AN ICMETRIC BASED FRAMEWORK FOR SECURE END-TO-END COMMUNICATION

Ruhma Tahir

A thesis submitted for the degree of

Doctor of Philosophy

at

School of Computer Science and Electronic Engineering

University of Essex

2018

Dedicated to Papa and Mama

ABSTRACT

Conventional cryptographic algorithms rely on highly sophisticated and well established algorithms to ensure security, while the cryptographic keys are kept secret. However, adversaries can attack the keys of a cryptosystem without targeting the algorithm. This dissertation aims to cover this gap in the domain of cryptography, that is, the problem associated with cryptographic key compromise. The thesis accomplishes this by presenting a novel security framework based on the ICMetric technology. The proposed framework provides schemes for a secure end-to-end communication environment based on the ICMetric technology, which is a novel root of trust and can eliminate issues associated with stored keys. The ICMetric technology processes unique system features to establish an identity which is then used as a basis for cryptographic services. Hence the thesis presents a study on the concept of the ICMetric technology and features suitable for generating the ICMetric of a system.

The first contribution of this thesis is the creation of ICMetric keys of sufficient length and entropy that can be used in cryptographic applications. The proposed strong ICMetric key generation scheme follows a two-tier structure, so that the ICMetric keys are resilient to pre-computed attacks.

The second contribution of this thesis is a symmetric key scheme that can be used for symmetric key applications based on the ICMetric of the system. The symmetric keys are generated based on zero knowledge protocols and the cryptographic services are provided without transmitting the key over the channel.

The third major contribution of this thesis is a scheme that assists security critical applications in the generation of asymmetric keys. For this purpose, RSA has been tuned to work with the ICMetric technology thus providing high levels of security for asymmetric key applications.

The fourth major contribution of this thesis is the investigation into the feasibility of employing the ICMetric technology for identifying Docker containers employed by cloud service providers for hosting their cloud services.

ACKNOWLEDGEMENTS

First and foremost, my deepest gratitude goes to Professor Klaus McDonald-Maier for being a great mentor and for expertly guiding me throughout my Ph.D. His unwavering enthusiasm for innovative and practical research kept me constantly engaged with my work, and his supportive nature helped make my time at University of Essex enjoyable.

My appreciation also extends to Dr Gareth Howells, for providing technical support which was crucial in undertaking this research. I also wish to thank Professor Dongbing Gu and Dr Luca Citi for their valuable comments and feedback during meetings and discussions.

I am highly indebted to my parents for their encouragement and prayers during all the phases of the Ph.D. They have been a continuous inspiration for me.

A big thankyou goes to my wonderful husband; Ali Sajjad for his love and support during this venture, it wouldn't have been possible without his reassurance and belief in me. Thank you for being a listener during those extremely stressful times and cheering me up.

I am highly grateful to my brother Hasan Tahir, without whose help and support I wouldn't have been able to accomplish this goal. I am thankful to my younger brother Shahzaib Tahir for his continuous encouragement. It would not have been able to complete this journey without their reassurance and support. Lastly, I would like to

mention my son Zaim, daughter Eesha and niece Anabia for who have been a source of happiness for me when I felt stressed. Their naughtiness has brought me joy even when I felt stressed...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AES	Advanced Encryption Standard
DES	Data Encryption Standard
DoS	Denial of Service
ECC	Elliptic Curve Cryptography
EKE	Encrypted Key Exchange
IC	Integrated Circuit
ICMetric	Integrated Circuit Metric
ICT	Information and Communications Technology
IDEA	International Data Encryption Algorithm
KDF	Key Derivation Function
KGC	Key Generating Centre
NES	Networked Embedded Systems
NHS	National Health Service
NIST	National Institute of Standards and Technology
PAKE	Password Authenticated Key Exchange
PBKDF	Password Based Key Derivation Function

PKC	Public Key Cryptography
PKE	Public Key Encryption
PKI	Public Key Infrastructure
PUF	Physically Unclonable Function
RAM	Random Access Memory
RC4	Rivest Cipher 4
RSA	Rivest, Shamir, Adleman (public key encryption technology)
SHA	Secure Hash Algorithm
SRAM	Static Random Access Memory
SSL	Secure Socket Layer
WSN	Wireless Sensor Network
XOR	Exclusive OR
ZKP	Zero Knowledge Proof
ZKPP	Zero Knowledge Password Proof

1

Introduction

We as human beings are part of the information age. It can be said that the father of the information age is Claude Shannon [1]. It was his work which laid the way for what we see today in terms of computing, networking, artificial intelligence etc. This change has been made possible due to the emergence of new devices, technologies, high speed networks and innovative methods of interactions between users. Computation systems in the information age are no longer standalone devices which process small data sets. Today computation systems are complex devices which are available in different forms, shapes, sizes and capabilities. For instance, some devices are designed to be carried, worn on the body or installed ubiquitously into our environment. When devices become part of everyday life the boundaries between the physical world and the virtual world often overlap. In such a situation, the importance of security cannot be denied. Today, the notion of security exceeds that of confidentiality. In fact, security means a comprehensive set of services which includes authentication, authorization, privacy,

etc. Advancement in the field of Information and Communication Technologies (ICT) has created a compelling case for enhancing the security requirements and ensuring secure communication over the internet. Effort is constantly being made to manufacture devices and system that are at the cutting edge. Yet security is often considered a secondary service or an unnecessary overhead [2]. Often systems are attacked owing to poor design, flawed implementation or a lack of understanding of why a system would be attacked in the first place.

A widespread attack in 2017 has resulted in adversaries being able to launch 75,000 attacks in 99 countries. The National Health Service (NHS) in the UK was one of the hardest hit [3] and has resulted in health records being compromised at an unprecedented scale. The attack was based on a ransomware, which carried out AES and RSA encryption on stored data while the adversaries had access to the encryption keys. It was interesting to note that the data was held ransom by the adversaries and only released when a cash sum was paid. Here it must be said that compromise of healthcare records can be a serious threat to patients and doctors [4].

Cryptography is a well-established method being used for centuries to address the military demands for secret sharing. Today the use of cryptography is not just limited to the military. Cryptographic services are the need of every user whether his devices or systems connect to the internet or not. A very important proposition made by Kerckhoff was that the security of a cryptosystem should not rely on the secrecy of the employed algorithms, instead on the secrecy of the key(s) [5]. To date, this axiom is widely accepted by cryptographers and secure communication schemes are designed following this recommendation. Traditional security implantations rely on keeping the key secret and the designers believe that the system cannot be broken owing to

algorithmic or mathematical intractability [6] even though the algorithm is published. Although this is true, one must understand that adversaries do not always target algorithms. Instead they resort to methods like side channel attacks [7] to capture keys and other secret data. Owing to this reason novel trust basis are being researched which can help in designing systems which can deter attacks related to key compromise.

This chapter states the purpose of this research by narrating the problem statement. Furthermore, this chapter sheds light on the goals that are to be achieved through this research.

1.1 RESEARCH MOTIVATION

Embedded computing systems play a major part in various aspects of life, ranging from their use in single user applications to large scale processes. Examples of their use in large scale processes could be sensors aggregating sensed data and transmitting it to a centralized server or simply their use in handheld devices for a range of single user applications [8].

Often security is considered a system overhead owing to lack of resources or increasing complexity. This is particularly evident when considering the resource-constrained nature of embedded systems. Resources planned during the initial development phase do not account for security functions which makes security a costly service for the system.

The security of many applications depends on the protection of its cryptographic keys, since the underlying algorithms are published and globally known. Therefore, there are inherent risks if keying materials are not protected. Without adequate storage and handling of keys, they could be easily captured [9][10][11], modified, or substituted

by unauthorized personnel who could then intercept sensitive communications and takeover the system. This can result in serious compromise in the confidentiality and availability of communications. Embedded systems have limitations in terms of power, memory and processing speed; therefore, care needs to be taken to choose cryptographic mechanisms for key generation, and to determine key sizes based on individual applications requirements.

Cryptographic algorithms designed to provide secure communication in embedded system applications rely on stored encryption/decryption keys [12]. These algorithms have the inherent disadvantage that the compromise of an embedded system device, can lead to key/ secret information being revealed to the adversaries. This can ultimately result in the entire network being overtaken or important data being revealed.

Secure one-to-one communication is a fundamental need for every device that is communicating with another device in any application environment. A wide range of attacks [13][7] like side channel, man in the middle; pre-computed attacks like brute force, dictionary attacks can all be used to compromise the security of a system. Even a system built with the latest security primitives can be broken. Effort is needed to ensure that the security of the primitives remains intact and that the keys of a system are impenetrable.

1.2 SECURITY AIMS

Before the design of a security scheme can begin it is important to identify the security goals which need to be fulfilled. These goals play a crucial role as they provide guidelines to the designers about which primitives are suitable for the problem at hand. The security goals of this research are presented below:

- Confidentiality - the fundamental requirement that protects information by restricting access to only authorised users of the system. As communication takes place over insecure networks, the system should provide communication security that secures information exchanged between all participants.
- Authentication - enables an entity to ensure the identity of the entity it is communicating with. Without authentication, an adversary could masquerade an entity, thus gaining unauthorized access to resource and sensitive information and interfering with the operation of other entities.

Mutual Authentication - all participants should obtain an unforgeable proof of other participant's identity before engaging in a protocol run with each other.

1.3 THESIS STATEMENT

Digital devices are becoming increasingly ubiquitous, and this gradual shift towards pervasive computing envisions many benefits in sectors as diverse as finance, entertainment, healthcare, information access, automotive etc. Along with these advantages, there are also some associated risks which need to be addressed. Security of communication protocols has always relied on the stored cryptographic keys. The stored keys can be compromised therefore a framework is required that assists in the generation of keys at runtime. It is in this context that this dissertation is situated. It provides contributions to the security of embedded devices and ensures the privacy of the humans interacting with them. The problems associated with key theft and stored keys have led to the development of new security paradigms that focus on a new root of trust, designed by using hardware and software features of a device. There are several factors which make attacks on embedded systems successful. While all factors are significant, this thesis starts from resolving security issues at the design phase of the

system development. In this research a principled approach to designing and deploying secure end-to-end Integrated Circuit Metric (ICMetric) [14] based cryptosystem has been studied. The goal of this research is to design and develop a cryptosystem that provides authentication, confidentiality and non-repudiation of data between entities by using the ICMetric technology. The ICMetric technology is a novel trust basis and proposes using the features of a system to create a device identification. Once this identification is generated it can be used for the provision of security services. Thus, the ICMetric technology functions as an alternative to stored keys and as the basis of cryptographic services. The ICMetric technology can be adapted to the target environment; however, the short length, low entropy and confidentiality of the device present design challenges. Besides this ICMetric has its own set of properties and constraints which need to be observed when designing protocols based on this novel technology. Therefore, the design of a comprehensive ICMetric based framework has been studied which facilitates symmetric and asymmetric ICMetric applications for secure services in the end-to-end environment.

1.4 THESIS CONTRIBUTIONS

The specific contributions of this work pertain to the design and implementation of an umbrella of schemes for secure end-to-end communication for entities based on the ICMetric technology.

Traditionally the security of cryptographic schemes relies on keeping the keys secret while the algorithms are widely published. If the keys are captured or exposed, then the system can be compromised. Security algorithms rely on algorithmic/ mathematical intractability to ensure that the system cannot be broken. It is now

possible for adversaries to attack and capture the cryptographic keys without targeting the algorithm. This thesis explores the use of ICMetric technology as a key theft deterrent and as basis for cryptographic key generation in the end-to-end communication environment. Given below is a summary of contributions made by this research.

The first contribution of this dissertation is the design of a strong key derivation scheme for ICMetric based entities. The ICMetric of a device cannot be used directly as a key [15] for cryptographic operations, therefore the proposed scheme demonstrates how to generate a high entropy ICMetric key of sufficient length that can be used with other cryptosystems.

