Tahir, Ruhma, Tahir, Shahzaib, McDonald-Maier, Klaus D., Howells, Gareth and Sajjad, Ali (2021) *A Novel ICMetric Public Key Framework for Secure Communication.* Journal of Network and Computer Applications, 195 . ISSN 1084-8045.

A Novel ICMetric Public Key Framework for Secure Communication

Ruhma Tahir[a], Shahzaib Tahir[b,*], Hasan Tahir[c], Klaus Mc-Donald-Maier[a], Gareth Howells[d], Ali Sajjad[e]

[a] *School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom,*

[b] *Department of Information Security, College of Signals, National University of Sciences and Technology, Islamabad, Pakistan, 46000,*

[c] *Department of Information Security, School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology, Islamabad, Pakistan,*

[d] *School of Engineering and Digital Arts, University of Kent, Canterbury, United Kingdom.,*

[e] *Research and Innovation, British Telecommunications, Adastral Park, Ipswich, United Kingdom,*

Abstract

The Integrated Circuit Metric (ICMetric) technology is a novel trust basis that uses the system features to create an identification of a device. The ICMetric of the device is used for the provision of security services, thereby addressing the issue of trust associated with device identity. The ICMetric technology can be adapted to function with varying environments; however, the short length and low entropy of the ICMetric key pose a major threat to applications based on ICMetric. This paper proposes a secure comprehensive ICMetric based architecture that facilitates asymmetric ICMetric applications for secure services in an end-to-end environment. This novel framework has been designed keeping in mind the construction principles of ICMetric thereby preventing threats that are prevalent in many security schemes. Finally, an empirical evaluation and feasibility has been presented by implementing the proposed framework and doing an extensive security analysis.

Keywords: Entropy, Key derivation function, Brute-force attacks, Shamir secret sharing, SHA-2, RSA.

[*] Corresponding author; email: shahzaib.tahir@mcs.edu.pk

1. Introduction

Digital devices are becoming increasingly ubiquitous, and this gradual shift towards pervasive computing [1] envisions many benefits in sectors as diverse as finance, entertainment, healthcare, information access, automotive etc. Embedded systems are composed of devices connected together through wireless communication links to achieve a goal [2][3]. However, the wireless communication links between the embedded system devices are vulnerable to being attacked by adversaries [4][5], that need to be addressed. Therefore, embedded systems security is a fundamental design requirement and is increasingly emerging. Many security services rely on stored keys for providing security in applications. The stored keys can be compromised through a variety of attacks many of which are based on side channel exploits. This highlights a need for the creation of a framework that assists in the generation of keys at runtime thereby providing improved security for embedded devices and the privacy to the humans interacting with them. The problems associated with key theft and stored keys [6] have paved way for design and development of security paradigms that focus on a novel root of trust, designed by using hardware and software features of a device.

ICMetric (Integrated Circuit Metrics) [7] technology has been developed to cater for the security vulnerabilities found in embedded system applications [8]. ICMetric technology identifies a system based on its various system level features[9]. The ICMetric serves as a secret key for the device and is based on hardware and software features of the device. The key is sufficiently unique, while the features are such that they provide distinguishability for similar devices using the same features. These features not only provide sufficient variance but also provide a reproducible key; additionally the features have to remain obscure to any unauthorized access. The generated ICMetric key exists only locally and is removed from memory. The ICMetric key is reproduced as required using hardware and software features of the device. ICMetric enables device verification with a very high accuracy, since the hardware/ software features of a system are directly being used in the generation cryptographic key being used by the system [8].

A strong key having high entropy and sufficient length is the basic requirement for the secure working of a cryptosystem. The use of an ICMetric in its original form is very challenging[10]. The ICMetric may have low entropy or insufficient length, thereby easily being guessed by the attacker leading to system compromise [11]. The generated ICMetric must have certain characteristics of having sufficient key entropy and length, to be securely used in applications for performing cryptographic operations. In this paper a strong ICMetric key generation protocol will be presented that can potentially improve the strength of the ICMetric for crytpographic applications. The proposed protocol for the generation of a strong ICMetric, results in the generation of secret key having required length and sufficient entropy. A major goal of the strong ICMetric key generation protocol is to secure the ICMetric from pre-computed and brute force attacks. This is a very critical requirement in ICMetrics since compromise of the ICMetric can potentially mean the compromise of the device and all subsequently the whole system. The ICMetric strong key generation protocol generates an ICMetric strong key based on the proposed two-tier key derivation function for the generation of strong ICMetric key from ICMetric data.

The importance of public key cryptosystems cannot be denied owing to the additional security advantages provided by public key cryptography. Therefore, designing a public key framework for ICMetric based applications is of utmost importance. The proposed ICM-RSA protocol is a step in this direction that aims to design and develop a protocol for ICMetric based entities thereby providing confidentiality and non-repudiation of data. The ICMetric RSA (ICM-RSA) protocol uses the ICMetric strong key to generate an ICMetric based public/private key pair for public key security applications.

1.1. Contributions

In this research we present a novel ICMetric public key framework to strengthen the use of ICMetric technology in various applications thereby making the following contributions:

- We present an ICMetric strong key generation protocol that generates ICMetric keys having an entropy very close to 8 bits per byte and required length. The proposed protocol safeguards the ICMetric key from brute force attacks and its two-tier architecture prevents the possibility of the original ICMetric being revealed to adversaries.

- We propose a modified ICM-RSA protocol that enables ICMetric based entities to perform secure one to one communication in a public key setting; keeping in mind the design principles of the ICMetric technology and security properties of RSA.

- We carry out security analysis of the proposed framework, thereby developing a proof of concept prototype to formally test the proposed framework.

1.2. Organization

The remainder of this paper is organized as follows; section 2 discusses the proposed architecture with the help of a scenario that helps formally present the threat model and thereby the design goals respectively. Section 3 discusses the ICMetric technology and its features elaborating on how ICMetric could be a viable solution. The major relevant literature is also concisely discussed in this section. In section 4 the ICMetric public key framework is formally proposed, tuning it in accordance to the proposed security definitions. Section 5 presents the security proof that helps validate the security of the proposed framework. The security analysis of our proposed framework is presented in section 6. The implementation details of our design with its performance analysis is presented in section 7. The conclusion and future work is presented in section 8.

2. Design of ICMetric-based Public Key Framework

This section discusses the proposed framework with the help of a scenario. The scenario leads to the threat model and the design goals respectively.

2.1. Proposed Architecture

Suppose Alice and Bob want to communicate with each other by sharing a message (m). They rely on a public key cryptosystem to securely transmit the message. Alice and Bob store their individual private keys on their machine and broadcast their public keys. This gives rise to the prevalent problem of key theft as the private keys need to be stored. ICMetric presents a possible solution to this problem [12]. Therefore Alice and Bob make use of the ICMetric technology that extracts the device features and generates the cryptographic keys at run time. This eliminates the possibility of key theft as the keys do not need to be stored in any form. Although these ICMetric keys can be used to communicate securely, they give rise to certain challenges that are narrated in the threat model below.

2.2. Threat Model

Cryptographic keys vary in strength and some are stronger than others. The weakness in cryptographic keys often exists due to a lack of sufficient entropy and length. The core principle is that cryptographic keys should be strong, have high entropy (not readily derivable) and be of sufficient length. Key generation/ derivation is normally done by incorporating a cryptographically secure pseudo random number generator. This ensures that the generated key has sufficient entropy and possesses qualities suitable for a cryptographic key. The ICMetric of a device is in no way a cryptographically secure primitive which raises the need for additional key strengthening mechanisms. A weak ICMetric key could make the underlying device open to pre-computed and brute force attacks, thereby defeating the purpose of the ICMetric technology. Following are the attacks/ problems commonly seen against key generation/ derivation schemes where hashing has been used as a leading primitive:

- Key brute force attack
- Rainbow table attacks
- Low key length
- Low key entropy
- ICMetric theft deterrence

Furthermore, the ICMetric is a fingerprint of the device and its compromise implies that the device can be compromised. However, the ICMetric of a device is created using unique device features that are unique, unspoofable but reproducible. In this case the greatest strength of the ICMetric technology becomes its greatest liability. Therefore should one set of ICMetric feature data be compromised, that one ICMetric feature set is compromised forever.

2.3. Design Goals

Given the threat model, the ICMetric public key framework focuses on the following security goals:

- Length and Entropy of the ICMetric Key - a fundamental goal of the ICMetric public key framework is to generate strong keys that have required length and 8 bits per byte of entropy to be safely used in security critical applications. The sufficient length of the key protects it from easily being compromised and high entropy ensures that the keys can't be easily compromised using pre-computed attacks.

- Deterring compromise of the device ICMetric - a crucial security goal of the designed ICMetric public key framework is deterring the effects of a possibly successful brute force attack on the original ICMetric. Therefore, the ICMetric should not be compromised during the working of the ICMetric public key framework.

- Data Confidentiality - a major goal of the proposed ICMetric public key framework is to provide confidentiality of data for ICMetric based entities. The proposed protocol uses an extended RSA encrypt/ decrypt algorithm for the provision of data secrecy between ICMetric based entities.

- Non-repudiation - a goal of the proposed ICMetric public key framework is non-repudiation of data. The key generation is based on the ICMetric of the entity, therefore the resulting cipher text can be traced back to the original source.

Definition (ICMetric-based Public Key Framework): The proposed FRSE comprises of five polynomial time algorithms $\Pi =$ (ICMKey, ICMStrongKey, KGenPK, Encpk, Dec) such that:

$ICMKey, MICM \leftarrow$ ICMGen(F_n): takes as input n device features F_n to generate a mini-ICMetric for each feature set and thereby an ICMetric for the entity.

$ICMStrongKey \leftarrow$ ICMStrongKeyGen($ICMKey, MICM$): takes as input the ICMetric and the mini-ICMetric's as pepper value to generate an ICMStrongKey with the required length and entropy.

$(sk, pk) \leftarrow$ KGenPK($ICMStrongKey$): generates a private/public key-pair (sk, pk) based on the ICMStrongKey.

$c \leftarrow$ Encpk(m): encrypts message m with private key sk and outputs ciphertext c.

$m \leftarrow Dec(c, sk)$: decrypts ciphertext c with secret key sk and outputs message m.

