestamp, are never exchanged in the open. Each synchronized update of the system states imply that these parameters are automatically updated, at times through encrypted messages. Thus, the internal parameters that are essential in computing the key materials are always in agreement in both entities, as long as they are synchronized and authenticated. Scyther validates that the framework configurations are synchronized, and that they are in agreement.

Authentication and Non-repudiation. Both algorithms used in the framework facilitate authentication, using message signatures and the authentication synchronization vector. This ensures that the system states are synchronized (established by Scyther assessment) and that the entities are (mutually) authenticated. Furthermore, the use of pre-shared parameters and random choice of one of the available algorithms means that the sender of each message cannot deny that it was sent from that particular entity. Since keys are updated and potentially a different algorithm is chosen for each key generation/encryption, it means that the internal states of each entity (for each algorithm) are always updated to the latest (synchronized) version, as long as they are authenticated. This, in a naïve manner helps the framework accomplish non-repudiation by association[2].

Forward and Backward security. To an observer without the knowledge of the internal states, the framework as a whole appears as a black-box sequence generation engine. This means that it appears to be a sequence generator that generates various keys and other parameters required to encrypt and sign messages. Unless the observing entity has a knowledge of the internal states and the algorithms chosen, knowledge of a contiguous set of keys either from the past or in the future would not yield useful information about the future keys or previously used keys, respectively. This is primarily

[2]Non-repudiation by association implies accomplishing this goal by being associated with a backend server, which can be authenticated by another trusted entity using trusted third parties and digital signatures.

due to the dependency on timestamp in the choice of algorithms as well as updates to internal states of the framework. This re-configurable or metamorphic property of our framework not only provides high security, but guarantees forward and backward secrecy as well.

6 Concluding Remarks

Motivated by the need for security proposals that consume less resources for computation, while providing high security by means of increased unpredictability, our work proposed a metamorphic (or, reconfigurable) security framework. We draw inspiration for our work from the manner in which a chameleon changes its color in response to the color of its surroundings. Since resource-constrained devices would require circuits to be pre-defined at deploy time, our framework is based on using multiple security solutions that help in accomplishing reconfigurability in its operation. Our framework uses a synchronized value of the timestamp, an incrementing integer and the ID of the resource-constrained entity for choosing one of N available algorithms.

Unpredictability is an attribute that is perhaps central to our framework. This is an aspect that defines the security of our proposals, since security of a published cryptographic technique is dependent on the nature of the keys used. This is defined by how related keys are to preceding and subsequent keys. Our assessment (Section 5) helps us establish that the keys, although seemingly related in the individual algorithms, are made even more unpredictable by the presence of the framework. This added layer of uncertainty ascertains that it remains hard for unauthorized entities to crack the system keys, keeping the overall security high. Our assessment also supports our claim of the proposal being suitable for resource-constrained applications, although this will depend on the attributes of other algorithms that may be deployed as part of the framework in other configurations.

The presence of multiple algorithms in a framework also means that the effect of any sub-optimal performance by one algorithm in the framework can be mitigated by the presence of other algorithms. Our results support our claims of increased security through unpredictability, while requiring less resources for ASIC implementations of our algorithms and the framework.

With its ability to combine the attributes of the available algorithms and being able to choose one at random, our work removes the need for a separate algorithm agreement phase in communication. Security is critical in wireless communications, especially in resource-constrained wireless networks with their added restrictions. When configured appropriately, our framework presents each resource-constrained device with an opportunity to accomplish several security goals, including mutual authentication and non-repudiation by association.

Our work was motivated by a need to remove key exchange messages, while facilitating resource-constrained devices to be able to dynamically choose from a set of available algorithms for various aspects of security; considering that the presence of security suites is also being suggested by the updated standard specifications [16, 15]. Our approach not only facilitates multiple algorithms, but the facility to choose one based on the specific context of a message exchange. This ability of dynamically being able to choose algorithms is similar to the framework reconfiguring itself with each message. The reconfigurable behavior makes our framework *metamorphic*, giving an illusion as though the framework is changing its structure at random. In our continuing work, we will consider extending our framework to evaluate it with more (and diverse) algorithms and to perform a critical complexity evaluation of the framework. While our focus in this work has been to develop a framework for application in resource-constrained wireless networks, the design of the framework is generic, allowing it to be extended and adapted for use in other non-resource-constrained application environments as well.

Acknowledgments

This work has been funded by the Boeing Company. The authors gratefully acknowledge the support and the feedback given by the company.

References

[1] M. Ågren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: A new version of grain-128 with optional authentication," *International Journal of Wireless Mobile Computing*, vol. 5, pp. 48–59, Dec. 2011.