The second contribution of the thesis is the design of an ICMetric symmetric key protocol for devices based on the ICMetric technology. The proposed protocol utilizes the concept of zero-knowledge proof to establish a symmetric key for secure end-to-end communication. This protocol provides secure admission control, authentication, confidentiality and non-repudiation of data for entities that are part of the communication.

Many modern cryptosystems use asymmetric keys as a necessary design element. This thesis explores how the ICMetric technology can be combined with RSA to generate a strong asymmetric key pair. Therefore, the third contribution of this research is the design of an asymmetric cryptographic scheme which can be used for applications based on the ICMetric technology. This scheme fulfils the goal of confidentiality and device based non repudiation.

The fourth contribution of this thesis is the deployment of the ICMetric technology on the Docker containers. The work demonstrates that the unique features of Docker containers can be used for their identification, thereby generating an ICMetric of the Docker container. Hence the use of the ICMetric in the cloud environment of leading cloud service providers is a testament to the fact that the technology has the potential to mitigate weaknesses found in cryptography.

Throughout this research effort has been made to ensure that the ICMetric technology does not add excessive overhead to the existing systems. Thus, design choices have carefully been made so that cryptographic algorithms are chosen that complement each other thereby enhancing the overall security of the system. The proposed schemes have been designed for adaptability, thus the ICMetric technology can be adapted to existing security systems without major modifications.

1.5 THESIS STRUCTURE

The subject matter of the dissertation is logically divided based on the main contribution each scheme makes for establishing secure end-to-end communication. The thesis chapters are organized as follows:

- Chapter 2 presents a detailed study on concepts related to secure end-to-end communications. The chapter begins with an introduction to embedded systems and the design constraints in this unique environment. A survey on attacks is presented to show that embedded system can be attacked through various methods. The remaining chapter focuses on novel security founding concepts like Physically Unclonable Functions [16] and the ICMetric technology. The

chapter also presents a detailed discussion on the ICMetric technology and how the ICMetric of a device can be generated.

- Chapter 3 demonstrates that the ICMetric technology can be used to create strong keys by using the key derivation functions. The ICMetric of a device possesses unique properties owing to which it cannot be used directly as a key. Hence this chapter shows that a strong key can be generated using the ICMetric of a device, thereby safeguarding the key against pre-computed attacks.
- Chapter 4 focuses on the generation of symmetric keys by using the ICMetric of a device. The proposed scheme uses zero knowledge password proofs [17][18] to generate the symmetric keys which can be employed to carry out confidentiality services in secure end-to-end communications. The chapter presents a detailed performance analysis of the proposed scheme to show that the system functions without excessive resource demands.
- Chapter 5 focuses on the design and implementation of an RSA key generation scheme which uses ICMetric technology as a design basis. By generating the RSA keys based on ICMetric of the entity, the resulting system provides a strong guarantee of secrecy and non-repudiation for asymmetric cryptosystems based on the ICMetric technology.
- Chapter 6 is a practical case study which shows how the ICMetric technology can be used in securing applications in Docker containers. The presented work studies the Docker container environment being used by a leading cloud service provider for hosting their applications in the cloud environment and uses the ICMetric technology for authenticating the Docker containers.

- Chapter 7 concludes the thesis with a summary of contributions and possible future research directions.

2

Literature Review

2.1 INTRODUCTION

Computation devices and systems come in many sizes and forms. Owing to the readily availability of computation power and sufficient bandwidth most computation systems have transitioned from standalone systems to dynamic entities that sense, process, and communicate data. The importance of security cannot be denied during any of these individual phases. Security of a system cannot be limited to just data secrecy. There are other concerns which need to be addressed like authentication, confidentiality and non-repudiation. Conventional computation systems can possess necessary resources required to establish security services. However, the same cannot be said for the embedded systems environment. Owing to several design limitations in embedded systems environment the designers must cut corners to create the best possible system [19]. This chapter begins with a survey on embedded systems and then

presents a detailed discussion on the inherent design constraints. The chapter then studies the security of embedded systems. A detailed study on attacks in embedded systems has also been presented in the chapter. Although there are many forms of attacks in embedded systems [20], effort has been made to particularly focus on side channel attacks because they attempt to expose a system without targeting the conventional root of trust (mathematical intractability). Hence a discussion on Physically Unclonable Functions and ICMetric is presented as a novel root of trust for designing cryptographic systems. A major contribution of this chapter is a discussion on the design and operation of the ICMetric technology. As the security of an ICMetric based system is founded on its features, therefore a comprehensive discussion on features for ICMetric based entities has been presented in the chapter.

2.1.1 Embedded Systems

Many modern devices are now designed with an embedded system. Pinning down an exact definition of embedded systems is a complex task because the area is constantly undergoing research [21]. This means that embedded systems are evolving thus a definition which is agreed upon today may change in the near future.

Embedded systems can be defined as applied computation systems based on both hardware and software [22][23]. An embedded system does not look like a computer even though it is composed of multiple processors and software components. What sets embedded systems apart from conventional computation systems is the fact that they are designed to perform a narrow range of predefined tasks. Another distinguishing factor is that embedded systems possess limited hardware and software resources compared to a computer. It is these characteristics that greatly influence

possible applications of embedded systems and how they are designed. Discussed below are the characteristics of an embedded system[24].

- Real time processing – embedded systems are designed to process data in real time. Furthermore, the practicality and suitability of a system is often a function of time.
- Resource limitations – embedded systems have limited hardware and software capabilities. In terms of hardware an embedded system can have limited ROM, RAM, computation capacity, etc. In terms of software the limitation can mean a scaled down or no operating system, reduced functionality in applications etc.
- Small size – embedded systems are generally small in size, so that the dimensions of the target system can be reduced. Restriction on the size can also influence the resources and capabilities of the embedded system.
- Limited power – embedded systems can be powered using batteries. The embedded systems are designed to conserve power which can mean that the system will have limited computation capabilities.
- Higher Reliability – embedded systems have a higher reliability and quality requirement compared to conventional computation systems (PC). This is particularly important in domains like healthcare, nuclear systems, vehicle control, etc.
- Harsh environment – embedded systems may be deployed in harsh conditions with excessive heat, humidity, vibrations, chemical corrosions, etc.

Embedded systems are being used and experimented with in many applications; like televisions, home automation systems, vehicle control systems, bank ATM

machines, etc. Generally embedded systems can be placed into four categories [20] based on capability. The four types of embedded systems are discussed below.

- Consumer electronics – embedded systems commonly found in desktop systems, video games, wearable devices, etc.
- Control systems – an embedded system designed to enable closed loop feedback in complex systems like vehicles, flight control, chemical processes, etc.
- Signal processing – embedded systems designed to process large amounts of data streams. These embedded systems are commonly found in Radar technologies, SONAR, satellite systems etc.
- Telecommunication and Networking – Embedded systems that are commonly found in communication systems like telephones, exchange equipment, network enabling hardware.

A typical embedded system can be represented as a three-layered block. Figure 2.1 below is a modular view of a typical embedded system [21].

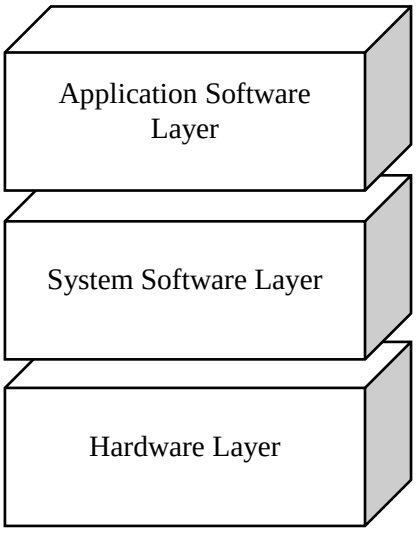


Fig 2.1. Modular View of a Typical Embedded System

The hardware layer is composed of all the physical components, sensors on the embedded board. The hardware layer connects to the system software layer which

is utilised by the embedded system for performing its functions. The system software layer is the firmware which is necessary to run the hardware of the embedded system. At the very top of the stack is the application software layer. This layer is an optional layer and is used by the users to run the embedded system.

2.1.2 Networked Embedded Systems

As devices with embedded systems become increasingly prevalent the next step in their evolution is the creation of Networked Embedded Systems (NES). The term networked indicates the presence of communication/ networking infrastructure and capability in embedded systems. Thus, a NES is a collection of spatially distributed nodes which communicate through a network interface. Owing to the presence of networking capabilities; NES components are actively embedded into trains, aircrafts, vehicles, industries [25], etc. The primary use of NES is to enable control, monitoring, evaluation of the target environment. Research [26] has shown that there are numerous applications of networked embedded systems like in home appliances, office automation systems, security systems, industrial automation, medical instruments, aerospace, etc. It is worth noting that all these applications of networked embedded systems rely on secure communications. From the presented applications of NES, it can be concluded that it is highly likely that a NES will be deployed in very demanding, high reliability requiring real time environment. In such a setting the NES will interface with other systems, thus connecting to a reactive system which can take steps to respond to the sensed stimulus.

2.1.3 Security of Embedded Systems

In the early generations of embedded systems, security was not given much importance [27]. It was assumed that hardware is impenetrable and security is a service

that is provided by the software layer. As adversaries now have access to high levels of computation resources, designers have realized that security is no longer an optional service[28][29]. Designing and implementing security for embedded systems is a challenge owing to the presence of design constraints. Given below are the design challenges faced by designers when implementing security in embedded systems.

- Processing shortcoming – embedded systems may not be capable of supporting the computational demands of cryptographic implementation.
- Energy limitation – cryptographic algorithms have an energy overhead which is not being fulfilled by the pace of development of embedded systems. Owing to power limitations designers make design choices for optimized security implementations [30][31][32].
- Tamper Resistance – embedded systems are designed to be tamper resistant. Software based attacks that use Trojan horse can provide access to the internal environment of the system. Besides this the Trojan horse can also disrupt the regular functioning of the embedded system. Research [13] has shown that hardware based attacks like probing, material removal and contactless radiation imprinting can be used to successfully attack a system.
- Inflexibility – security schemes are always under attack owing to discovered weaknesses, and improvements are suggested accordingly. Embedded systems can lack flexibility which will make it difficult to adapt the system to emerging design requirements. Designers face this issue and have to choose schemes which can be adapted to the embedded system with minimum design disruptions.

- Cost – designers of embedded systems have to make crucial decisions when designing the system i.e. provide higher levels of security at the expense of increased effort, cost and design time. This is particularly important when designing mass marketed consumer devices [26].

To unlock the full potential of embedded systems it is necessary that the devices possess the resources and have been designed for the provision of secure services. The inherent design challenges show that implementing security in embedded systems is a complex task which can influence the security of the resulting system.

Embedded systems are a rapidly emerging area where devices are being manufactured with chips from a handful of companies. Companies make design decisions based on economic returns. Thus the cheapest chip is selected and a device is built around what is made available to the designers [33]. Beyond this, very little engineering is put into the design. To make the situation complex security experts repeatedly try various attacks and post their weaknesses on the web. This means that companies are expected to release patches and updates to fix the vulnerabilities. When considering internet devices like wireless routers it is worth pointing out that most users never update their router's firmware to incorporate the latest security updates. A recent attack on internet devices was launched by using the Mirai malware [34]. The malware targets internet devices like routers and IP cameras which are using default settings (usernames and password). Once a device is infected, it is used to drown a remote server with large amounts of data, thus creating distributed denial of service [35]. It is estimated that 100,000 compromised nodes (botnet) were used in this attack to globally compromise popular websites related to social media and news agencies. The attack

continues to infect systems today, based on an evolved malware [36] which targets the Linux OS with focus on the application layer, rather than the network layer.