3. The ICMetric Technology and Existing Literature

The security of many cryptographic schemes lies in the secrecy of the keys. This is in line with the Kerckhoff security principle[13] which states that the secrecy of a system should lie in keeping the keys secret and not the algorithm. These keys are often stored on the system thus making them vulnerable. An ideal solution to this problem would be to entirely eliminate stored keys and be able to have them available when required. ICMetric is an important technology in this direction that aims to resolve the threats associated with the use of stored keys, by making use of system level characteristics to recalculate the cryptographic keys at runtime.

3.1. Working of ICMetric Technology

ICMetrics is a revolutionary new trusted computing approach that avoids storage of root-of-trust encryption keys by creating them on demand based on measurable properties and features of the desired device or system itself. ICMetrics is essentially the electronic' equivalent of biometrics, with the additional advantage of not needing to store unique templates for key creation.

The Template-Free technique employed by ICMetrics uses a novel normalisation and combination strategy to merge a series of metrics, which represent the properties, features and behavioural characteristics of the desired device to generate a secret vale termed ICMetric. Abstractly, the ICMetric architecture has three key stages as shown in Figure 1:

The ICMetrics technique involves generating and obtaining feature values from the operation of the device in question. The obtained metrics are selected and combined using advanced pattern recognition techniques. The primary purpose of ICMetrics is to encrypt components of devices, systems or services using properties or features derived from their own construction and behaviour to form a digital signature capable of assuring both their authenticity and freedom from malware whilst simultaneously operating within their designed specification and execute on an arbitrary platform. The ability to securely access data from a remote sensor by an online server offers significant advantages for ease of remote monitoring and data processing. A major novelty of the ICMetric system is that the measured characteristics

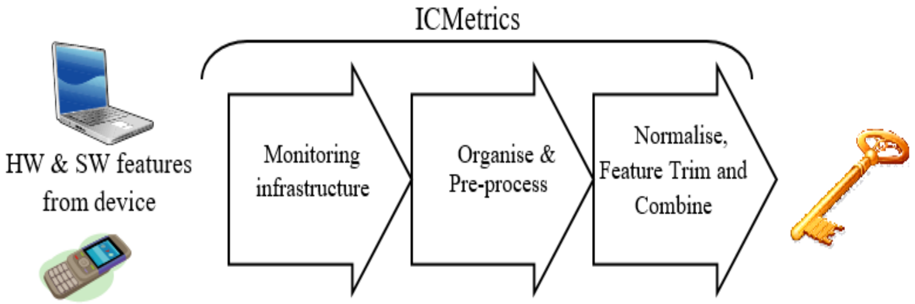
ICMetrics
HW & SW features from device
Monitoring infrastructure
Organise & Pre-process
Normalise, Feature Trim and Combine

Figure 1: Operational Phase of ICMetrics

need not remain absolutely constant but are free to vary within deduced parameters, thus allowing the device to operate in several states. ICMetrics creates identifiers for embedded devices enabling the generation of unique digital signatures and potential secure encrypted communication between services and sensor nodes.

The practical operation comprising of two parts is presented in this section:

3.1.1. Calibration Phase

(applied once)

a. For the device in question, the desired feature values are measured; that will typically be operating characteristics of the associated software.
b. Feature distributions are generated for each measured feature value illustrating the frequency of each occurrence.
c. The feature distributions are normalised thereby generating normalisation maps for each feature.

3.1.2. Operation Phase

(applied at the time of generation of encryption key)

a. Measure features for the given device for which digital signature (or an encryption key) is desired.
b. Apply the normalisation maps to generate values suitable for key generation.
c. Apply the key generation algorithm which combines the normalised feature values into a single key (termed the ICMetric)

It should be emphasised that the ICMetric system may be used in conjunction with a range of encryption techniques such as RSA, ECC, DSA and so on; and is independent of any particular encryption algorithm [7].

The ICMetric technology generates an ICMetric based on the software and hardware features of a device. The system characteristics used in the calculation of an ICMetric are not static and vary in a pre-determined fashion depending on the system characteristics and its interaction with the environment, thereby making it difficult for the attacker to deduce at any point in time. The features are individually processed by using algorithms for feature set establishment, feature extraction, feature value correlation analysis and change probability analysis to produce a device identification that can distinguish between two devices that have the same model, specifications, manufacturer and environment. To achieve this the technology uses a range of features such as communication addresses, CPU IDs, data in the RAM/ROM, network profiles, users content and locations specific information. The calculated ICMetric is stored temporarily and whenever a secure operation is to take place the ICMetric is recalculated based on the system characteristics of the device.

For a particular device, the same ICMetric can be calculated by measuring the individual features, applying the normalisation maps [14], [15] and combining these individual feature values to generate an ICMetric. Papoutsis *et al.* propose that the measured individual feature values can be combined by adding individual feature values [16][15]. The ICMetric is used as input in the key derivation algorithms to generate a secure strong ICMetric key.

An advantage of such a mechanism is that since the keys are not residing on the system then for an attacker there is nothing to steal or infiltrate. The ICMetric system will provide another secure layer upon the existing security infrastructure. This results in the development of a key theft proof system that can be authenticated and can resist impersonation attacks. Papoutsis in his extensive work [16][15] has carried out a detailed discussion on employing the ICMetric technology for the generation of encryption keys.

3.2. Existing Literature

Extensive research on the use of ICMetric technology Papoutsis *et al.* [16][15], propose a technique where they generate a key directly from the measurable properties of a given hardware device, thereby safeguarding devices from threats related to key storage.

In a very recent study [17], the researchers have shown that it is possible to create symmetric keys for group of devices using MEMS PUF. The research makes two contributions. The first outcome is the feasibility of the MEMS PUF for the creation of device keys. The second contribution of the research is a Diffie Hellman inspired symmetric key for the group that is created by taking contributions from the group members. The proposed scheme suffers from denial of service attack and the regeneration of keys for changing group members becomes a constant problem. Moreover the scheme creates a symmetric key which although secure does not provide the benefits offered by asymmetric keys.

A patent on secure group communications through ICMetric [18] shows the establishment of a group ring key by using asymmetric keys. A prerequisite for the proposed system is the availability of asymmetric keys.

In a study [19], the qualities and characteristics of a device ICMetric have been explored and a scheme presented. The research introduces a security scheme for wearable devices based on the ICMetric technology. Wearable devices are able to provide a wide range of services as they are embedded with sensors that detect movement while worn on the body. The ICMetric scheme presented uses the bias in a MEMS accelerometer and gyroscope as a device fingerprint to create a device ICMetric thereby extracting the cryptographic key. The authors have shown that the ICMetric of a device can be used to extract varying key sizes and then made to function with multiple confidentiality schemes. The authors in [19][12] claim that the generation of encryption keys requires developing suitable methods for combining selected feature values to produce a unique ICMetric. In [14], two alternative techniques have been designed namely feature addition and concatenation. The authors claim that although with the feature concatenation technique a bigger encryption key in length is produced, this key is less stable compared to one produced with addition technique. Moreover, with feature combination using addition, a lower number of samples are required to produce the same ICMetric for the ICMetric device compared to applying the concatenation combination technique.

Conventional cryptography is based on using stored keys for the provision of security. Cryptographic algorithms rely on algorithmic intractability for the creation of a cryptographic primitive. While this method has proven sufficiently secure and is the basis for cryptographic algorithms today; it does not protect against side channel attacks [20]. As adversaries become increasingly well equipped to target cryptosystems, a renewed approach to help protect them is required.

Physically Unclonable Function (PUF) provide a novel basis upon which cryptographic services can be provisioned. A PUF is a one way function that takes an input and provides an output that is a representative of a unique characteristic of the device [21]. The device characteristics suitable for a PUF are unique, unpredictable, stable and repeatable. These device characteristics are introduced by a number of factors that cannot be recreated. For instance, soldering of MEMS sensors onto the mainboard introduces stresses that cause a bias in the sensor readings which has proven to be an identifying characteristic [22]. Similar characteristics have been studied and proven successful in studies related to SRAM [23]. Research has shown that PUF's can be used to create stronger security primitives leading to provision of improved security that is resistant to many issues faced by conventional cryptographic systems [24].

The work presented in this paper is an extension of ICMetric, whereby a public key generation protocol is designed that generates keys pairs for devices based on ICMetric. The proposed protocol aims to increase the entropy of generated ICMetric key, thereby generating public key pairs useful for embedded system applications.

ICMetric is a patented technology and most research has been done to look into suitable features which can be used to support the establishment of a device ICMetric. The research in [18] requires ICMetric based asymmetric keys for the establishment of the ICMetric group ring keys. No existing research addresses this thus forming the research gap and a motivation of this research. Thus this research studies the design of a two-tier ICMetric asymmetric key generation scheme which can be used for security provisions or possibly adapted with other systems that are based on public key cryptosystems.

4. The ICMetric Public Key Framework

As discussed in the definition presented in Section 2, the proposed framework comprises of five polynomial time algorithms. This section presents each of the phases/algorithm in details. The ICMetric Public Key Framework is comprised of two protocols namely the ICMetric strong key generation protocol that is responsible for generating a strong ICMetric key and an asymmetric key protocol based on an extended RSA algorithm that has been tuned in accordance with the design principles of ICMetrics.

4.1. ICMetric Strong Key Generation

The proposed strong key generation protocol aims to improve the security of the ICMetric based cryptosystem by proposing a protocol that generates a strong ICMetric key for use with symmetric or asymmetric cryptosystems. The proposed protocol is an effort to generate high entropy ICMetric keys of sufficient length which can be used for secure cryptographic operations in various applications. Secrecy of the ICMetric is of utmost importance, since compromise of the ICMetric will result in compromise of the whole device for any future operations. Therefore, the proposed protocol has been developed following this critical requirement and the design principles of the ICMetric technology.

The proposed protocol is comprised of two main phases; namely the sub-key generation phase and the strong key generation phase. Figure 2 shows a general model of the proposed ICMetric strong key generation protocol, depicting a generalization of the components that become part of the protocol design.