[2] I. Beretta, V. Rana, M. D. Santambrogio, and D. Sciuto, "On-line task management for a reconfigurable cryptographic architecture," in *IEEE International Symposium on Parallel Distributed Processing (IPDPS'09)*, pp. 1–4, May 2009.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *The Keccak Sponge Function Family*, July 15, 2016. (`http://keccak.noekeon.org/`)

[4] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 364–383, 1986.

[5] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems (CHES'07)*, LNCS 4727, pp. 450–466, Springer Berlin Heidelberg, 2007.

[6] Cas Cremers, *The Scyther Tool*, Apr. 4, 2014. (`https://www.cs.ox.ac.uk/people/cas.cremers/scyther/`)

[7] Cas Cremers, *Scyther User Manual*, Department of Computer Science, University of Oxford, 2014. (http://documents.mx/documents/scyther-manual.html)

[8] A. Dandalis and V. K. Prasanna, "An adaptive cryptographic engine for internet protocol security architectures," *ACM Transactions on Design Automation of Electronic Systems*, vol. 9, pp. 333–353, July 2004.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644–654, Nov. 1976.

[10] Q. Dong, J. Zhang, and L. Wei, "A SHA-3 based RFID mutual authentication protocol and its implementation," in *2013 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC'13)*, pp. 1–5, Aug. 2013.

[11] B. A. Forouzan, *Data Communications and Networking*, McGraw-HilI Forouzan Networking Series, 2007.

[12] B. Glover and H. Bhatt, *RFID Essentials*, O'Reilly Media, First edition, 2006.

[13] M. J. Hakeem, K. Raahemifar, and G. N. Khan, "A novel key management protocol for RFID systems," in *9th International Wireless Communications and Mobile Computing Conference (IWCMC'13)*, pp. 1107–1113, July 2013.

[14] IEEE Standards Association, "IEEE 802.15: Wireless Personal Area Networks (PANs),", 2005.

[15] IEEE Standards Association, '*IEEE Standard for Local and Metropolitan Area Networks - Part 15.6: Wireless Body Area Networks*, IEEE Standard 802.15.6-2012, pp. 15–172, 2012.

[16] ISO/IEC, *Information Technology – Automatic Identification and Data Capture Techniques – Part 1: Security Services for RFID Air Interfaces*, International Standard ISO/IEC 29167-1: 2014, pp. 10, Aug. 2014.

[17] A. F. Jaimes and F. R. de Sousa, "A taxonomy for learning, teaching, and assessing wireless body area networks," in *IEEE 7th Latin American Symposium on Circuits Systems (LASCAS'16)*, pp. 179–182, Feb. 2016.

[18] A. K. Jones, R. Hoare, S. Dontharaju, S. Tung, R. Sprang, J. Fazekas, J. T. Cain, and M. H. Mickle, "An automated, FPGA-based reconfigurable, low-power RFID tag," *Microprocess. Microsystems*, vol. 31, pp. 116–134, Mar. 2007.

[19] A. Juels and S. A. Weis, "Authenticating pervasive devices with human protocols,". in *Advances in Cryptology (CRYPTO'05)*, LNCS 3621, pp. 293–308, Springer, 2005.

[20] A. V. N. Krishna, A. H. Narayana, K. M. Vani, "Window method based cubic spline curve public key cryptography," *International Journal of Electronics and Information Engineering*, vol. 4, no. 2, pp. 94–102, 2016.

[21] C. Li, J. Zhou, Y. Jiang, C. Chen, Y. Xu, and Z. Luo, "A reconfigurable and scalable architecture for security coprocessor," in *5th IEEE Conference on Industrial Electronics and Applications (ICIEA'10)*, pp. 1826–1831, June 2010.

[22] J. Liu and K. S. Kwak, "Hybrid security mechanisms for wireless body area networks," in *Second International Conference on Ubiquitous and Future Networks (ICUFN'10)*, pp. 98–103, June 2010.

[23] J. Liu, Z. Zhang, X. Chen, and K. S. Kwak, "Certificateless remote anonymous authentication schemes for wireless body area networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 332–342, Feb. 2014.

[24] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, pp. 5:1–5:33, Mar. 2009.

[25] H. Martin, E. S. Millan, L. Entrena, P. P. Lopez, and J. C. H. Castro, "AKARI-X: A pseudorandom number generator for secure lightweight systems," in *IEEE 17th International On-Line Testing Symposium (IOLTS'11)*, pp. 228–233, 2011.

[26] J. Melia-Segui, J. Garcia-Alfaro, and J. Herrera-Joancomarti, "Multiple-polynomial LFSR based pseudorandom number generator for EPC gen2 RFID tags," in *37th Annual Conference on IEEE Industrial Electronics Society (IECON'11)*, pp. 3820–3825, 2011.

[27] M. O'Neill (nee McLoone), "Low-cost SHA-1 hash function architecture for RFID tags," in *Proceedings of the Workshop on RFID Security (RFIDsec'08)*, pp. 1–11, 2008.