2.2 ATTACKS ON EMBEDDED SYSTEMS

Cryptographers have traditionally assumed that a cryptosystem is designed and implemented in a closed, reliable environment [7]. This is not always the case because of which many cryptographic systems are broken. Attackers now have access to resources that are sufficient to launch a high-level attack. For an adversary, the motive behind an attack could be to capture secret information, commit financial fraud or even just cause nuisance. Adversaries will use multiple methods of attack aided with a range of tools to gain access to the system and its resources.

An internet baby monitor is a good example of a system which is designed to provide a specific service and often possesses limited resources for the provision of security. Research [37] on the Foscam baby monitor has shown that the camera can be hacked by a remote adversary. The attack takes place by first determining the IP address of the camera. Once this is done an adversary can remotely download the entire memory contents of the camera which includes the username and password. The username and password is used to gain access to the device and its functionalities. This type of attack targets authentication credentials leading to compromised confidentiality. The baby monitor like most video monitoring systems has no provisions for integrity and non-repudiation. Just providing confidentiality and access control is not a sufficient security solution.

Perhaps the most common type of attack in embedded systems is the side channel attack and is accomplished through power analysis [38]. When electrons flow

across a silicon substrate the components of the embedded system give off radiations. These radiations are captured and studied to extract the cryptographic keys. The power variations are indicative of the low-level structure of the cryptographic algorithm; thus, an adversary can identify instructions like MOV, XOR, etc. To successfully carry out the attack an adversary needs access to the target device along with a digital oscilloscope connected to a computer. The adversary will use probes to extract the power consumption at a specific sampling rate. Once the information is recorded on a computer of the attacker can see variations between clock cycles owing to the execution of varying microprocessor instructions [38]. This data is processed to yield the cryptographic key of the system.

An early yet widely recognized study [39] has shown that it is possible to use timing analysis to determine the cryptographic keys of a system. The attack is based on making predictions about bits of a cryptographic key and then using statistical methods to determine if the behaviour of the target machine corresponds to the expected behaviour. The adversary starts the attack by predicting the first bit (0 or 1) of the key sequence. Then statistical analysis is used to indicate which bit comes next based on predicted and actual computation time. This process is repeated until the entire key is captured. When the attack was launched in 1993, it took only a few thousand iterations to reveal the 256-bit secret exponent of RSA. Here it must be mentioned that the same attack with today's computation speeds will require substantially less processing time.

A recent study [37] has shown that embedded system devices can have implementation flaws or poor design. The firmware of a smart television uses the XOR cipher [40] to perform encryption. The implementation uses a key which is much smaller in size than the plaintext which means that a large part of the plaintext is never

encrypted. The weakness can be exploited to extract the key by using a single bit plaintext which will result in the entire key being exposed as a result of conducting the XOR operation.

Researchers [9][41] have shown that it is possible to steal RSA and ElGamal encryption keys by using a portable instrument that can detect fluctuation in electronic field surrounding a laptop. The authors show the design of Portable Instrument for Trace Acquisition (PITA) which is a compact untethered, concealable device for stealing keys from a common laptop. A target laptop is supplied with "carefully crafted" ciphertexts which causes the decryption system to generate special structured values. These values cause observable electromagnetic fluctuations around the laptop. PITA operates at a 1.7 MHz frequency and picks up these fluctuations from the laptop and stores them on the internal memory. The data in the internal memory can be used to extract the cryptographic keys. The authors have successfully tested the attack on various laptops and confirm that the attack can be replicated.

The above presented attacks show that embedded systems can be attacked without much effort. From the presented examples, it can be concluded that to attack a cryptosystem the fabric of the security algorithm does not need to be attacked. There are many forms of attacks, but a source of major concern is that the standalone or networked embedded system can be attacked to compromise the cryptographic keys. Once a cryptographic key is captured then the security of the entire system can be compromised. Weaknesses in design, implementation and limited resources shows that a renewed approach is required for improved safety, security and privacy in embedded systems. A novel approach can be based on incorporating an alternative root of trust which can ensure higher levels of security without compromising the resource demand.

2.3 PHYSICALLY UNCLONABLE FUNCTIONS

Cryptographic algorithms use mathematical intractability as a necessary ingredient in their design. As attackers become increasingly powerful it is possible for them to break a system by finding weaknesses in the cryptographic system. Studies presented earlier show that adversaries will not always use mathematical intractability to break a system. Instead they can resort to methods that do not even attack the cryptographic algorithm. Variations of the side channel attacks, brute force, dictionary attacks, etc., all attempt to compromise a system by exploiting weaknesses or by exhaustively searching for secret information. It is worth noting here that these attacks do not target the mathematical intractability of the system.

Attackers have multiple and diverse methods of attacking a system, therefore researchers now consider alternatives to the conventional algorithmic intractability as a root of trust. A relatively new root of trust undergoing research is the Physically Unclonable Function (PUF). An embedded system is composed of hardware elements which possess minor variability owing to difference in the manufacturing processes. Thus, two identical embedded systems that are manufactured by a single manufacturer will behave slightly different to each other. These measurable variations are used to create a one-way function called a Physically Unclonable Function (PUF), as shown in Figure 2.2.

Figure 2.2. Architecture of a PUF

As depicted in Figure 2.2, a PUF accepts as input; a challenge and produces as output a response which is then used for the provision of wide range of services among which is cryptography. The working of a typical manufactured PUFs is based on the following steps:

1. The server obtains access to PUF and generates a table of challenge-response pairs that are stored in the internal storage.
2. The PUF is then given to client, whereby a client submits a request to the owner to authenticate.
3. The server in the next step picks a known challenge response pair and submits the challenge to client.
4. The client runs the challenge on PUF and returns the response to server.
5. The server checks to see if the response is correct and marks the challenge response as used.

When studying PUF's it is necessary to identify features which exhibit properties like unpredictability, unclonability, robustness and tamper evidence. These properties are crucial to the design of PUFs and distinguish it from any other one way function. PUF's have been used and experimented in various settings. For instance, PUF can be used for cryptography, IC-identification, device authentication, random number generation and counterfeit prevention. A particularly interesting application of PUF is to identify if the binding between the hardware and software is correct [42]. As PUF's are tamper evident, therefore they can be used to detect and prevent overclocking in computation systems. Perhaps the greatest advantage of PUF based cryptography is that it can be used to eliminate the need for stored keys. Thus, the PUF can be used as a basis for cryptographic key generation. When a PUF is challenged with an input it

will provide a response which will be processed for key generation. If the device is tampered or cloned, then it will not possess the same PUF environment required to create the correct cryptographic key.

The stability and security of a PUF based system is heavily dependent on the features used for creating the PUF. An early research [43] on features suitable for a PUF was on optical PUF's. In the research, authors show that a stationary scattering medium placed in path of a laser beam results in a unique scatter pattern. Even though the research is convincing it still has limited applications. Another research [44] has shown that it is possible to use an improvised RFID as a PUF. The system is designed by providing a 64-bit input challenge and obtaining the associated response. The system uses a scrambling circuit for preventing learning based attacks. A list of challenge and associated responses is created. This list can then be used for authentication. Similarly research shows that the delay in logic gates is suitable as a PUF [45]. The problem with using logic gates is that their behaviour is susceptible to change with varying voltage and temperature.

Experiments [46][47] on SRAM based PUF implementations show that each Static Random Access Memory (SRAM) cell has a unique and unpredictable start-up state (either zero or one). Thus, the start-up state from the individual cells can be combined as a unique device signature. The start-up state of a SRAM cell is truly random but repeatable which makes it a very suitable feature for use in PUF. An SRAM is a faster RAM which is why it is commonly used in the design of cache inside a processor. This means that an SRAM PUF [48] can be deployed in embedded systems.

A silicon chip can be tamper evident with the help of a coating PUF [49]. A silicon chip is coated with a randomised dielectric coating which can affect the

capacitance of the embedded sensors. This unique quality is highly effective as a tampering indicator because the slightest change in a single component will cause a change in PUF operations. The problem with this type of PUF is that coating the silicon chip is a costly procedure which will increase the cost and complexity of the IC manufacturing process.

2.4 INTEGRATED CIRCUIT METRIC

Integrated Circuit Metric (ICMetric) [14][50][51] refers to a modern technology that can be used to extract measurable and unique attributes from the hardware and software environment of a system. The technology exploits the fact that each device is unique in its internal environment, therefore the factors that make each device different can be used to generate a single and unique identification for every device. This unique identification can be used for the provision of cryptographic service. Hence the ICMetric technology is a novel root of trust which can resolve problems associated with cryptographic key theft, side channel attacks, device cloning, tampering, etc. The concept of ICMetric is significantly different from a Physically Unclonable Function(PUF), whereby authentication is based on a challenge response mechanism and the entity in question is only authenticated if its response to the challenge matches to the one stored by the server. The ICMetric technology is not a challenge response mechanism like PUFs, infact it is comparable with biometric technology [52] i.e., the detection of individuals using their varying physical and behavioural characteristics such as fingerprints, iris patterns, voice, etc. Unlike biometric systems, little work has been undertaken in building analogous systems based on machine-derived data and linking a machine's behaviour with data.

The ICMetric technology is based on the idea of extracting features that, when combined, uniquely characterize the functioning of the given electronic device. This is possible as typically devices operate under unique conditions; they sense different environments, run different software, perform different tasks and interact with different users. Research has shown that such features can be used for generating unique encryption keys [51][50] and detecting cases of untypical device behaviour. ICMetric allows the use of the distinct characteristics that are generated even for identical hardware and software, based on the individual user's usage and the effect of the environment on the system. Example features are location information and patterns, app usage patterns, content and content changes on the device etc., together with low level operating characteristics of the system itself. It is these features of the ICMetric technology, that make it significantly advantageous over PUFs. The ICMetric technology provides non-repudiation in addition to authentication, since the ICMetric serves as the identity of the device and not a response to a challenge in question. This leads to a major advantage of the authenticity of the sender in the application making use of the ICMetric technology.

The measured characteristics need not remain constant but are free to vary within deduced parameters, thus allowing the software to operate in several states and on a variety of differing platforms, whilst still ensuring that any illegal clone or malware infecting the software is detected via unacceptable changes in the operating parameters. This also acts as a security enhancing feature in that an attacker cannot easily derive the information to reproduce the characteristics in an attempt to break the code by observing the device. ICMetric represents an approach for generating unique identifiers for systems enabling secure encrypted communication between devices significantly

reducing both fraudulent activity such as eavesdropping and device cloning. Specifically, ICMetric has the potential of providing the following secure services.

- Use of ICMetric technology implies that no encryption keys or device characteristic templates are stored, and are regenerated as and when required.
- Prevention of unauthorised access to the device.
- Prevention of the fraudulent cloning or imitation of the device to compromise its identity and subsequent communication.
- Implicit detection of tampering of the software associated with the device via the inclusion of spyware or similar malware software since this will implicitly cause the generated ICMetric number to vary.

A significant novelty of the system being developed lies in the potential for the direct generation of ICMetric extracted from ICMetric samples, which characterize the identity of the device. Such a system offers many significant advantages over conventional security system. One advantage is the removal of the need to store any form of template for validating the device, hence directly addressing the major weakness that the feature templates are accessed and used to circumvent the security afforded by the system. The security of the system is as strong as the ICMetric and encryption algorithm employed (there is no back door). Hence a system can be broken if a device ICMetric is revealed or by breaking the employed encryption technology. The compromise of a system does not release sensitive ICMetric template data, which would allow unauthorized access to other systems protected by the same ICMetric or indeed any system protected by any other ICMetric templates present. Tampering with the constitution of the software will cause its behaviour to change, potentially causing

the features underlying the ICMetric to change, perhaps dramatically, thus causing the generated ICMetric to change. Consequently, a faulty or maliciously tampered device will be autonomously prevented from decrypting its own stored data or participating in any initiated secure communications, as the regenerated keys will differ from those created before it was compromised i.e., the ICMetric approach can be made to fail securely and provide a very high immunity from cloning and tampering. This also means removal of the need for the storage of the private key. As there is no physical record of the key, it is not possible to compromise the security of sensitive data via unauthorized access to the key.