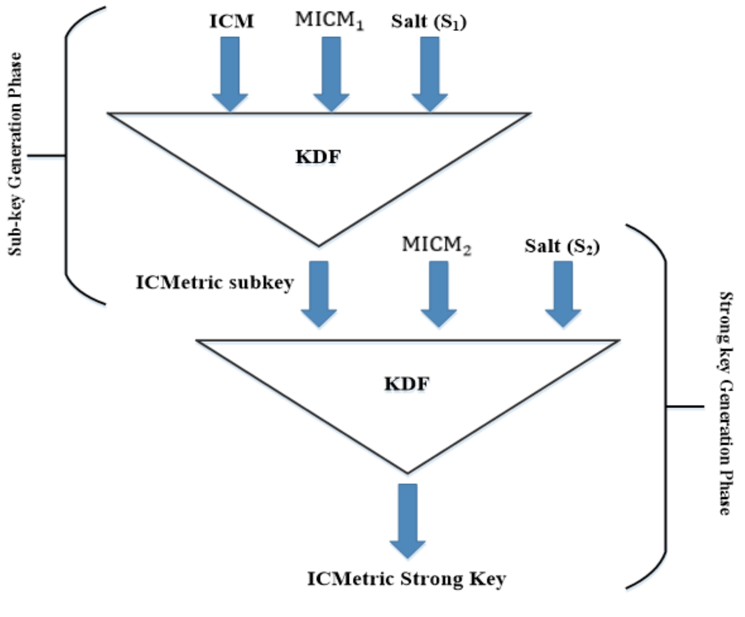

Figure 2: General Model of the ICMetric Strong Key Generation Protocol

Both phases, the sub-key generation phase and the strong key generation phase are interlinked where the output of the sub-key generation phase is used as input to the strong key generation phase, thereby generating a strong ICMetric key. The device ICMetric, a mini-ICMetric $MICM_1$ and a random salt S_1 become input to the chosen key derivation function (KDF) and generate an ICMetric subkey, which is the output of the sub-key generation phase. In the next phase, the ICMetric sub-key, a second mini-ICMetric $MICM_2$ and a random salt S_2 become input to the chosen key derivation function (KDF) and generate the final ICMetric strong key. The two mini-ICMetric values are used as cryptographic peppers and prevent the possibility of brute force attacks. They also enable the possibility of multiple strong derived keys that can be used for various secure cryptographic operations in the ICMetric application. The concept behind the two-tier ICMetric strong key generation approach is to safeguard the device ICMetric from being captured by an adversary, which could in effect compromise the device for any future operations.

The following section details each step involved in the working of the ICMetric strong key generation protocol.

4.1.1. Key Setup

The first phase of the protocol requires the device to generate an ICMetric based on the extracted feature values

$$ICM = ICMetric\, generated\, using\, features\, of\, Device \tag{1}$$

Device features are the foundation of creating an ICMetric for the device. There are certain aspects of each feature that are considered to determine their adequacy. Features are thoroughly examined to ensure they have high inter-sample variation and low intra-sample variation. Features showing promise have possible values that can be clustered for a device separate from another devices' grouped values. These enable finding small clusters of values that are far enough apart from another device's cluster of values. These features can include low-level hardware features that measure a device's processing capabilities such as execution time and memory to the software features of a device. These offer more scope for identifying devices and an undefined range of values are harder to spoof than a static value. Dynamic features also offer the ability to compare how values can change during operation [7]. Correlations offer a measurement of how values can change in relation to how another feature's value changes. Two devices can have feature values fall between a similar range, one device's values can rise with an increase in another feature's value while the other devices values can drop, while the range is similar the behaviour is different. This can be reflected in one device having a positive correlation between two features and other device exhibiting a negative correlation between same two features. These features having high correlations are combined together and these multidimensional correlations are analyzed to find the boundaries of distinct clusters of values for these features. The selected features of a device are logically categorized into specific sets called mini-ICMetrics. Each set contains features which share similar traits or are affected by the same modifications of a device. The creation of mini-ICMetric feature sets allows for fault tolerance system to be implemented into the ICMetric key generation process [25] [7].

The fault tolerance is achieved by using Shamir's secret sharing [26] to allow a fixed minimum number of mini-ICMetrics to be required to produce the same key. If the tolerance number of sets is not reached, the system fails, as it does not meet the secret sharing reconstruction threshold and a different key will be produced.

The advantage of this secret sharing process is that the ICMetric key is not stored on the system and the only values that are stored are one half of each coordinate that is necessary to generate the polynomial that produces the key. The halves stored on the system cannot be used to find out the polynomial that was used to generate them, which is a proven characteristic of the secret sharing process. Each mini-ICMetric is also discarded and half of the stored value specific to each set cannot be used to recover the original mini-ICMetric [27].

4.1.2. Parameter Setup

This phase of the framework is responsible for setting up various input parameters for the working of the strong ICMetric key generation protocol. Table I, lists the parameters that are required in the setup of the protocol.

Salting is the inclusion of a random value in the password hashing process that greatly decreases the likelihood of identical keys returning the same hash. If two keys are identical, salting can make it highly unlikely that their resulting hashes are the same. Detailed specifications [28][29] suggests that the salt value should be at least 64 bits long. This makes collisions (i.e. occasions that two stored passwords use the same salt) unlikely [30]. The random salts in the proposed architecture are generated using timestamps as the seed to the random number generator. As shown in table 1, the salt values S_1 and S_2 employed by the proposed ICMetric strong key generation protocol

Table 1: Description of Symbols/ Input Parameters in the Protocol

Symbol	Description
ICM	ICMetric of device
$\|\|$	Concatenation
$\oplus$	bitwise exclusive OR
S_1, S_2	128 bits random salt values
$MICM_1$, $MICM_2$	Mini-ICMetric pepper values
$SHA2(256/512)$	The selected SHA2 variant
N	Number of iterations
L	Required length of the strong key: (256/512/1024/2048 bits)
S	ICMetric sub-key
M	ICMetric strong key

are 128 bits long. A pepper performs a similar role to a salt, however unlike a salt the pepper value has to be kept in a place to prevent it from being obtained by the attacker in case of a breach. Therefore the pepper value in the proposed scheme is selected randomly from the set of possible mini-ICMetric values. The mini ICMetric value does not need to be stored and discarded after use, thereby safeguarding it from possible compromise. SHA-2 is the hash function used for the purpose of key stretching, and the SHA2 variant of choice can be selected based on the required length of the generated strong key as shown in table 1. Therefore, the required length $'L'$ of the strong key will help in the selection of the exact SHA2 variant. The recommended SHA-2 iterations for a key stretching function recommended by NIST guidelines are “as high as can be tolerated while still allowing acceptable server performance”.

The two-tier mechanism is critical in hindering the attacker’s ability of ever getting hold of the raw ICMetric number. At no point can a system afford to lose its ICMetric and result in compromise of the fingerprint of the device. The computations required by the two-tier mechanism for generating the derived ICMetric key increases with the number of iterations [31]. There is obviously a trade-off, larger iteration counts make the systems more secure but at the same time hurt performance. The number of iterations should be set as high as tolerable for an environment/user; ranging from constrained systems where at least 1000 iterations are recommended, to very powerful systems where 10,000,000 iterations are appropriate. In many cases ICMetric key derivation is only done once for a long session, hence not being detrimental for the efficiency of the ICMetric based application.

4.1.3. Generation of Iterated ICMetric Hash

This step of the algorithm is responsible for performing N iterations on the device ICMetric by using the selected SHA2-variant, a mini-ICMetric value $MICM_1$ and a salt value S_1. The number of iterations in the proposed strong ICMetric key generation protocol make the key derivation function intentionally slow to compute and, can be adjusted to control the slowness. The strong ICMetric key generation process begins with the device's ICMetric, a mini-ICMetric value and a 128 bits random salt S_1 passed through the selected SHA2-variant function. In the second round, the result from the first round of the SHA2-variant function and the ICMetric becomes the output of the second iteration. The SHA2-variant function is iterated 10,000 times to generate a strong ICMetric key for secure onward operations. Both the mini-ICMetric, 128 bits salt and ICM are combined via SHA-2, as shown in the set of equations (2). The purpose of the SHA-2 based key stretching and derivation algorithm is to combine and thus stretch the key, so that it qualifies for use in secure operations. The 128 bits random number serves as a salt value and the mini-ICMetric are used for the key stretching function and safeguards against rainbow table attacks. By adding a salt and mini-ICMetric pepper value to the ICM, the possibility of using pre-computed hashes for attacks is reduced.

The purpose of the salt and pepper is to allow the generation of a large set of keys corresponding to each password, for a fixed iteration count. Therefore, using a salt makes it difficult for the attacker to generate a table of resulting keys, for even a small subset of the most-likely passwords. If I_i is the result of a single iteration where $i \in N$ then

$$\begin{aligned} I_1 &= SHA2(ICM, MICM_1, S_1) \\ I_2 &= SHA2(ICM, I_1) \\ &\vdots \\ I_{100,000} &= SHA2(ICM, I_{99,999}) \end{aligned} \tag{2}$$

Finally, the result of each SHA2-variant function is XORed as follows:

$$X_i = I_1 \oplus I_2 \oplus \cdots \oplus I_N \tag{3}$$

This gives the iterated ICMetric hash X_i based on the selected SHA2 variant. For SHA2(256), the final hashed output block size is 256 bits, whereas in SHA2(512) the final hashed output block size is 512 bits. Exclusive OR adds an extra layer of protection and the iterations of the hashing function and the salt value add security.

Figure 3 shows a blocked diagram of how the generation of the iterated ICMetric hash is connected to the rest of the sub-components in the generation of the ICMetric sub-key. It is evident from the figure that the ICMetric and the salt (S_1), become input to the 'Iterated ICMetric hash generation' module. The generated iterated ICMetric hashes become input to the 'ICMetric sub-key generation' module, which after performing computations on the input iterated ICMetric hashes, generates an ICMetric sub-key $'S'$, as shown in figure 3.