[28] C. Paar and J. Pelzl, *Understanding Cryptography*, Springer Berlin Heidelberg, 2010.

[29] P. Peris-Lopez, E. S. Millan, Jan C. A. van der Lubbe, and L. A. Entrena, "Cryptographically secure pseudo-random bit generator for RFID tags," in *International Conference for Internet Technology and Secured Transactions (ICITST'10)*, pp. 1–6, 2010.

[30] D. Poulin, "A rough guide to quantum chaos,", 2006. (http://www.physique.usherbrooke.ca/poulin/utilisateur/files/enseignement/rgtqc.pdf)

[31] J. Rainier, "SHA-1 sequential implementation," *Github*, 2014.

[32] C. H. Roth, Jr., *Digital Systems Design Using VHDL*, PWS Publishing Company, 1998.

[33] H. Saini, "1-2 skip list approach for efficient security checks in wireless mesh networks," *International Journal of Electronics and Information Engineering*, vol. 1, no. 1, pp. 9–15, 2014.

[34] R. V. Sampangi and S. Sampalli, "RFID mutual authentication protocols based on gene mutation and transfer," *Journal of Communications Software and Systems*, vol. 9, pp. 44, Mar. 2013.

[35] R. V. Sampangi and S. Sampalli, "RFID encryption scheme featuring pseudorandom numbers and butterfly seed generation," in *22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM'14)*, pp. 128–132, Sept. 2014.

[36] Lu Shi, M. Li, S. Yu, and J. Yuan, "BANA: Body area network authentication exploiting channel characteristics," *IEEE Journal on Selected Areas in Communications*, vol. 31, pp. 1803–1816, Sept. 2013.

[37] T. Sorensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons," *Biologiske Skrifter Kongelige Danske Videnskabernes Selskab*, vol. 5, no. 4, pp. 1–34, 1957.

[38] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson Prentice Hall, 2010.

[39] W. Trappe and L. C. Washington, *Introduction to Cryptography with Coding Theory*, Pearson Prentice Hall, 2006.

[40] N. S. Voros, M. Hübner, J. Becker, M. Kühnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schüler, H. Sahlbach, S. Whitty, R. Ernst, E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein, and W. Putzke-Röming, "MORPHEUS: A heterogeneous dynamically reconfigurable platform for designing highly complex embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 12, pp. 70:1–70:33, Apr. 2013.

[41] E. W. Weisstein, "Linear recurrence equation," *MathWorld–A Wolfram Web Resource*, 2012.

[42] Xilinx, *Getting Started with the Spartan-6 FPGA SP605 Embedded Kit*, Xilinx Inc., June 2010.

[43] Xilinx, *Spartan-6 FPGA Configurable Logic Block: User Guide*, Xilinx Inc., Feb. 2010.

[44] Xilinx, *7 Series FPGA Configurable Logic Block: User Guide*, Xilinx Inc., Nov. 2014.

[45] Xilinx, "All programmable low-end portfolio product selection guide,", 2014.

[46] J. Yu, G. Khan, and F. Yuan, "XTEA encryption based novel RFID security protocol," in *24th Canadian Conference on Electrical and Computer Engineering (CCECE'11)*, pp. 58–62, May 2011.

[47] J. Zhou, Y. Xu, and X. Li, "Reconfigurable and scalable security module of active RFID for security-sensitive applications," in *The 2nd IEEE International Conference on Information Management and Engineering (ICIME'10)*, pp. 135–140, Apr. 2010.

[48] G. Zhu and G. N. Khan, "Symmetric key based RFID authentication protocol with a secure key-updating scheme," in *26th Annual IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'13)*, pp. 1–5, May 2013.

Biography

Raghav V. Sampangi Dr. Raghav V. Sampangi is a Postdoctoral Fellow at the Faculty of Computer Science, Dalhousie University, Canada. His research interests include security, privacy, and usability in Context-Aware Systems and the Internet of Things. He has worked on reconfigurable security in emerging wireless networks such as RFID and wireless body area networks (WBAN). Currently, he focuses on key generation and authentication in resource-constrained devices, and usable security in Context-Aware Systems and the Internet of Things.

Srinivas Sampalli Dr. Srinivas (Srini) Sampalli is a professor and 3M National Teaching Fellow in the Faculty of Computer Science, Dalhousie University, Halifax. His research is in emerging wireless technologies, especially in the intersection of smartphones, near field communications (NFC) and mobile cloud computing. He has investigated protocol vulnerabilities, security best practices, risk mitigation and analysis, design of intrusion detection and prevention systems, and applications in healthcare and mobile commerce. His projects have been sponsored by NSERC, Industry Canada and NRC. Dr. Sampalli has received many teaching awards at the Faculty, University, provincial and national levels, including a named teaching award and 3M National Teaching Fellowship, Canada's most prestigious teaching acknowledgment.