The ICMetric technology has been designed as a distinct layer which can be incorporated into the existing application stack with minimum changes to the design of existing cryptosystems. Doing so creates an adaptable design and reduces complexity thus creating a pluggable system based on ICMetric.

2.4.1 Creating an ICMetric

The ICMetric of a device is not a simple amalgamation of device features. Complex mathematical methods ensure that the resulting ICMetric is truly unique and can be used in a cryptographic key generation algorithm. The ICMetric generation process is composed of two phases i.e., calibration phase and operation phase. The ICMetric and the ICMetric key are generated when required and discarded thereafter. During its lifetime, the ICMetric is never communicated even to trusted entities.

The purpose of the calibration phase is to extract suitable features for the ICMetric generation. The individual device features are extracted and processed to create a device ICMetric. The calibration phase is only applied once in the ICMetric

generation lifecycle. The individual steps in this phase are as follows:

- For each device, the desired feature values are measured; which will typically be based on operating/interaction characteristics of the associated software operating on the hardware device. Interactions of the user and environment with the device are also used as features for ICMetric generation. User content stored or utilized by the device can be incorporated into the ICMetric of the device.
- Generate feature distributions for each feature illustrating the frequency of each occurrence of each discrete value for a sample device. This will allow the same digital signature to be generated from the normal variations of operation of the device concerned but ensure any abnormal variation fails to generate the correct digital signature.
- Normalise the feature distributions generating normalisation maps for each feature. These essentially relate the range of measured values for a given device to a fixed range of values chosen for that device. These calibration maps are present in the memory.

The operation phase is applied each time an ICMetric encryption key is required for a given device. Once the key is used the device ICMetric, associated data and the keys are entirely removed from the system.

1. Measure features for the given device or software system for which an encryption key is desired.
2. Apply the normalisation maps to generate values suitable for key generation.
3. Apply the key generation algorithm which combines the normalised feature values into a single key.

2.4.2 Features for ICMetric Generation

Device features are a collection of characteristics of a specific device which can be used to differentiate it from other devices. The selection of features is so crucial that it has an impact on the reproducibility, security of the device ICMetric. If the wrong set of features are used, then the device identification may not be reproducible or may be easy to predict by an adversary. Device features can range from specific hardware details to the habits of the device user. Device features are the foundation of ICMetric to creating secure encryption keys. When conventional security systems encrypt data, they must create and store templates of the generated encryption keys to enable decryption of the data at a later time. The use of device features for ICMetric based key generation entirely removes the need to store the keys for any cryptographic service.

If sufficient distinct features of a device can be gathered, a highly secure unique key for the device can be generated. Much in the same way that a human can be differentiated from anyone else if enough of their physical features are measured. These features do not need to be recorded or stored as a template at any point during this process as they can be collected each time the system requires a cryptographic key. Hence the same cryptographic key will be produced, proving the devices authenticity and untampered state. However, if an adversary were to either clone the application to another device or tamper with the original device's environment, a different key would be created and access to the data will be denied.

As the features of a device are used to create the encryption key, the strength of the system lies primarily on the selection of robust and diverse features. There are certain aspects of each feature that need to be considered to determine its adequacy. Firstly, features should be able to achieve a certain level of obfuscation, either innately

or by conducting certain procedures on them. It is highly important that the core components that make up the key, are as obscure as possible from potential hackers. Even if a few features remain completely obscure; the key becomes almost impossible to fraudulently replicate. There are many ways to obscure features such as creating composite features, which record how certain device features interact with each other. Not only would the device features need to be discovered, but also their various interactions with each other.

Previous studies on ICMetric generation have tried to identify a wide range of suitable features. Earlier studies [53][54][55] on hardware features for ICMetric generation have shown that the program counter and cycles per instructions can be used to generate a device ICMetric. Similarly a recent study [56] has used the bias in MEMS sensors as a unique feature. The authors have used the accelerometer and gyroscope, and show statistically that there is a bias which sufficiently identifies an individual wearable health device. The authors establish a testbed of multiple sensors to show that each sensor has a unique and reproducible bias, provided the device and its internal environment has not been compromised. A good feature will have a high inter-sample variation thus offering many potential variants of it. An example of this would be a simple serial number that possesses a very high inter-sample variation, as each device's serial number is completely unique. Unfortunately, such serial numbers also possess a very low level of obfuscation. Thus, a features positive and negative attributes must be considered as a whole during the selection process.

Many features produce values that are non-static, they fluctuate as the device is in operation. An example of such a feature would be the current free RAM on the device. While the value should be quite stable it will change when applications are

opened (thus consuming RAM) or closed (freeing RAM). Intra sample variation describes how much these value change over the lifecycle of a device. At any given moment, a devices current free RAM will reside somewhere along the axis but will periodically move up and down. On its own this feature is too volatile to record over time; however, it can be managed if we create ranges of values, in this case ranges of 400MB. Each of these ranges are mapped to an arbitrary value and if the feature stays within the range its resulting arbitrary value will stay the same. For the feature to be stable enough to use in this way the intra sample variation must be as low as possible, as features with a high intra sample variation fluctuate too much. This becomes particularly difficult to map and ultimately cannot offer a persistent value to contribute to the key generation. The features of a system can be categorized into system specific, device settings and user data. The three feature categories are discussed below.

System features are a function of the internal environment of a system. When considering system features one may be tempted to use obvious features like a MAC address. The MAC address is a system feature but it can be considered weak since it can be easily captured by using common network sniffing tools. The features which are unique to the system but difficult to capture by an adversary can be used in the generation of ICMetric. All devices contain a CPU; and with variations of CPUs in devices, CPU info is used to establish a metric for ICMetric generation. The CPU hardware has several features that are used in the generation of the ICMetric; such as, the amount of memory allocated to buffers, the available memory, dirty memory, shared memory, MemFree, Exec_time, active memory, inactive memory, VmallocChunk, slab, and writeback.

Research [56] has shown that the ICMetric of a device can be generated using the internal settings of a device. This can be achieved by using serial numbers, calibration settings, onboard identification modules, Bluetooth identification, SSID etc. Device settings like the Bluetooth identification is modifiable therefore using only one device setting can compromise the security of the ICMetric implementation. Even though serial numbers of a device are unique they should be used with care since they often appear on the exterior of the device making spoofing them an effortless task. On the other hand an example of a strong serial identification chip is the DS-2411 [57]. This is a factory lasered single line chip which enables serial identification based services. Once the chip is lasered with a serial number it cannot be overwritten or altered in anyway thus deterring cloning, fabrication and spoofing based attacks. The ICMetric of a device can also be based on the calibration settings of a device. The calibration of a sensor is often based on multiple variables which makes this a strong feature for ICMetric generation. Identical devices that have been mass-produced by a single manufacturer will possess similar hardware and software environment. If the devices are operated by two different users, then the internal environment will be sufficiently different owing to the presence of unique user data. Again, it is important to base the ICMetric generation on features that are unique to a user and are not predictable by an adversary.

2.4.3 Feature Correlation

As correlated features are a combination of singular features therefore they provide a greater level of obfuscation compared to singular feature. This adds an extra level of depth when recreating the values, as they have to be generated and can't be read directly from a device. Each correlated feature can itself be used as a feature, which

has the benefit of increasing the entropy of the key, generated by the ICMetric algorithm. Additionally, the correlated features are likely to be more stable than the singular feature as they represent a relationship rather than coming from a specific range, meaning that there is less intra-sample variance thus increasing reproducibility of the generated key. Another significant aspect of correlated features is their ability to help distinguish devices; since singular features have a higher chance of having an overlap when the possible range for the feature is analysed across multiple devices.

2.4.4 Feature Sets and Fault Tolerance

The features of a device can be logically categorized into specific sets. The device features contained within these sets share similar traits or are affected by the same modifications and re-orientations of a device.

A practical implementation of ICMetric technology could potentially use hundreds of these feature sets. With such a large potential for scaling, it cannot be assured that every feature within every set will produce the same ICMetric result every time on a device. Fault tolerance must be created for these sets to create a system that not only produces a key that is highly unique but one that is also wholly reliable. This can be achieved for ICMetric technology through the use of the Shamir's secret sharing cryptographic algorithm. When using this algorithm, a key is generated and split between several entities, in this case one part for every feature set that is contributing features to ICMetric generation. The amount of these key shares required to reconstruct the key can be defined when using this algorithm; either that number of shares or more will be required in order to produce the same key again. Using this algorithm, enables how many of these feature sets are allowed to fail while still maintaining assurance of the systems security. In a case where hundreds of categories contribute to the key, if a

few set values change due to device modification or reorientation then the other feature sets will still be able to reconstruct the key with their shares. However, once enough sets fail, the system falls below the defined secret sharing reconstruction threshold, a different key will be produced and measures to protect the data would be taken. Furthermore, this algorithm can also be applied in the same way within specific sets, allowing a single set to contribute to the key even if it contains a volatile feature that could sometimes potentially fail.

2.5 SUMMARY

Transition from standalone devices to networked systems means that the importance of security can no longer be denied. Conventional views of computation were limited to laptops and computers. Technological advancements have resulted in a new class of systems called embedded systems. An embedded system is a complex system which has its own hardware, firmware and possibly an application software. An embedded system is often part of a larger system for example a temperature regulator in a chemical plant. Embedded systems are different from conventional computation systems owing to limitation in design. Often, they possess limited resources and processing capability which could mean that security may be considered a secondary system feature. To make the situation complex embedded systems are now network capable. Thus, a networked embedded system possesses the same design constraints with increased complexity.

Conventional systems rely on stored cryptographic keys. These keys can be captured through various methods which can result in the system being exposed. Often the security of an algorithm is proven by showing that the algorithm is intractable. Adversaries may not always attack a system by targeting the algorithm or the

mathematical basis. This chapter has illustrated this fact by presenting an in-depth study on side channel attacks which target the cryptographic keys. Effort has been made to show that a cryptosystem can be attacked without attacking the fabric (mathematical design). Owing to this, cryptographers consider other sources to establish a root of trust. This chapter has also studied the Physically Unclonable Function as a root of trust in cryptographic systems. Examples of different types of Physically Unclonable Function have also been presented in this chapter.

A relatively new root of trust which has applications in areas beyond security is the ICMetric technology. The ICMetric technology proposes using the features of a device to create an identification which is then used in cryptographic algorithms. This eliminates the need for stored keys thus deterring key theft. This chapter has studied the ICMetric technology in detail and shows how a device identification is created. Since the security of the ICMetric technology is primarily dependent on the selected features, therefore an in-depth study on suitable features has also been presented in this chapter. Hence features which describe hardware properties, system settings and user data have been discussed in this chapter.

The individual device features are collected and processed to create keys which can be used for the provision of cryptographic services. Many cryptographic services require keys for the provision of security. Therefore, the upcoming chapter demonstrates how a device ICMetric can be used for creating a key by using the PBKDF algorithm.

3

ICMetric Strong Key Derivation

3.1 INTRODUCTION

Cryptographically secure applications require authentication of users/devices for secure communication to take place. The maintenance of user passwords constitutes a significant factor related to the provided security of a service. A study of security breaches shows that massive amounts of user data is captured thus harming the reliability of the application provider [58].

The notion of trustworthy authentication revolves around using appropriate keys of sufficient strength for carrying out secure operations. This primarily depends on the generation and protection of appropriate cryptographic keys, so that there is no threat to confidentiality and availability of the cryptosystem. Without appropriate generation and handling of keys; they could be guessed, modified, or substituted by unauthorized personnel which could then lead to the interception of sensitive communications.