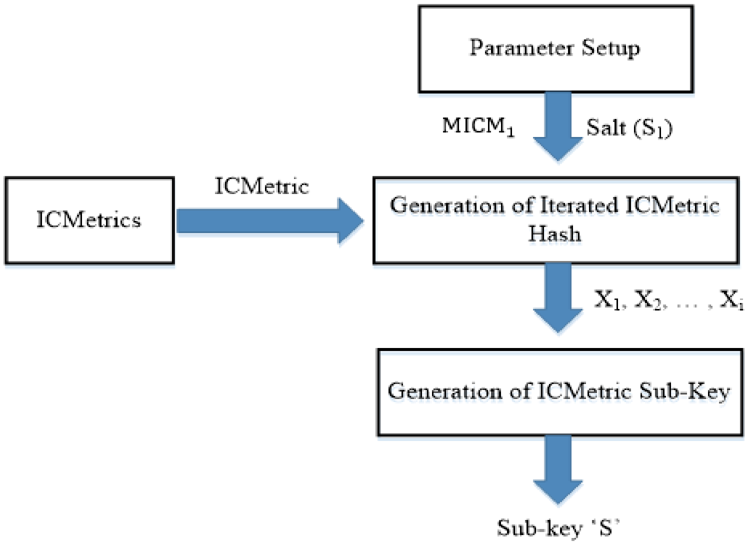

Figure 3: ICMetric Sub-Key Generation

4.1.4. Generation of ICMetric Sub-key

This section gives details of the operations in the ICMetric sub-key generation block shown in figure 3. In this phase, each generated iterated ICMetric hash is concatenated to generate a strong ICMetric key of required length $'L'$ that can be used for further cryptographic operations.

For SHA2(256), the hashed block size is 256 bits. Therefore, if the required length of the strong ICMetric key is 1024 bits, then there should be 1024/256=4 blocks concatenated to produce the strong ICMetric key $'S'$.

$$S = X_1 \| X_2 \| \cdots \| X_{L_{/256}} \tag{4}$$

For SHA2(512), the hashed block size is 512 bits. Therefore, if the required length of the generated strong ICMetric key is 1024 bits, then there should be 1024/512=2 blocks concatenated to produce the strong sub key ‘S'.

$$S = X_1 \| X_2 \| \cdots \| X_{L_{/512}} \tag{5}$$

The ICMetric strong key generation protocol can derive strong ICMetric keys of specified length.

4.1.5. Generation of Iterated Sub-Key Hash

This step of the algorithm does a stretching operation similar to the iterated hash function outlined in (3) above, however the input parameters are different. The sub-key generated in (4), the mini-ICMetric $MICM_2$ and salt S_2 are fed as input to the N iterations of the selected SHA2 variant. In the second round, the result from the first round of the SHA2-variant function and the sub-key $'S'$ becomes the output of the second iteration. In this step, 100,000 iterations are carried out, resulting in hashed values $H_1, H_2, \ldots, H_{100,000}$ as outlined in the set of equations (6).

$$\begin{aligned} H_1 = SHA2(S, MICM_2, S_2) \\ H_2 = SHA2(S, I_1) \\ \vdots \\ H_{100,000} = SHA2(S, I_{99,999}) \end{aligned} \tag{6}$$

Finally, the result of each SHA2-variant function is XORed as follows:

$$SX_i = H_1 \oplus H_2 \oplus \cdots \oplus H_N \tag{7}$$

the iterated sub key hash SX_i based on the selected SHA2 variant. For SHA2(256), the final hashed output block size is 256 bits, whereas in SHA2(512) the final hashed output block size is 512 bits.

Figure 4 shows a blocked diagram of how the generation of the iterated sub-key hash is connected to rest of the sub-components in the generation of the ICMetric sub-key. It is evident from the figure that the ICMetric sub-key, the mini-ICMetric ($MICM_2$) and the salt (S_2) become input to the 'Iterated sub-key hash generation' module. The generated iterated sub-key hashes become input to the 'ICMetric strong key generation' module, which after performing computations on the input iterated sub-key hashes, generates an ICMetric strong key 'M', as shown in figure 4.

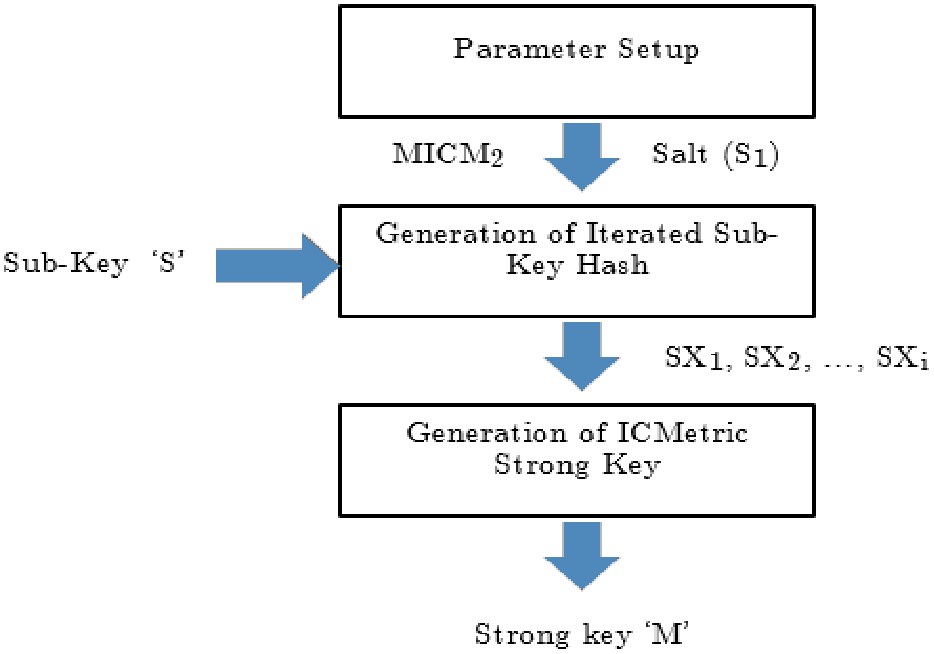

Figure 4: ICMetric Strong Key Generation

4.1.6. Generation of Strong ICMetric Key

This section gives details of the operations in the ICMetric strong key generation block shown in figure 4. In this phase, each generated iterated ICMetric hash is concatenated to create a strong ICMetric key of required length $'L'$ that can be used for further cryptographic operations.

For SHA2(256), the hashed block size is 256 bits. Therefore, if the required length of the strong ICMetric key is 1024 bits, then there should be 1024/256=4 blocks concatenated to produce the strong ICMetric key $'S'$.

$$M = SX_1 \| SX_2 \| \cdots \| SX_{L/256} \tag{8}$$

For SHA2(512), the hashed block size is 512 bits. Therefore, if the required length of the generated strong ICMetric key is 1024 bits, then there should be 1024/512=2 blocks concatenated to produce the strong sub key 'S'.

$$M = SX_1 \| SX_2 \| \cdots \| SX_{L_{/512}} \tag{9}$$

The ICMetric strong key generation protocol can derive strong ICMetric keys of required length. It generates as many blocks as required to achieve the required key length. Each block is produced by iterating the key stretching function 100,000 times.

Once generated, all blocks are concatenated to create the required sized key, which gives the strong ICMetric key that has sufficient length and entropy.

4.2. The ICMetric Asymmetric Key Protocol

The proposed ICM-RSA cryptosystem details an ICMetric centred asymmetric key protocol that can be used with asymmetric key applications based on the ICMetric technology. The proposed protocol is an idea to improve the security of the ICMetric based applications using ICMetric based public/private key pairs for achieving confidentiality. The following section details the working of the ICM-RSA protocol.

4.2.1. Key Generation

The keys for the ICM-RSA algorithm are generated the following way:

a. Employ the generated strong ICMetric key as the private key d'.

b. Choose two large prime numbers p and q such that d' is co-prime with $\varphi(n)$, where $n = pq$.
   - Check if d' is co-prime with $\varphi(n)$, else set the Offset d' such that $d = f(d', offset)$ and gcd(d,$\varphi(n)$)=1 where function f may be XOR or + operation

c. Determine e as $e \equiv d^{-1}(mod\,\varphi(n))$; i.e., e is the modular multiplicative inverse of $d(mod\,\varphi(n))$.
   - e is released as the public key.
   - $d\,d', p, q, \varphi(n)$ are discarded.
   - f and the offset are retained locally.

   The pair (n, e) is the public key.

4.2.2. Encryption

The message m is represented as an integer in the interval $[0 \cdots n-1]$ and the public key (n, e) of the entity is used to compute

$$c = m^e \, mod \, n$$

The ciphertext c is sent to the receiver.

4.2.3. Decryption

To recover the message m from the ciphertext c, the entity can recover message as

$$m = c^d \, mod \, n$$

5. Security Proof

This section presents the corollaries and lemmas that help validate the security of the proposed framework.

Corollary 1 – *The proposed framework is secure considering the ICMetric is strong and the ICM-RSA is secure.*

The proof to this corollary is dependent upon the following two lemmas:

Lemma 1: *The security of the proposed framework is dependent upon the strength of the ICMetric derived key.*

Lemma 2: *The proposed ICM-RSA does not violate the hard assumption/intractability of the original RSA.*

Proof of lemma 1:

To test the proposed protocol, the key generation protocol is partially based on existing cryptographically secure pseudorandom number generators, i.e., random number generators which are deterministic but nonetheless indistinguishable from truly random sequences, and then they are analysed whether the keys generated through the designed protocol have strong or weak entropy. Therefore, the goal of this section is to assess the entropy of the generated ICMetric keys based on the proposed strong key derivation function. The entropy analysis helps decide whether an ICMetric key is of sufficient length and can safely be used in various cryptographic applications to perform security critical operations.

In order to evaluate the cryptographic strength of the keys generated through the ICMetric based key generation method, this section compares the entropy measurements of the designed protocol against two well-known cryptographically secure key generation schemes that are widely used in modern information security systems. The first one is a standardized key generating algorithm based on 'SHA1PRNG' [32], which is a pseudorandom number generation (PRNG) algorithm based on SHA-1 cryptographic hash function and comes as a built-in default in a lot of programming languages e.g., C++ and Java etc. The second one is /dev/random, a blocking pseudorandom number generator available in most Unix-like operating systems e.g., Linux, FreeBSD, OpenBSD, macOS and iOS etc. It allows access to environmental noise collected from device drivers and other sources to include a measure of true randomness in its working, however, not all operating systems implement the same semantics for /dev/random [33].