A strong key having high entropy and sufficient length is the basic requirement for the secure working of any cryptosystem. The ICMetric cannot serve as a useful key for cryptographic operations in its raw form [15]. It may have insufficient length or low entropy, thereby making it easy for the attacker to guess and compromise the system. In this chapter, the problems associated with a weak ICMetric are identified, thereby discussing the potential threats linked with the use of a weak ICMetric in a cryptosystem. Keeping in mind the threats associated with using weak ICMetric keys, a strong ICMetric key generation scheme is proposed that can potentially improve the strength of the ICMetric and make it suitable for use in cryptographic operations. The presented approach for the generation of strong ICMetric key, results in the generation of cryptographic keys possessing properties of sufficient key entropy and length. This scheme aims to resolve the issue of weak keys and make the weak ICMetric key suitable for use in secure cryptographic operations.

A major goal of the strong ICMetric key generation scheme is to safeguard the ICMetric keys from pre-computed attacks; such as rainbow table attacks [59], dictionary attacks or brute force attacks. The proposed scheme safeguards the ICMetric from brute force, rainbow table and dictionary attacks; which is very critical since compromise of the ICMetric itself can potentially mean the compromise of the device and all its subsequent operations. As the ICMetric of a device is its unique possession therefore a compromised ICMetric cannot be just replaced with another ICMetric.

3.1.1 Key Length

Key length directly translates into security of the system. The key length is measured in bits and each bit of the key increases the security of the cryptosystem, thereby exponentially reducing the likelihood of a brute force attack being successful

against the key. On the contrary an addition of every single bit to the cryptographic key slows down the cryptosystem [60]. Therefore, the key length must be tuned very carefully, observing the practicality of the cryptosystem and the required level of security. Different cryptosystems require different key lengths for security. For instance, modulo based public key systems such as Diffie Hellman and RSA require rather long keys (generally around 1024 bits), whereas symmetric systems, both block and stream ciphers use shorter keys (generally 256 bits). Most block ciphers use only one key length, but asymmetric ciphers use a varied number of key lengths.

3.1.2 Key Entropy

Secrecy of keys corresponds to the lack of knowledge that an adversary has about the keys. All keys must have some amount of entropy to remain secure. The entropy is used to measure the difficulty of guessing cryptographic keys. This generally means that if all the bits of a key are not equally random, then the system is much more vulnerable to being attacked. In an ideal uniformly distributed random key generation scheme, all bits of the key are independent of each other. The concept of Shannon entropy was introduced to estimate the uncertainty of a random variable. The output is the number of bits, on average, required to describe the random variable. The bits are independently generated, the entropy of the key is the sum over the entropy of the individual bits and is given by Shannon entropy [61].

Definition 3.1 (Shannon entropy) [61]. For a random variable X with k outcomes $(x_1, x_2, \cdots, x_k)$, the entropy is defined as,

$$H(X) = -\sum_{i=1}^{k} P_r(x_i)\, log_2\, P_r\,(x_i)$$

where P_r is the probability. As the entropy of the random variable is expressed in bits, hence logarithm base 2 is used in the formula. The following calculations use the chance $P(x)$ of a zero or one bit as 0.5, as both are equally likely.

$$-2(0.5\ log_2(0.5)) = -log_2(0.5) = -log_2(2^{-1}) = 1$$

Also, from the proof of the additivity of Shannon entropy [62], it follows that the entropy of all the bits should be equal to the sum of the entropy of the key. Therefore, the ideal Shannon entropy of a theoretically perfect random bit generating scheme can be simplified to the formula key_size. This is commonly referred to in the literature in the units of 'bits per byte', with 8 bits of entropy per byte the ideal score.

Therefore, a goal of all real-world strong random key generation systems is to have very high entropy and be as close to the theoretical maximum as possible. However, it is very difficult to come up with practical strong key generation algorithms that achieve near perfect entropy scores.

3.1.3 Randomness

There are a lot of randomness tests existing in literature, based on techniques and measures based on statistical tests, transforms, and complexity or a mixture of these. For example, NIST (National Institute of Standards and Technology) has defined in 'NIST Special Publication 800-22' an extensive set of statistical tests for verifying a source of randomness [63]. Similarly, the diehard tests [64] are a battery of statistical tests for measuring the quality of randomness. Federal Information Processing Standards (FIPS) 140-2 publication [65] for cryptographic modules also specifies statistical tests for randomness. Instead of making the user select appropriate significance levels for these tests, explicit bounds are provided which the computed

value of a statistic must satisfy. A single bit stream of 20000 bits, output from a generator, is subjected to the tests as tests on shorter bit streams are not statistically conclusive. The description of some of the most common tests is given below:

- **Entropy**: As discussed earlier, it is the information density of the bits of the key, expressed as a *number of bits per byte*. A result close to the ideal value i.e., 8, indicates that the key is extremely dense in information, therefore essentially random [66]. However, as we will see shortly, the results are far from ideal on bit streams of shorter lengths that are more suitable to be used as keys i.e., 128, 256 bits.
- **Chi-square Test** [67]: This is the most commonly used test for the randomness of data, and is extremely sensitive to errors in pseudorandom sequence generators. The chi-square distribution is calculated for the stream of bytes in the key. It is expressed as an absolute number and a percentage which indicates how frequently a truly random sequence would exceed the value calculated.
- **Arithmetic Mean**: This is simply the result of summing all the bytes in the key and dividing by the key length. If the data are close to random, this should be about 127.5. If the mean departs from this value, the values are consistently high or low.
- **Monte Carlo Value for Π** [68]: Each successive sequence of six bytes is used as 24 bit X and Y co-ordinates within a square. If the distance of the randomly-generated point is less than the radius of a circle inscribed within the square, the six-byte sequence is considered a "hit". The percentage of hits can be used to calculate the value of Π. For very large streams (this approximation converges

very slowly), the value will approach the correct value of Π if the sequence is close to random.

- **Serial Correlation Coefficient** [67]: This quantity measures the extent to which each byte in the key depends upon the previous byte. For random sequences, this value (which can be positive or negative) will, of course, be close to zero. A non-random byte stream will yield a serial correlation coefficient on the order of 0.5. Wildly predictable data such as uncompressed bitmaps will exhibit serial correlation coefficients approaching 1.

3.2 FUNDAMENTALS OF CRYPTOGRAPHIC KEYS

Cryptographic keys vary in strength and some are stronger than others. The weakness in cryptographic keys exists owing 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. The following section explains the major threats associated with the usage of weak keys in security applications.

3.2.1 Weak Keys

A threat in security applications that targets weak/low entropy keys is brute force/exhaustive search attack [9]. A brute force attack is a common method adopted by adversaries to compromise cryptosystems that are based on strong cryptographic operations with weak keys; whereby instead of finding weaknesses in the encryption system, the attacker tries to crack the cryptographic key used for the provision of cryptographic services. Brute force attacks try to discover a valid key by trying out different possible key combinations [9]. The attacker tries each possible key combination to find the correct key which has been employed for carrying out the

cryptographic operations. From an adversaries perspective, longer keys are harder to break compared to shorter keys, since the resources required to launch a brute force attack grow exponentially with an increase in key size [69][70].

A successful brute force attack allows an attacker to find the cryptographic key, thereby breaking into the cryptosystem. Brute force attacks can be prevented through multiple methods [71], one of which is by using strong keys of sufficient entropy and length for secure cryptographic operations. Strong keys of sufficient length hinder the attacker's ability to launch a brute force attack by using a large key space, and making it impossible for the attacker to recover the key in a reasonable amount of time. The notion of a successful brute force attack is to find the key that has been used to perform ciphering operations, thereby deciphering all the encrypted data using the found key to recover the plaintext [72].

A dictionary attack [60] is a guessing attack which uses precompiled list of words to break the key. A dictionary attack tries to break passwords by trying out real word combinations which are likely to work rather than random strings of numbers. The password guesses in a dictionary attack are often based on some information already known to the adversary about the target. This could include information based on the details of the target, observed patterns or globally likely answers. The execution time of a dictionary attack depends on the key combinations in the dictionary list and the number of keys tested per unit of time. This means that a possible password could be missed, if it was not included in the dictionary list.

Rainbow table attacks target schemes that compute cryptographic hashes for calculating the cryptographic keys [72]. These attacks are launched by constructing rainbow tables that store a hashed value for every word in the key dictionary. Rainbow

table attacks pre-compile a series of hashes and their corresponding keys for key lookup. Each rainbow table entry is started with a randomly selected input and its corresponding hash. Rainbow table entries are computed by chains of hash functions and reduction functions that transform a random input value to its final hash value [72]. A hash function is a one way function that takes as input a plaintext to generate a hash value, while a reduction function in a rainbow table attack does the reverse to transform a hashed value to its respective plaintext; these characteristics enable storage of the pre-computed hashed keys in a compact manner [73]. However, a reduction function is not the inverse of a hash function, so computation of a reduction function on a hash value generates a completely new plaintext value. Rainbow tables store the starting plaintext and final hash value after performing multiple iterations of hash and reduction functions. Rainbow table attacks are more efficient as compared to brute force and dictionary attacks, and are therefore able to crack the cryptographic keys quickly. Rainbow tables store pre-computed hashes for all possible key combinations, so that the key can be efficiently cracked by hash and corresponding key lookup.

3.2.2 Key Strengthening

An authentication server authenticates users by using stored data which is then used to validate a password. A basic setup would be to store the keys themselves, and validation would be a simple comparison. But in this setup, if an attacker gets a glimpse of the stored keys then the whole server would be compromised. The list of the stored keys is used by the attacker, who can use them to launch an offline pre-computed attack. These types of attacks are unavoidable; therefore, techniques such as key derivation functions that are based on hash functions are employed that can strengthen the key and make them as stronger against attacks.

Hash functions are non-invertible mathematical functions that are very efficient. To safeguard the sever passwords from being compromised, the passwords are stored in hashed format that don't reveal the password in its plaintext form. Cryptographic hash functions take an arbitrary block of data and return a fixed-size message digest, the cryptographic hash value, such that any accidental or intentional change to the data will change the hash value, thus alerting the receiver about the modification. Therefore, the server can validate the password by computing hash on the presented password and comparing it against the stored hashes. A cryptographic hash function h, is designed to withstand pre-image and collision attacks:

- Pre-image resistance: it is computationally infeasible to find an input x that hashes to the output y, i.e., given y, it is difficult to find x such that $h(x) = y$.
- Collision resistance: It is computationally infeasible to find any two distinct inputs x and x_0 which hash to the same output, i.e., such that $h(x) = h(x_0)$.

Hash functions are used in a wide variety of applications including the construction of other cryptographic primitives such as message authentication codes (MACs), digital signatures, pseudo random functions (PRFs) and pseudorandom number generators (PRNGs). The security of these applications relies on the cryptographic strength of the underlying hash function used and its resistance to attacks. Widely used hash functions include MD5 [74], SHA-1 [75] and SHA-256 [76]. Today, it is practically feasible to find collisions in MD5 [73], and hence MD5 should not be considered for constructing other cryptographic primitives. SHA-1 collision resistance has also been under attack [73].

MD5 is a hashing algorithm which is still widely used but cryptographically flawed as it's prone to collisions. MD5 is broken in regard to collisions, but is still pre-image resistant. The first attacks on MD5 were published in 1996 [77], this an attack on the compression of MD5 rather than MD5 itself. In 2004 a theoretical attack was produced which allowed for weakening the pre-image resistance property of MD5 [78].

SHA or Secure Hashing Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS). Currently three algorithms are defined:

- SHA-1: A 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1, and the standard was no longer approved for most cryptographic uses after 2010 [78][79].
- SHA-2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32 bit words where SHA-512 uses 64 bit words. There are also truncated versions of each standardized, known as SHA-224 and SHA-384. These were also designed by the NSA.

A major problem with hashing unsalted keys is the possibility of a rainbow table attack being launched, that can break the key in an instant, since a hashed key always corresponds to a specific original key. Used correctly, salts can thwart all pre-computed attacks, since a password can have varying associated salts resulting in different hash values. A salt is a randomly generated value that is appended to the user's password

before hashing. Salts are used because users tend to choose the same passwords which are not random in nature. Often chosen passwords are short real words that are easy to remember, but this also enables several attacks. Passwords are generally not stored in cleartext, but rather hashed. So, we now have a process for storing passwords in our system in a secure form that cannot be decrypted, thus closing the door that allows attackers access to all the passwords stored in the system.

The salt is stored in the database with the user's hashed password. The addition of a random salt value to the password lowers the probability that the resultant hash-value will be found in any pre-calculated table. A common way to combine the salt and the password is to simply concatenate and hash them [60] i.e., $h(salt \parallel password)$. A salt is used to prevent pre-computed attacks such as brute force attacks and rainbow table lookups. When brute forcing, the attacker can't generate all possibilities in one run, he/she would need to brute force for each password taking the specific salt into account, which is a very costly operation. Salts also prevent people from using rainbow table attacks, as the table would need to be generated, per password, based on the salt.

With time computers are becoming faster and faster, this means the attackers can try more and more potential passwords within the same time frame. To counter such a trend, hashing can be made slow by defining hash functions to use several iterations of the hash function. A hash function alone has its limitations; therefore, a combination of the above-mentioned ingredients is needed to thwart attacks on keys.

3.2.3 Key Derivation Functions

A key derivation function derives secret keys from secret value using a pseudorandom function. Key derivation functions make use of traditional key strengthening ingredients to derive longer strengthened cryptographic keys. A key

derivation function is used in applications to derive keys from secret passwords, which typically don't have the desired properties to be used directly as cryptographic keys. These properties include sufficient length and entropy as mentioned in section 3.1 and 3.2. Key derivation functions make use of key strengthening based on the key and a random salt to derive longer stretched keys. A key derivation function (KDF) is generally represented as

$$D_k = KDF(key, salt, iterations) \tag{3.1}$$

where the key, salt and the number of iterations are passed as parameters to a key derivation function to derive a strengthened key D_k. Key derivation functions are deliberately made slow with a high iteration count to delay the successful launch of pre-computed parallel attack on the passwords. The section below explains some of the state of the art key derivation functions.

PBKDF2 [74] is part of the RSA laboratories cryptographic standards. PBKDF2 is well recognised by NIST SP 800-132 and implemented in various frameworks. It came as a replacement to the earlier key derivation function PBKDF1, which could produce derived keys up to 160 bits long. PBKDF2 applies a pseudorandom function to the input password with a salt, and repeats this value several times based on the iteration count to produce a strengthened derived key. This derived key can be used as a cryptographic key in subsequent operations. The number of iterations directly translate into more computational work and makes password cracking more difficult. It is considered secure when used with a proper iteration count and cryptographic hash function such as SHA-256 and SHA-512.

PBKDF2 is designed to be computationally intensive, so that it takes a relatively long time to compute. Legitimate users only need to perform the function once per

operation (e.g., authentication), and so the time required is negligible. However, a brute-force attack would likely need to perform the operation billions of times, at which point the time requirements become significant. PBKDF2 has very low resource demands, meaning it doesn't require elaborate hardware or very much memory to perform, thereby being well suited for resource constrained environments. However, it also allows an attacker with sufficient resources to launch a parallel attack by building hundreds or even thousands of implementations of the algorithm in hardware and having each search a different subset of the key space. This divides the amount of time needed to complete a brute-force attack by the number of implementations available. It has a configurable output length that makes it very useful for being used in various applications.

Bcrypt is derived from the Blowfish [80] and was designed by reusing and expanding elements of the block cipher. The password can be upto 56 bytes and the produced output size is fixed to 192 bits. To handle longer input passwords, Bcrypt must be combined with another hash function which can bring the size of the input password to 56 bytes. The iteration count is a power of two and as with blowfish it needs certain amount of memory space for its look up tables that are initiated in memory. This means a certain amount of memory space needs to be used before a hash can be generated. This can be done on CPU, but for resource constrained devices it will become a lot more cumbersome due to memory restrictions. Bcrypt is slightly stronger than PBKDF2 at defending attacks on ASICs and GPUs. Bcrypt can easily be optimized by attackers with FPGA. FPGA chips have a lot of small embedded RAM blocks which are very convenient for running many Bcrypt implementations in parallel within one chip thereby compromising the passwords.

Scrypt is another key derivation function which has the same properties as Bcrypt, except that when the rounds are increased, it exponentially increases calculation time and memory space required to generate the hash. Scrypt is designed to require a large amount of memory to compute efficiently. Scrypt was created as response to evolving attacks on Bcrypt and is completely infeasible when using FPGAs or GPUs due to memory constraints. Scrypt requires the storage of a series of intermediate state data called “snapshots”, which are used in further derivation operations. These snapshots, stored in memory, grow exponentially when rounds increase. Hence adding a round, will make it exponentially harder to brute force the password. Scrypt is still relatively new compared to Bcrypt and has only been experimented with for a couple of years, which makes it less vetted than Bcrypt.

Although Scrypt function is designed to prevent parallel attacks owing to the huge resource demands of the algorithm. However, the algorithm is designed to use a large amount of memory making it infeasible for embedded system applications or systems with limited resources.

3.3 THE ICMETRIC

3.3.1 Threat Model

In the following section three scenarios are presented to further elaborate on the threat model of brute force, dictionary and rainbow table attacks that can be launched on weak ICMetric keys. These three threats depicted in Figure 3.1, show an attacker who tries to launch these pre-computed attacks on various devices making use of weak ICMetric keys.

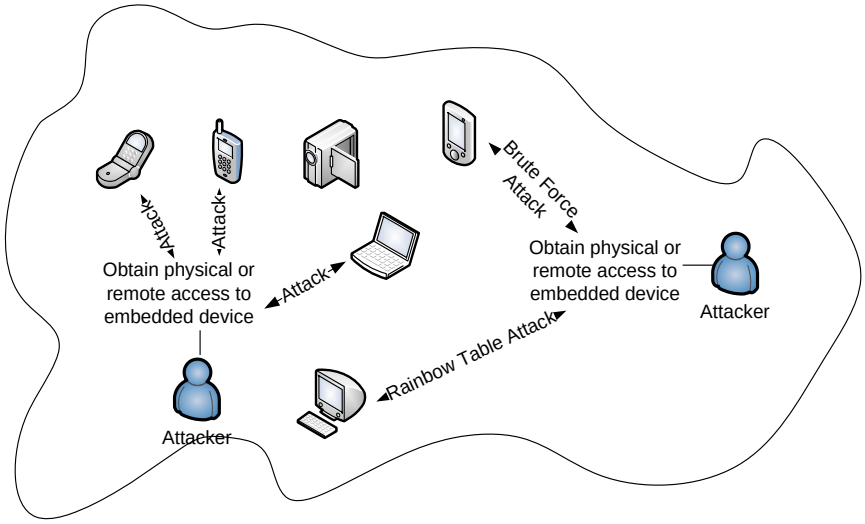


Figure 3.1. Threat Model for Weak ICMetric Keys

The first major threat to a device ICMetric is the possibility of being cracked by brute force attack. The ICMetric key might be short in length and low in entropy, so the adversary's goal is to break the ICMetric key using a software which tries different possible character combinations in quick succession, until the ICMetric key is found. The brute force algorithm uses a trial and error technique, which tests several key combinations to find the ICMetric key used for cryptographic operations. This results in the adversary being able to decrypt all information destined to the embedded system device. To launch a brute force, attack the attackers make use of a high-performance computer. The adversaries machine tries out a large number of key combinations at a very fast pace to recover the ICMetric key in a short period of time. A brute force attack is more likely to break the password if no other attack is successful. Here it must be mentioned that as the entropy increases the number of possible keys also increases. Thus, making it difficult for an attacker to brute force.

Another threat to cryptographic keys is the possibility of being cracked by dictionary attacks. A dictionary attack consists of trying every word in the dictionary as a possible password. A dictionary of possible passwords is formally referred to as a

wordlist [81]. A dictionary attack is generally more efficient than a brute force attack, since it may make use of many large dictionaries. Dictionary attacks often make use of string manipulations on the words found in the dictionary. This involves trying the wordlist backwards, with letter or number replacements or with different capitalizations. A small dictionary can sometimes lead to the dictionary attack being launched very quickly particularly when the user password is weak.

However, dictionary attacks are less successful against passphrases. Passphrases are composed of several meaningful words separated by spaces and are typically quite long. Dictionary attacks on ICMetric can be made highly impracticable by choosing several random words in the passphrase, and selecting at least one word that cannot be found in any dictionary significantly increases passphrase strength, thereby making it impossible to break the ICMetric.

In the third scenario of cracking an ICMetric key, the attacker tries to launch a rainbow table attack on the hashed strong ICMetric key. To launch a rainbow table attack, the attacker generates a large database of all possible key combinations and their corresponding calculated hash values. Depending on the length of the key and the employed character set, the attacker pre-computes many hash values and then stores the key and corresponding calculated hash value in the form of rainbow tables. If the key employed in the cryptosystem is too long or uses a character not part of the character set known to the attacker, then the generated rainbow tables are completely useless and the key cannot be cracked with the generated rainbow tables available to the attacker. To successfully launch a rainbow table attack on the hashed strong ICMetric key, the adversary searches the rainbow table for the hashed key and its corresponding plaintext key. If the hashed key is found in the rainbow table, then the attacker is successful in finding the original key; otherwise the attacker is unsuccessful

since the hashed value might not be present in the rainbow table.

The ICMetric is a fingerprint of the device and its compromise implies that the device has been compromised. In this case the greatest strength of the ICMetric technology becomes its greatest liability. It is a fact that a device's ICMetric data does not change over time, it remains same throughout the life of the device. This means that should a set of ICMetric data be compromised, it is compromised forever. A device ICMetric is based on only a limited number of reproducible ICMetric features. For authentication systems based on physical tokens such as keys and token, a compromised token or key can easily be cancelled and the user can be assigned a new token. Similarly, user ids and passwords can be changed as often as required. But if the ICMetric is compromised, the device will run out of features to be used for authentication. Owing to this reason the ICMetric is based on features which are not easy to predict or spoof by an adversary. To further ensure secrecy the ICMetric is not stored on the system and never communicated even to trusted parties.

3.3.2 Security Goals

Given the threat model, the ICMetric strong key generation scheme focuses on the following security goals:

- Length of the ICMetric Key – a fundamental goal of the strong ICMetric key generation scheme is to generate strong keys that have sufficient length and can be safely used in security critical applications. The sufficient length of the key protects it from easily being compromised.
- Entropy of the ICMetric Key – another essential security goal of the strong key generation scheme is to generate strong ICMetric keys that have high entropy.

The property of high entropy ensures that the keys can't be easily compromised using pre-computed attacks.