The testing of the proposed design involves generating key of lengths 128, 256 and 512 bits respectively; using the three key generation methods and then measuring the entropy of the information content of these keys. The entropy measurement code treats the input key as a stream of 8-bits bytes and the reported statistics reflect this property of the bits-stream. The detailed results for the three key lengths are given in Figure 5, that show the entropy of each generated strong ICMetric key as a result of the rivalling key generation methods.

It is clear from the above Figure 5, the measured entropy of the keys generated from the three methods increases steadily as the size of the keys increases. However, it is still quite short of approaching the perfect entropy score of 8 bits per byte. The measured entropy of all three methods for all the different key sizes stays very close to each other, i.e., around 4 bits per byte for 128 bits keys, around 4.8 bits per byte for 256 bits keys and around 5.8 bits per byte for 512 bits keys. Therefore, the ICMetric based key generation protocol is faring as well as some other common and widely used cryptographically secure key generation protocols.

Furthermore, the suspected reason for these three schemes falling short of approaching the ideal Shannon Entropy was that, the amount of underlying information content being measured for entropy calculation is far too less. Therefore, the experiment was performed again, but this time the key generation schemes were modified to produce longer bits-streams. In other words, the inputs to the entropy measurement calculations were increased to 100 keys; of lengths 128, 256 and 512 bits respectively using the three key generation methods. Therefore, the entropy measurement code treats the input key as a stream of 12800, 25600 and 51200 bits respectively. The results of this experiment are given in Figure 6.

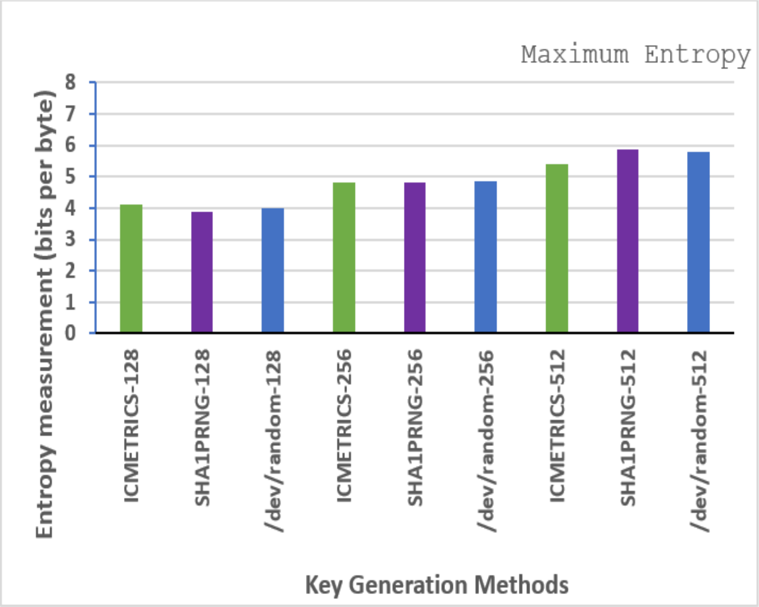

Figure 5: Entropy Measurements for Strong Key Variants

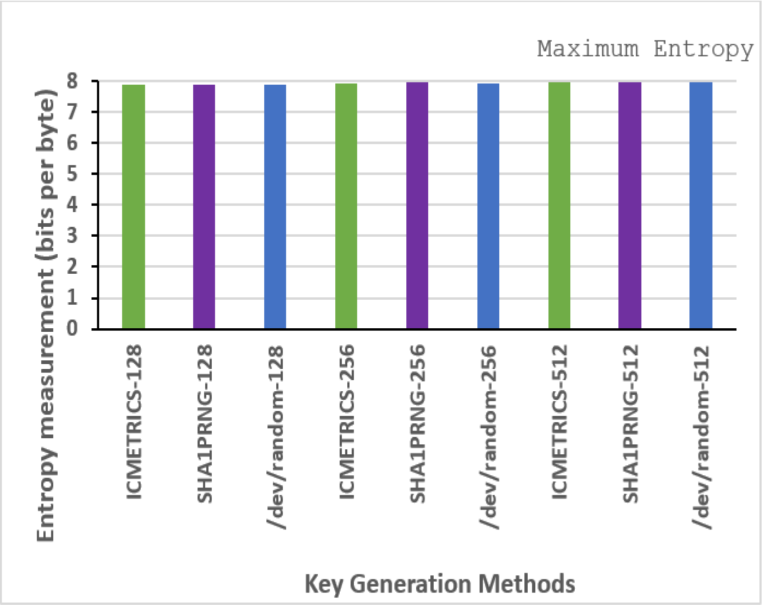

Figure 6: Entropy Measurements of Strong Key Variants based on Larger Inputs

As is clear from the above Figure 6, now the results of the measured entropy are almost equal to the theoretical maximum entropy of a uniformly distributed random key generation system. The number of 100 keys were chosen arbitrarily, however upon further experiments, the measured entropy stays very close to 8 bits per byte for even huge number of keys, although never exactly equal to 8.

Proof of lemma 2:

Corollary 2 - *The security of RSA depends on the hardness of factoring n=pq, i.e., the prime factors p and q cannot be determined within polynomial time and therefore becomes a hard problem.*

Claim - The security of the proposed ICM-RSA is based on the following assumptions:

- Having $\varphi(n)$ secure disables factoring n.
- Keeping d secure disallows factoring n.

To prove the security of ICM-RSA, we firstly prove that the above assumptions are strictly met in the proposed protocol. For this, we firstly assume that $\varphi(n)$ is known, so how can it lead to solving the integer factorization problem [34]. Solving the corresponding equations (b) and (c) may reveal the value of p and q respectively. Therefore, to solve the factorization problem, one possibility is to know the $\varphi(n)$ which contradicts our assumption because ICM-RSA is based on the strict assumption of $\varphi(n)$ being kept secure by the entity and not being revealed to an adversary.

Now, we assume that d is known to an adversary. To show how it leads to the solving the factorization problem we refer readers to [34][35]. Which is a clear contradiction of our assumption as d is based on the ICMetric of the device whose property is that it is kept always secret and regenerated when required. Therefore, d is also not revealed to an adversary. This leads to the conclusion that the proposed ICM-RSA does not weaken the security of the primary RSA protocol and hence ICM-RSA is secure.

Based on the proofs of the lemma 1 and lemma 2, it is evident that the proposed framework is secure.

6. Scheme Evaluation Against Threat Landscape

In this section, we perform a security analysis of the designed ICMetric public key framework against the threat model and design goals mentioned

Table 2: ICMetric Public Key Framework Evaluation

Attack Vectors	ICMetric Strong Key Protocol	ICM-RSA
Brute-force Attacks	✓	
Rainbow table Attacks	✓	
Length	✓	
Entropy	✓	
Deters ICMetric Capture	✓	
Confidentiality		✓
Adaptability	✓	✓
Non-repudiation		✓

in section 2.2 and 2.3 respectively. Particular focus has been on the security goals accomplished via the proposed scheme. In table II, a summary overview of the goals accomplished/ unaccomplished by the designed ICMetric Public key framework are presented.

6.1. Length of the Strong Key

The originally generated device ICMetric has inherent weaknesses like insufficient length and entropy, owing to which it cannot serve as a key for cryptographic operations [10]. The strong ICMetric key generation protocol makes use of key derivation functions on the device ICMetric coupled with the mini ICMetric as a pepper and a random salt, to generate strong ICMetric key variants of sufficient length that can serve as cryptographic keys in various cryptographic applications. The proposed ICMetric strong key generation protocol generates keys that have sufficient entropy to be securely used in cryptographic applications. The inclusion of two pepper values and random 128 bits salt values to the key derivation function, at two stages of the protocol adds entropy to the ICMetric and generates a longer stretched key having sufficient entropy. The large iteration count repeats the hash operation over the ICMetric multiple times to produce a strong key that has more entropy. This makes the weak ICMetric key suitable for use in various cryptographic applications requiring security.

6.2. Safety against ICMetric Compromise

The ICMetric technology is an innovative concept, designed to deter key theft. The proposed strong ICMetric key generation protocol also deters all forms of key capturing. The proposed protocol works in accordance with the design principles of the ICMetric technology and at no point does it disclose the device ICMetric to an adversary. When the keys are no longer required both the ICMetric and the cryptographic key are discarded hence deterring all forms of key capturing. This implies that at no point does the system rely on stored keys/ credentials to deliver cryptographic services. To further safeguard the ICMetric from being compromised, the proposed two tier architecture of the strong ICMetric key generation protocol is able to hinder an attacker's ability to get hold of the ICMetric at any point of the working of the protocol.

6.3. Mitigation of Pre-Computed Attacks

The proposed strong ICMetric key generation protocol makes use of a two tier approach, where it generates an internal ICMetric sub-key in the first phase and a strong ICMetric key in the second phase based on the device ICMetric. Two pepper and salt values are associated with the hashing operation at two tiers of the proposed protocol, for the generation of a longer derived ICMetric key, and several key derivation function rounds make it more time consuming to crack the strong ICMetric key. The pepper values are based on the generated mini-ICMetrics that are discarded after use in the key derivation function. Therefore the attacker is never able to capture the pepper value and launch a pre-computed attack on the key. The salt values add an additional layer to the ICMetric, thereby preventing the possiblity of brute force or rainbow table attack on the strong ICMetric. The strong ICMetric key generation protocol produces a stretched high entropy key bringing an increase in both the entropy and length of the key thus safeguarding against pre-computed attacks.

The designed strong ICMetric key generation protocol is based on hash function computations. A major problem associated with schemes making use of hashing is the possibility of pre-computed attacks [36]. These pre-computed attacks were of major concern for the proposed protocol and have been dealt with by the two-tier model of the designed protocol. The proposed protocol can defeat pre-computed attacks; by passing the device ICMetric through two tiers of the protocol based on two salt and pepper values.