- Hardening against Brute Force Attacks – a vital goal of the strong ICMetric key generation scheme is to prevent the successful launch of brute force attacks. With a brute force attack, the attacker will try all possible combinations of characters, until a successful attack is launched.
- Hardening against Dictionary Attacks – a goal of the ICMetric strong key generation scheme is the mitigation of dictionary attacks. Dictionary attacks are faster than brute force attacks and use word lists as mentioned in section 3.2. If the hash of a word possessed by the attacker, hashes to the same hash which is maintained in the database then it can be said that the ICMetric is known to the attacker. Therefore, accomplishing this goal means the attacker can never map the hashed values to the original ICMetric.
- Hardening against Rainbow Table Attacks – a necessary goal of the proposed scheme is to prevent rainbow table attacks. Rainbow table attacks are a consequence of using hash functions to generate strong ICMetric keys and is significantly faster as compared to a brute force or dictionary attack. The strong keys are generated using random salts which prevent the attacker from launching pre-computed attacks.
- Deterring compromise of the device ICMetric – a crucial security goal of the designed ICMetric strong key generation scheme is deterring the effects of a successful brute force, rainbow table or dictionary attack on the original ICMetric. This is very crucial to the secure functioning of the device, since compromise of the ICMetric will result in the compromise of device's

fingerprint. The ICMetric is a fingerprint of the device and if it's compromised, all subsequent cryptographic operations carried out by the device would also be compromised. Fulfilling this security goal means that the proposed scheme doesn't reveal the original ICMetric, even if the generated strong ICMetric key is compromised by a pre-computed attack on the strong key. If an ICMetric key is compromised, then the key can be revoked and a new one issued thus preventing communications on the captured key. Therefore, the ICMetric should never be compromised during the working of the strong ICMetric key generation scheme.

- Adaptability – The adaptability of the ICMetric strong key generation scheme, to the key length requirements of various applications is another central requirement. The proposed strong ICMetric key generation scheme acts as a bridge between the device and the application layer. This requirement is very critical for the successful integration of the ICMetric technology with other applications.

3.3.3 ICMetric Strong Key Generation Scheme

The proposed strong key generation scheme is a novel scheme and aims to improve the security of the ICMetric based cryptosystem by proposing a scheme that generates a strong ICMetric key for use with symmetric or asymmetric cryptosystems. The proposed scheme 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 scheme has been developed following this critical requirement and the design

principles of the ICMetric technology. At no point during the working of the strong key generation scheme, can the attacker capture the ICMetric. Therefore, the proposed scheme generates a strong ICMetric key that is strong against attacks such as brute force, dictionary and rainbow table attacks; while also deterring the possibility of ICMetric compromise.

The proposed ICMetric strong key generation scheme is comprised of two main phases; Step 1 – the sub-key generation phase and Step 2 – the strong key generation phase, with both phases having further sub-phases. Figure 3.2 shows a general model of the proposed ICMetric strong key generation scheme, depicting a generalization of the components that become part of the scheme design.

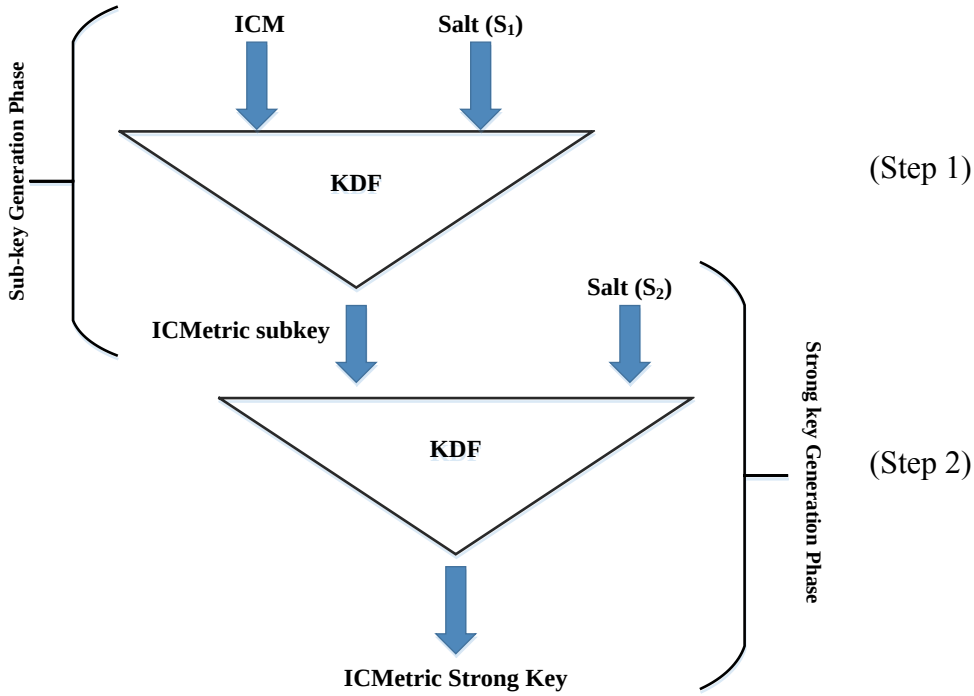


Figure 3.2. General Model of the ICMetric Strong Key Generation Scheme

Both steps 1 and 2; the sub-key generation phase and the strong key generation phase respectively, 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. In Step 1, the device ICMetric and 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 Step 2 of the scheme, the ICMetric sub-key and the random salt $'S_2'$ become input to the chosen key derivation function (KDF) and generate the final ICMetric strong key.

The novelty of the scheme lies in the fact that the proposed scheme confuses and diffuses the ICMetric of the device in such a manner that it is not possible for the attacker to recover the original ICMetric. This is accomplished by 10000 iterations of each function and 128 bit salts fed to SHA-2, which completely hinder the ability to revert back to the original ICMetric. The scheme even safeguards against rainbow table attacks, which was not possible using only KDF and salts in literature, thereby making the proposed scheme a very secure option for use with the ICMetric technology.

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 scheme.

Step 1 – Step 1 of the ICMetric strong key scheme is responsible for generating an ICMetric sub-key that will be used as input in Step 2 of the scheme. Step 1 is comprised of three sub-steps namely Step 1.1, Step 1.2 and Step 1.3 respectively that are outlined in Figure 3.3.

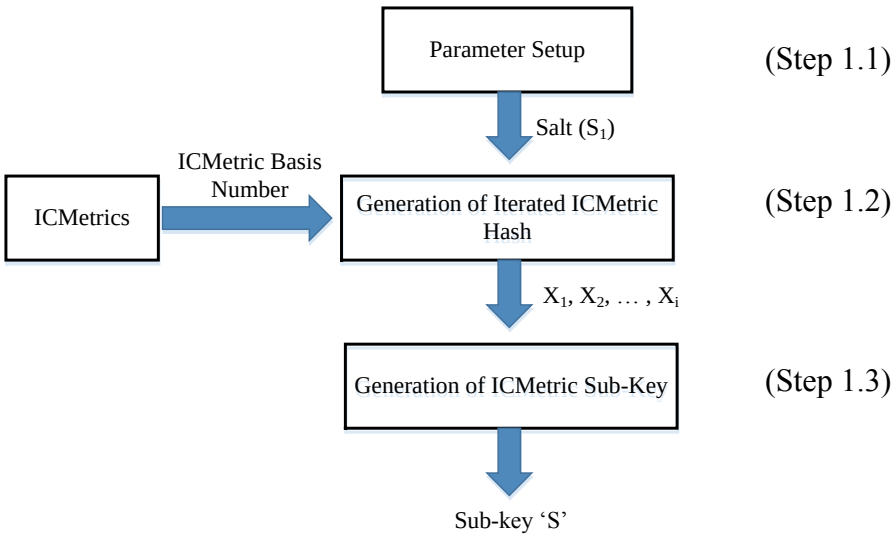


Figure 3.3. ICMetric Sub-Key Generation

The blocked diagram above shows how the generation of iterated ICMetric hash is connected to rest of the sub-components in the generation of the ICMetric sub-key. It is evident from the figure that in Step 1.1; the ICMetric and the salt (S_1), become input to the 'Iterated ICMetric hash generation' module. In Step 1.2, the generated iterated ICMetric hashes become input to the 'ICMetric sub-key generation' module, which after performing computations on the input iterated ICMetric hashes, generate an ICMetric sub-key $'S'$ in Step 1.3, as shown in Figure 3.3. The step wise explanation to the working of the scheme is given below:

Step 1.1 of the scheme requires the device to generate an ICMetric based on the extracted feature values. In [50], Papoutsis et al. proposes two possibilities for combining the extracted feature values of a device. The extracted feature values can be combined by concatenating or by adding all the feature values to generate an ICMetric. The authors also show that an ICMetric generated as a result of concatenation of extracted feature values creates a long secret key but the secret key is not stable. Whereas, the ICMetric generated as a result of addition of feature values is short but is

more stable. The ideal case would be to have an ICMetric that is stable as well as long, so that it qualifies as a secure key for use in various cryptographic operations. The proposed strong key generation scheme aims to strengthen a short ICMetric, that was generated as a result of addition of feature values rather than concatenation, so each of the extracted feature values are added to generate an ICMetric.

Therefore, in the key setup algorithm the device in question generates its ICMetric ICM as outlined in chapter 2.

$$ICM = \text{ICMetric generated using features of Device} \tag{3.2}$$

The ICMetric of the device can be generated based on the system features, device settings or user data, depending on the particular application scenario in question. The framework is responsible for setting up various input parameters for the working of the strong ICMetric key generation scheme. Table 3.1, lists the parameters that are required in the setup of the scheme.

Table 3.1. Description of Symbols/ Input Parameters in the Scheme

Symbol	Description
ICM	ICMetric of device
$\|\|$	Concatenation
$\oplus$	Bitwise exclusive OR
$Salt_1, Salt_2$	128-bit random salt values
$SHA2(256/512)$	The selected SHA2 variant
N	Number of iterations (10,000)
L	Required length of the strong key (256/512/1024/2048 bits)
S	ICMetric sub-key
M	ICMetric strong key

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 [74][82] 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. The random salts are generated using timestamp as the seed to the random number generator. As shown in Table 4.1, the salt values $Salt_1$ and $Salt_2$ employed by the proposed ICMetric strong key generation scheme are 128-bit long.

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 4.1. 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". However, the latest NIST guidelines (June 2017) recommend having 10,000 iterations for sufficient security [83].

Step 1.2 of the algorithm is responsible for performing N iterations on the device ICMetric by using the selected SHA2-variant and the salt value $Salt_1$. The number of iterations in the proposed strong ICMetric key generation scheme 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 and a 128-bit random salt $Salt_1$ 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 number 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 128-bit salt and icm are combined via SHA-2, as shown in the set of equations (3.3). 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-bit random number serves as a salt value for the key stretching function and safeguards against rainbow table attacks. By adding a salt value to the ICM, the possibility of using pre-computed hashes for attacks is reduced.

The purpose of the salt 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 $i \in \mathbb{N}$ then

$$\begin{aligned} I_1 &= SHA2(ICM, Salt_1) \\ I_2 &= SHA2(ICM, I_1) \\ &\vdots \\ I_{10,000} &= SHA2\left(ICM, I_{9,999}\right) \end{aligned} \tag{3.3}$$

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

$$X_i = I_1 \oplus I_2 \oplus \cdots \oplus I_N \tag{3.4}$$

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.

This step of the algorithm does a stretching operation similar to the iterated hash function however the input parameters are different. The sub-key generated at the end of step 1.3 and salt $Salt_2$ are fed as input to the N iterations of the selected SHA2 variant. The process begins with the generated sub-key $'S'$ and a 128-bit random salt $'Salt_2'$ passed through the selected SHA2-variant function. 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, 10,000 iterations are carried out using

the sub-key $'S'$ and $salt_2$, resulting in hashed values $H_1, H_2, \ldots, H_{10,000}$ as outlined in the set of equations (3.7).

$$\begin{aligned} H_1 &= SHA2(S, Salt_2) \\ H_2 &= SHA2(S, I_1) \\ &\vdots \\ H_{10,000} &= SHA2\left(S, I_{9,999}\right) \end{aligned} \tag{3.7}$$

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

$$SX_i = H_1 \oplus H_2 \oplus \cdots \oplus H_N \tag{3.8}$$

which gives 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. Exclusive OR adds an extra layer of protection but the actual security derives from the iterations of the hashing function and the salt value.