The proposed protocol safeguards against rainbow table attacks, owing to its two-tier architecture. An adversary makes use of two 128 bits salt values and two pepper values thus dramatically increasing the amount of space required by the tables. This design element makes use of rainbow tables infeasible. The purpose of the salt and pepper values is to make the rainbow table a user specific element thereby increasing the complexity of the attack. A second salt and pepper value is used to provide a second layer of security with a random string in addition to the salt already being used. Doing so increases the workload and makes the activity very slow and cumbersome for the attacker. This process of pre-computation of rainbow tables for salted hashes is very slow; since it first requires the adversary to find salt value, and further to compute the rainbow tables with two 128 bits random values at run time. The proposed protocol follows a two-tier approach; where in the first stage the output of thousands of hash operations on the device ICMetric with a pepper and salt value become input to another huge number of hashing operations with a second salt and pepper value. These two stages of intensive hashing make the creation of rainbow tables more time consuming. For the attacker trying to break the stretched ICMetric key using rainbow table attacks, the proposed strong ICMetric key generation protocol increases the time and complexity thereby making the strong ICMetric key difficult to break.

6.4. Confidentiality

A vital security goal of the ICMetric public key framework is to maintain confidentiality of data. Preserving the secrecy of data is of utmost importance, so that the data is not leaked to adversaries. As communication takes place over insecure networks, the system should provide communication security that secures information exchanged between all participants. The proposed strong key generation protocol can be used in conjunction with any cryptographic algorithm. However in this paper, the RSA protocol has been used in conjunction with the proposed protocol to provide confidentiality of data in a public key setting. The proposed ICM-RSA protocol is based on an extended RSA key generation algorithm and the simulations for confidentiality are based on modified RSA encryption/ decryption algorithm. The greatest advantage of combining the ICMetric technology with RSA is that it aids in deterring attacks on cryptographic keys which can otherwise result in the system being exposed.

6.5. Non-repudiation

Asymmetric keys possess qualities which are not directly provisioned by symmetric keys. A prominent quality is non-repudiation. The proposed ICM-RSA algorithm has established that it is possible to create asymmetric keys based on the ICMetric of a device. The non-repudiation service is provisioned by virtue of the qualities of asymmetric keys. Thus when a message is encrypted with the private keys, a sender cannot deny the action as the private key is singularly held by that individual/ sender. The non-repudiation claim is backed by another strong guarantee as the scheme is based on ICMetric technology which does not rely on stored keys. Thus any claim related to key theft is rejected at source.

7. Performance Evaluation of the ICMetric Public Key Framework

This section evaluates the performance of the ICMetric strong key generation protocol and the ICMetric asymmetric key protocol. The strong ICMetric key generation protocol using SHA-256 and SHA-512 supports four strong ICMetric key variants to facilitate operation with varying key sized cryptosystems.The ICMetric key protocol supports four variants for the generation of ICMetric public key pairs.

7.1. Implementation

A working prototype of the proposed ICMetric public key framework was implemented using C on Linux. The framework has been implemented on a first generation Intel Core i3 3.2 GHz processor with 6 GB RAM using the OpenSSL cryptographic library [37].

The strong ICMetric key generation protocol has been tested based on SHA-256 and SHA-512 counterparts respectively. These counterparts each have four strong key variants to facilitate operation with varying key sized cryptosystems. The strong ICMetric key generation counterpart based on SHA-256 has been tested to support four strong ICMetric key size variants i.e. 256, 512, 1024 and 2048 bits; with a 128 bits device ICMetric as input. Similarly, the strong ICMetric key generation counterpart based on SHA-512 supports a 256 bits, 512 bits, 1024 and 2048 bits strong ICMetric keys respectively. The features used for the generation of the ICMetric were sequential input, sequential output, random seeks, max speed for copy function, etc. The two pepper values used in the implementation are also 128 bits long. The two unique salt values used in the implementation of the strong ICMetric key generation protocol are each 128 bits long. These salts are generated as a sequence of random bytes using timestamp as a seed to the random number generator.

The efficiency of the framework is tested by measuring the RAM consumption and execution time of the implemented framework. The runtime has been used for evaluating the execution time of the system, whereas Valgrind [38] is used for evaluating the memory consumed by the prototype. Valgrind is a code profiling and memory debugging tool for Linux. For memory profiling Valgrind uses Massif [39]. Massif is a memory profiler in Valgrind that measures how much memory the program uses during its lifetime. It gives information about the following thereby measuring all possible parts of memory required for the program:

- **Useful heap-** heap used by the program over time.
- **Extra heap-** heap blocks allocated for administration data over the programs lifetime.
- **Total heap-** Total of the useful heap and extra heap allocated for the program over its lifetime.
- **Stack-** stacks used by the program over time.

Therefore, all the results for memory evaluation are represented in terms of the above parts of memory occupied by the implementation of the ICMetric public key framework.

7.2. Execution Time

This section evaluates the time requirements of the proposed ICMetric strong key generation protocol variants using SHA-256 and SHA-512 respectively. The prototype uses a 128 bits device ICMetric and two 128 bits salt and pepper values as input for testing all the variants of the prototype.

To prove the scalability of the strong key generation protocols, the time taken for the generation of keys of varying sizes has been considered.

7.2.1. ICMetric Strong Key Generation

The performance comparison between the proposed strong ICMetric key generation protocol key variants and PBKFD2[30] is carried out based on the time taken for the generation of a strong ICMetric key over several experiments. Each variant has been further evaluated based on either SHA-256 or SHA-512. The figure 7.2.1 performs a time performance comparison between the two variants of the proposed ICMetric strong key generation protocol and PBKDF2, for generating 256 bits, 512 bits, 1024 bits and 2048 bits strong

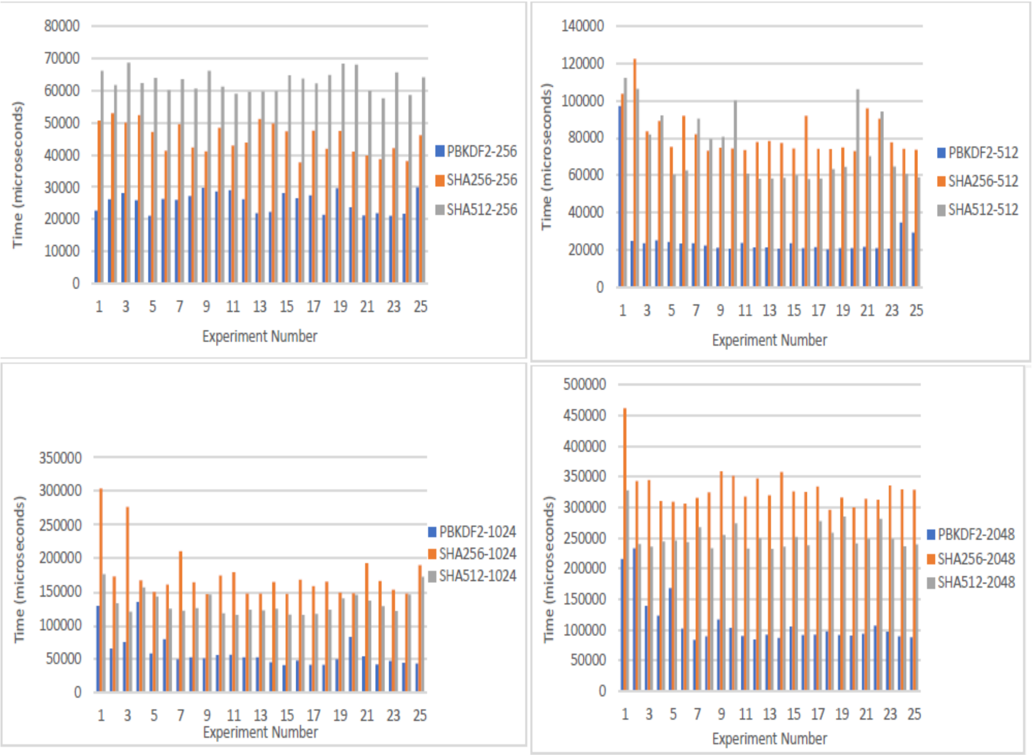

Figure 7: Execution Time Comparison of the ICMetric Strong Key Protocol Variants with PBKDF2 for the Generation of (a) 256 bits (b) 512 bits (c) 1024 bits (d) 2048 bits Strong Keys (Clockwise starting upper-left corner)

ICMetric keys respectively. The two variants namely SHA256-required key size and SHA512-required key size are compared with PBKDF2 to generate the required sized strong ICMetric key in graphs *(a)* to *(d)* in figure 7. The graph shows the number of the experiment versus the time taken in microseconds by the experiment. It is clear from the graph that the time taken by the strong ICMetric key variants is more than standard PBKDF2.There is an increase in the execution time for the proposed protocol, however the security advantages of the strong key protocol far outweigh the traditional PBKDF2. These results help assess the resource consumption of the strong key generation protocol.

7.2.2. ICM-RSA Key Generation

The evaluation of the proposed ICM-RSA key generation protocol is done by comparing it with the performance of the standard RSA key generation algorithm. These tests have been carried out to show the practicality of the system, while the security of the system was established theoretically above. As key sizes are increasing, therefore bigger key sizes have been simulated as a baseline. The graphs in this section show the trend of the time consumed by each run of the algorithm and is execution profile of the program. The graphs *a* to d *in* figure 8 are a comparison of the standard RSA and the ICMetric based RSA (ICM-RSA) key generation algorithm; when generating 1024, 2048, 3072, 4096 bits keys respectively. Repeated executions of the algorithm show that standard RSA algorithm possesses better performance. Additional resources are taken up by ICM-RSA due to the checks/computations performed on d at steps (e) and (f) of the ICM-RSA key generation protocol. At the same time, it must be stated that the use of the ICM-RSA algorithm is justified owing to the benefits it offers.

Minor fluctuations were observed when executing the key generation algorithm. The spikes are present because of the probabilistic nature of the protocol, and therefore the execution time may vary for each run of the program, but remains flexible as the complexity measure of the algorithm does not change. Although this is a substantial increase compared to the basic RSA key generation algorithm but the increase does not heavily impact the performance of modern computation systems. The 3072 bits key size is simulated in figure 8 (c) to determine how the ICMetric technology would influence futuristic applications where the key sizes can be 3072 bits or more. The basic RSA key generation algorithm is highly efficient in operations but this should not deter cryptographers from choosing the slightly expensive ICM-RSA.

Like the previous key sizes the RSA algorithm outperforms the ICM-RSA when considering the 4096-bits key in graph 8(d). The execution time of the ICM-RSA is excessively high but since measurements are in the microseconds scale therefore the performance is not severely hampered. After a large number of iterations, the average is computed for both the proposed protocol and the rivalling scheme. It can be concluded that the standard RSA algorithm is faster and on average takes 1704.008 microseconds, while the ICM-RSA algorithm offers higher levels of security but requires much more average execution time i.e. 773117.61 microseconds.

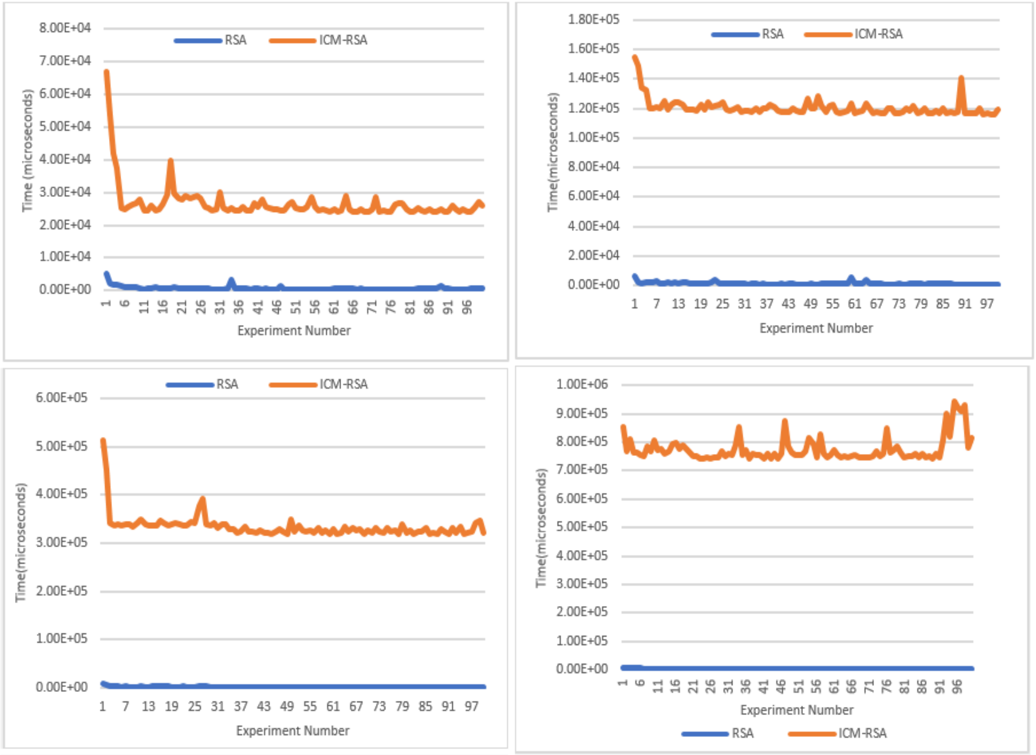

Figure 8: Execution Time Comparison of the RSA Key Generation with Extended RSA Key Generation for a (a) 1024 bits (b) 2048 (c) 3072 and (d) 4096 bits Key (Clockwise starting upper-left corner)

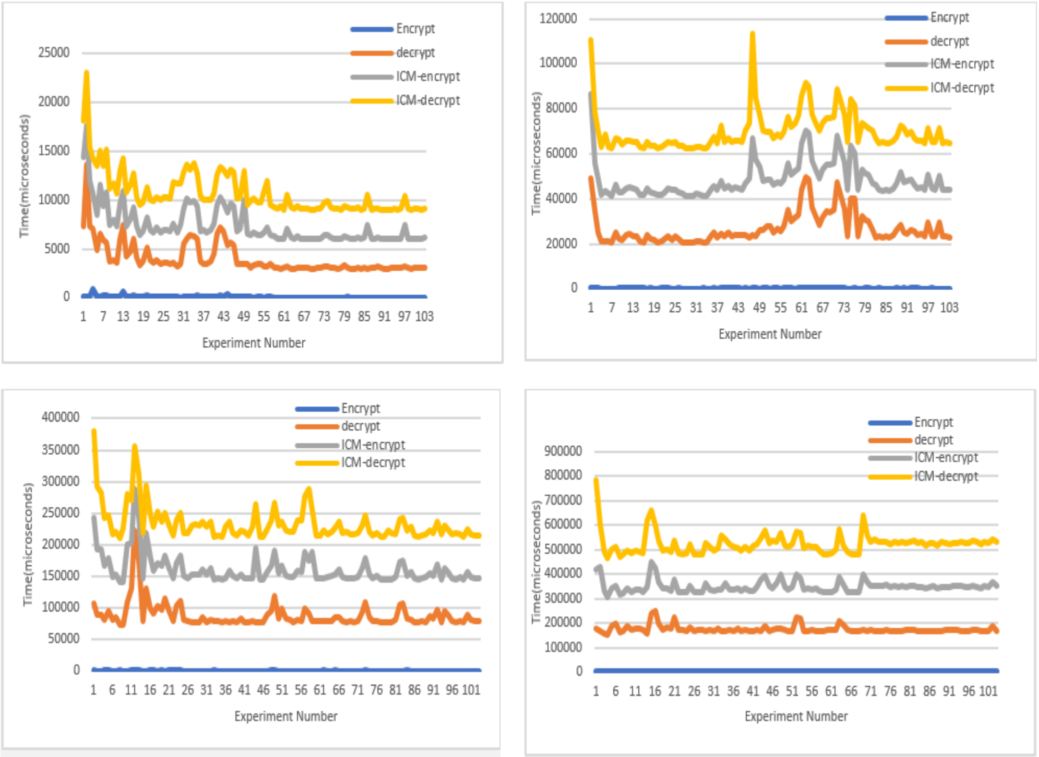

Figure 9: Execution Time Comparison of RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using (a) 1024 bits Key (b) 2048 bits key (c) 3072 bits key and (d) 4096 bits key (Clockwise starting upper-left corner)

7.2.3. ICM-RSA Encryption/Decryption

To understand the effect of key size on encryption and decryption operations, the generated ICM-RSA key variants are tested using both the proposed and the standard RSA protocols. It is evident from the superimposed graphs a to d in Figure 9 that the standard RSA is faster while performing encryption/ decryption.

A comparison of RSA encryption/ decryption with the proposed ICM-RSA encryption/ decryption shows very competitive results. The graphs *a* to *d* in figure 9 depict the proposed protocol performance comparison with the standard RSA using a 1024 bits, 2048 bits, 3072 bits and 4096 bits key respectively. Both protocols are very competitive in performance although the standard RSA outperforms the proposed protocol. The average time for key generation using RSA is very moderate while the ICM-RSA protocols average execution time is slightly elevated. The biggest factor influencing the resource consumption of ICM-RSA encryption/decryption is the value of e and d. Both the encryption/decryption in ICM-RSA are exponentiation functions; with the values of e and d being quite large. Therefore, the execution time is also fairly large since the time taken is proportional to the size of the exponent. The exponent d is the ICMetric number and quite large, therefore the time taken for decryption is quite large. Encryption time is also much greater in ICM-RSA, since e is much greater in case of ICM-RSA as compared to e in standard RSA. As there are frequent fluctuations in execution time therefore the average execution time is computed.

7.3. Average RAM Consumption Performance Comparison

In this section, the performance comparison between the proposed strong ICMetric key generation protocol and PBKDF2 is carried out based on the RAM consumed for the generation of a strong ICMetric key. The purpose here is to demonstrate that strong ICMetric based keys can be generated with minimum impact on the target system.

The graphs a to d in figure 10 performs an average RAM consumption comparison between the two variants of the proposed ICMetric strong key generation protocol versus the PBKDF2 for generating a 256 bits, 512 bits, 1024 and 2048 bits strong key respectively. These two used variants namely SHA256 and SHA512 are compared with PBKDF2 to generate varying sized strong ICMetric keys. The graph shows the name of the protocol versus the average RAM consumption in bytes. It is evident from the graphs a to d that the SHA-512 bits variant of the proposed protocol has the maximum average RAM consumption. A comparison of strong 256 bits ICMetric key generation based on SHA-256 and SHA-512 shows that, there is a 1.19 precent increase in RAM for strong ICMetric key generation based on SHA-512. This percentage increase in RAM translates to 16 bytes which is readily available in many modern computation systems. There is very slight difference in the RAM consumed by the four variants of the strong ICMetric key generation protocol based on SHA-256 and SHA-512. This is because the internal structures of both key derivation function blocks are very similar. Both the SHA-256 and SHA-512 have the same internal block size and the same internal state size. It is for this reason that there is very small difference between the memory consumption readings of the SHA-256 based strong ICMetric key variants and the SHA-512 based strong key variants.

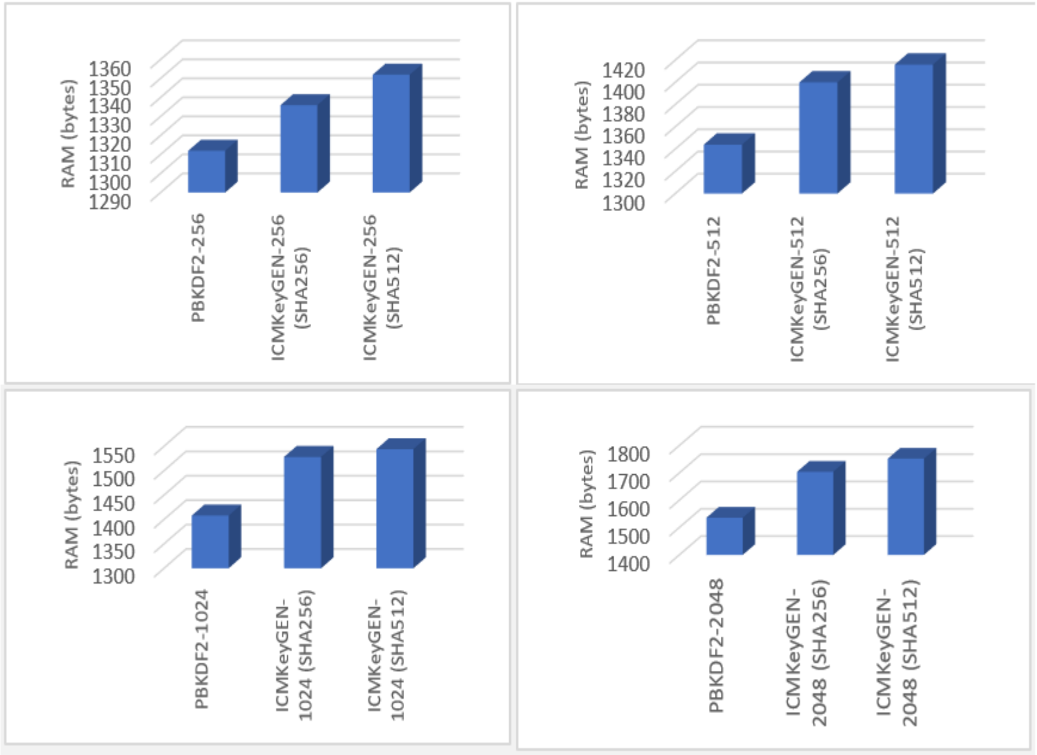

Figure 10: Average RAM Consumption Comparison of the ICMetric Strong Key Protocol Variants with PBKDF2 for the Generation of (a)256 bits (b) 512 bits (c) 1024 bits and (d) 2048 bits Strong ICMetric Keys

8. Conclusion

This paper presents a public key framework based on the ICMetric technology thereby demonstrating the feasability of the ICMetric technology for asymmetric key cryptosystems.The proposed framework bridges the gap between the ICMetric technology and its use in security applications. The proposed ICMetric strong key generation protocol and the ICMetric asymmetric protocol make the ICMetric public key framework adaptable to the requirements of different public key cryptosystems. The two-tier approach followed by the strong key generation protocol aims to strengthen the weak ICMetric key, thereby safeguarding it from pre-computed attacks and ICMetric capture. The RSA key generation algorithm is highly stable therefore efforts have been made to ensure that it is very carefully tuned to work with ICMetric technology, thereby providing confidentiality and non-repudiation.

The proposed framework has been simulated and tested to show that ICMetric based asymmetric keys can be generated without excessive resource demand. Simulation results show that the basic RSA algorithm outperforms the proposed algorithm, but this should not deter system designers and cryptographers as the ICM-RSA algorithm is both feature rich and provides increased security. The results obtained from performance evaluation and security analysis lead to the conclusion that at the expense of additional time and memory, the proposed protocol makes the ICMetric technology very viable for deployment in various applications.

References

[1] A. J. Brush, J. Hong, J. Scott, Pervasive computing moves in, IEEE Pervasive Computing 15 (2) (2016) 14–15.

[2] J. Soni, R. Goodman, A Mind at Play : How Claude Shannon Invented the Information Age, Simon & Schuster, 2017.

[3] Y. Dodis, The Cost of Cryptography (2013).
URL `http://nautil.us/issue/7/waste/the-cost-of-cryptography`

[4] C. Graham, NHS cyber attack: Everything you need to know about 'biggest ransomware' offensive in history (may 2017).
URL `http://www.telegraph.co.uk/news/2017/05/13/nhs-cyber-attack-everything-need-know-biggest-ransomware-offensive/`

[5] L. J. Camp, M. E. Johnson, The Economics of Financial and Medical Identity Theft, Springer US, Boston, MA, 2012. doi:10.1007/978-1-4614-1918-1.
URL http://link.springer.com/10.1007/978-1-4614-1918-1

[6] M. Miśkiewicz, B. Ksieeżopolski, Cryptographic keys management system based on dna strands, in: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS), 2019, pp. 231–235.

[7] K. Mcdonald-Maier, Device to generate a machine specific identification key, Tech. rep. (aug 2013).

[8] P. Barry, P. Crowley, Modern embedded computing : designing connected, pervasive, media-rich systems, Morgan Kaufmann/Elsevier, 2012.

[9] D. Genkin, L. Pachmanov, I. Pipman, A. Shamir, E. Tromer, Physical key extraction attacks on PCs, Communications of the ACM 59 (6) (2016) 70–79. doi:10.1145/2851486.
URL http://doi.acm.org/10.1145/2851486%

[10] R. Tahir, H. Huosheng Hu, D. Dongbing Gu, K. McDonald-Maier, G. Howells, Resilience against brute force and rainbow table attacks using strong ICMetrics session key pairs, in: 2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA), IEEE, 2013, pp. 1–6. doi:10.1109/ICCSPA.2013.6487307.

[11] D. Merli, R. Plaga, Physical unclonable functions: devices for cryptostorage, in: Proceedings of the 3rd international workshop on Trustworthy embedded devices - TrustED '13, ACM Press, New York, New York, USA, 2013, pp. 1–2. doi:10.1145/2517300.2528524.
URL http://dl.acm.org/citation.cfm?doid=2517300.2528524

[12] H. Tahir, R. Tahir, K. McDonald-Maier, On the security of consumer wearable devices in the Internet of Things, PLOS ONE 13 (4) (2018) e0195487. doi:10.1371/journal.pone.0195487.
URL https://dx.plos.org/10.1371/journal.pone.0195487

[13] J. Andress, Foundations of Information Security: A Straightforward Introduction, No Starch Press, 2019.
URL https://books.google.com.pk/books?id=kVv6DwAAQBAJ

[14] S. Tahir, I. Rashid, Icmetric-based secure communication, in: Innovative Solutions for Access Control Management, IGI Global, 2016, pp. 263–293.

[15] E. Papoutsis, Investigation of the Potential of Generating Encryption Keys for ICMETRICS, Doctoral thesis, University of Kent (2009).

[16] E. Papoutsis, G. Howells, A. Hopkins, K. McDonald-Maier, Integrating Feature Values for Key Generation in an ICmetric System, in: 2009 NASA/ESA Conference on Adaptive Hardware and Systems, IEEE, 2009, pp. 82–88. doi:10.1109/AHS.2009.30.

[17] M. Mehdi, M. T. Ajani, H. Tahir, S. Tahir, Z. Alizai, F. Khan, Q. Riaz, M. Hussain, Puf-based key generation scheme for secure group communication using mems, Electronics 10 (14) (2021). doi:10.3390/electronics10141691.
URL `https://www.mdpi.com/2079-9292/10/14/1691`

[18] G. H. W. G. J. W. G. T. R. I. G. Mcdonald-maier, Klaus Dieter (Harwich, Trusted ring (March 2020).
URL `https://www.freepatentsonline.com/y2020/0099521.html`

[19] R. Tahir, H. Tahir, K. McDonald-Maier, Securing health sensing using integrated circuit metric, Sensors (Switzerland) 15 (10) (2015) 26621–26642.

[20] K. Schramm, K. Lemke, C. Paar, Embedded Cryptography: Side Channel Attacks, in: Embedded Security in Cars, Springer-Verlag, Berlin/Heidelberg, 2006, pp. 187–206. doi:10.1007/3-540-28428-$1_1$1.

[21] R. Maes, Physically Unclonable Functions: Constructions, Properties and Applications, Ph.D. thesis, Katholieke Universiteit Leuven (2012).

[22] H. Bojinov, Y. Michalevsky, Mobile Device Identification via Sensor Fingerprinting, Tech. rep. (2014). arXiv:1408.1416.
URL `http://arxiv.org/abs/1408.1416`

[23] G.-J. Schrijen, V. van der Leest, Comparative analysis of SRAM memories used as PUF primitives, in: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE, Dresden, 2012, pp. 1319–1324.

[24] R. Maes, A. Van Herrewege, I. Verbauwhede, PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator, in: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, Berlin, Heidelberg, Leuven, 2012, pp. 302–319. doi:10.1007/978-3-642-33027-8$_1$8.
*URL*http://link.springer.com/10.1007/978-3-642-33027-8_18

[25] K. Dieter, Howells, W. G. James, Tahir, Ruhma, Trusted ring - metrarc limited (Mar 2020).
URL http://www.freepatentsonline.com/y2020/0099521.html

[26] E. Karnin, J. Greene, M. Hellman, On secret sharing systems, IEEE Transactions on Information Theory 29 (1) (1983) 35–41.

[27] S. Yadav, G. Howells, Secure device identification using multidimensional mapping, in: 2019 Eighth International Conference on Emerging Security Technologies (EST), 2019, pp. 1–5.

[28] M. S. Turan, E. Barker, W. Burr, L. Chen, Recommendation for Password-Based Key Derivation - Part 1: Storage Applications (2010).
URL http://csrc.nist.gov/publications/nistpubs/800-132/

[29] B. Kaliski, PKCS #5: Password-Based Cryptography Specification (2000).
URL http://www.ietf.org/rfc/rfc2898.txt

[30] A. F. Iuorio, A. Visconti, Understanding optimizations and measuring performances of pbkdf2, in: International Conference on Wireless Intelligent and Distributed Environment for Communication, Springer, 2018, pp. 101–114.

[31] NIST, Digital Identity Guidelines: Authentication and Lifecycle Management (2017).
URL http://csrc.nist.gov/publications/drafts/800-63-3/sp800-63b-draft.pdf

[32] D. Hook, Beginning Cryptography with Java, Wrox Press Ltd, Birmingham, 2005.
URL http://dl.acm.org/citation.cfm?id=1051127

[33] Z. Gutterman, B. Pinkas, T. Reinman, Analysis of the Linux random number generator, in: 2006 IEEE Symposium on Security and Privacy, IEEE, Berkley, 2006, pp. 15 pp.–385. doi:10.1109/SP.2006.5.
URL http://ieeexplore.ieee.org/document/1624027/

[34] A. Das, C. E. V. Madhavan, Public-key cryptography theory and practice, Pearson Education, 2009.

[35] D. R. Hankerson, S. A. Vanstone, A. J. Menezes, Guide to Elliptic Curve Cryptography, Springer, 2003.

[36] A. Visconti, S. Bossi, H. Ragab, A. Calò, On the weaknesses of pbkdf2, in: International Conference on Cryptology and Network Security, Springer, 2015, pp. 119–126.

[37] J. Viega, M. Messier, P. Chandra, Network security with OpenSSL, O'Reilly, 2002.

[38] Valgrind Documentation, Tech. rep., Valgrind Developers (2016).

[39] M. Wolff, Massif-Visualizer Memory Profiling UI, Tech. rep. (2011).