In Step 1.3, 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 $'S'$.

$$S = X_1 \parallel X_2 \parallel \cdots \parallel X_{L/256} \tag{3.5}$$

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'.

$$S = X_1 \parallel \cdots \parallel X_{L/512} \tag{3.6}$$

The ICMetric strong key generation scheme can derive strong ICMetric keys of specified length. It generates as many blocks as required to achieve the required key length. Each block is produced by iterating the key stretching function 10,000 times. Once generated, the required blocks are concatenated to create the required sized key, which gives the ICMetric sub-key that has sufficient length and entropy.

Step 2 – Step 2 of the ICMetric strong key scheme is responsible for generating an ICMetric strong-key based on the sub-key derived from Step 1 of the scheme. Step 2 is also comprised of three sub-steps namely Step 2.1, Step 2.2 and Step 2.3 respectively that are outlined in Figure 3.4.

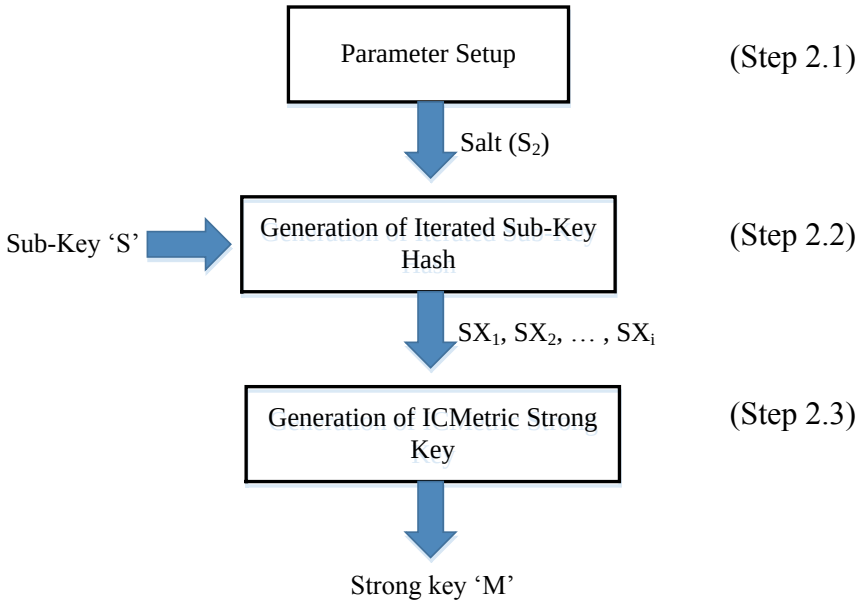


Figure 3.4. ICMetric Strong Key Generation

The blocked diagram in Figure 3.4 shows how the generation of iterated sub-key hash is connected to rest of the sub-components in the generation of the ICMetric strong-key. It is evident from the figure that in Step 2.1 the ICMetric sub-key and the salt (S_2), become input to the 'Iterated sub-key hash generation' module. All further

operations in the Step 2 are based on these two values. The ICMetric sub-key is the output of the Step 1 and used in Step 2 as it is. While the salt value is a 128 bit random salt that is generated based on the rules outlined in Step 1.2 above. In Step 2.2 the iterated sub-key hashes are computed based on the output of Step 2.1, fed to the algorithm outlined in Step 1.2. In Step 2.3, the output of Step 2.2 becomes 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 3.4. In Step 2.3, 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 $'S'$.

$$M = SX_1 \parallel SX_2 \parallel \cdots \parallel SX_{L/256} \tag{3.9}$$

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'.

$$M = SX_1 \parallel \cdots \parallel SX_{L/512} \tag{3.10}$$

The ICMetric strong key generation scheme 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 10,000 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 at the end of Step 2.

3.4 IMPLEMENTATION AND EVALUATION

A working prototype of the proposed strong key generation scheme was implemented using C on Linux. The scheme has been implemented on a first generation Intel Core i3 3.2 GHz processor with 6 GB RAM using the OpenSSL cryptographic library [84]. The scheme aims to generate strong ICMetric keys for carrying out cryptographic operations.

The strong ICMetric key generation scheme has two counterparts, based on SHA-256 and SHA-512 respectively. These counterparts each have four strong key variants to facilitate operation with varying key sized cryptosystems. The strong ICMetric key generation scheme counterpart based on SHA-256 supports four strong ICMetric keys size variants i.e. 256, 512, 1024 and 2048 bits; with a 128-bit device ICMetric as input. Similarly, the strong ICMetric key generation scheme counterpart based on SHA-512 supports a 256 bit, 512 bit, 1024 and 2048 bit strong ICMetric keys respectively. The two unique salt values used in the implementation of the strong ICMetric key generation scheme are each 128-bit long. These salts are generated as a sequence of random bytes using timestamp as a seed to the random number generator.

This section evaluates the efficiency of the scheme design by measuring the RAM consumption and execution time of the implemented scheme. The runtime has been used for evaluating the execution time of the system, whereas Valgrind [85] 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 [86]. 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 strong ICMetric strong key generation scheme.

3.4.1 ICMetric Strong Key Generation Scheme using SHA-256

This section evaluates the performance of the ICMetric strong key generation scheme using SHA-256. The strong ICMetric key generation scheme using SHA-256 supports four strong ICMetric key variants to facilitate operation with varying key sized cryptosystems.

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

To prove the scalability of the strong key generation schemes, the time taken for the generation of keys of varying sizes has been considered. Figure 3.5 shows the experiment number versus time taken graph for the proposed strong key scheme variants, namely 256, 512, 1024 and 2048 bits. The graph in Figure 3.5 shows a time consumption comparison for the generation of 256 bit strong ICMetric key, 512 bit strong ICMetric key, 1024 bit strong ICMetric key and the 2048 bit strong ICMetric

key; using the proposed ICMetric strong key generation scheme. The time consumed in the generation of each strong ICMetric key is measured in microseconds. It is evident from the graph that no matter which SHA variant is used; an increase in the key size brings a change to the time performance of the application.

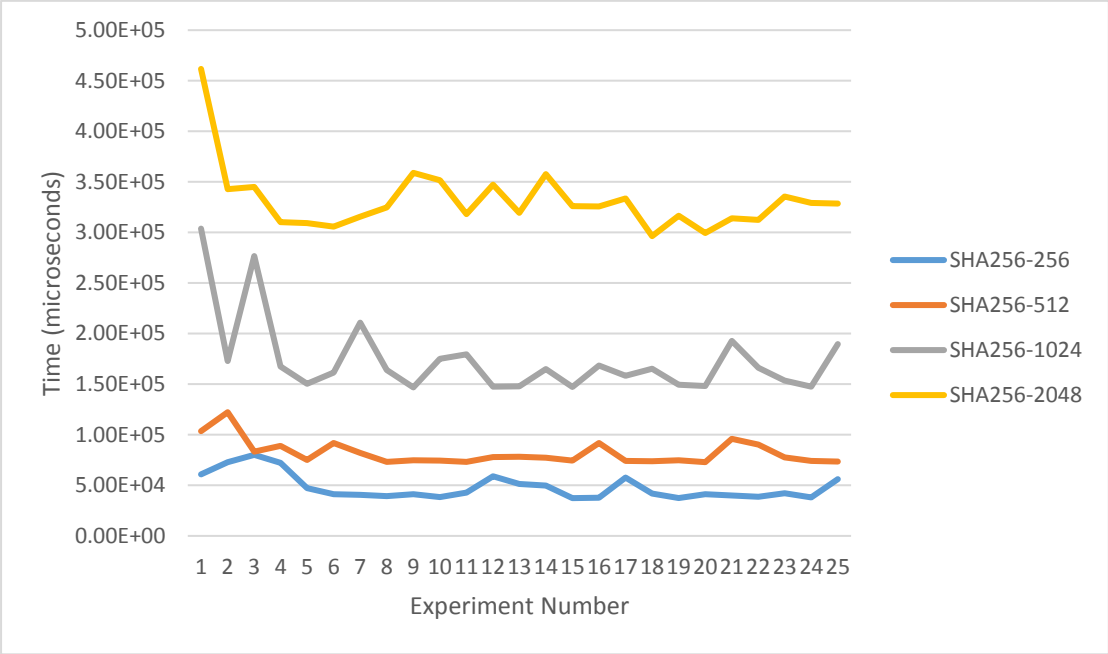


Figure 3.5. Execution time for ICMetric Strong Key Variants using SHA-256

Table 3.2 shows the average execution time for the generation of each ICMetric strong key variant based on the proposed ICMetric strong key generation scheme versus the experiment number being carried out. It is eveident that the average time required for the generation of strong ICMetric key increases with increase in the key size.

Table 3.2. Average Execution Time for ICMetric Strong Key Variants using SHA-256

ICMetric Key Variant	SHA256-256	SHA256-512	SHA256-1024	SHA256-2048
Average Time (microseconds)	48186.16	81988.52	174228.04	331397.68

This section evaluates the memory performance of strong ICMetric key generation scheme using SHA-256. Once again, a 128-bit device ICMetric is used as input for testing all the variants of the prototype. The memory performance of the strong ICMetric key generation scheme has been evaluated using Valgrind. The generated graphs depict memory profile of the variants by presenting the program lifecycle and the memory (bytes) consumed. It will be evident from all the RAM consumption graphs that cyassl instructions run directly from the chip, and hence are based on low level system calls. This in affect shows that only the stack memory is used during execution, thereby creating only a stack profile in the RAM consumption graphs for the designed scheme. Figure 3.6 shows a memory profile graph for the generation of a 256-bit ICMetric Strong Key using SHA-256.

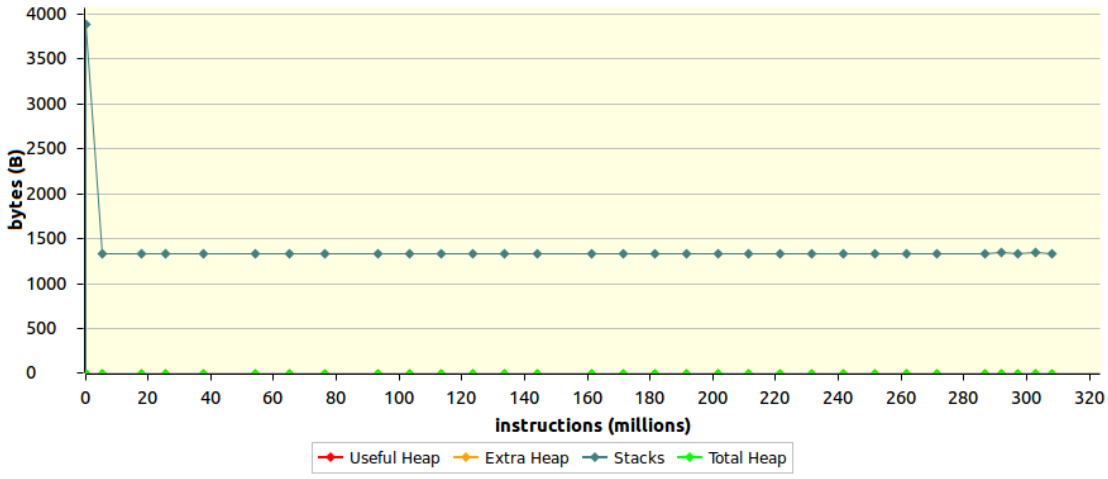


Figure 3.6. RAM Consumption for 256-bit ICMetric Strong Key using SHA-256

It is evident from the graph that the maximum memory consumed for the generation using 128-bit ICMetric is around 4KB. Figure 3.7 presents a memory profile graph for the generation of a 512-bit strong ICMetric key using SHA-256. It is clear from the graph that the maximum stack memory consumed for the generation of 512-bit key is around 4 KB. The stack memory, shown in green depicts low level systems

call being made by the proposed scheme implemented in cyassl. The low level system calls are particularly important from the perspective of embedded system environments.

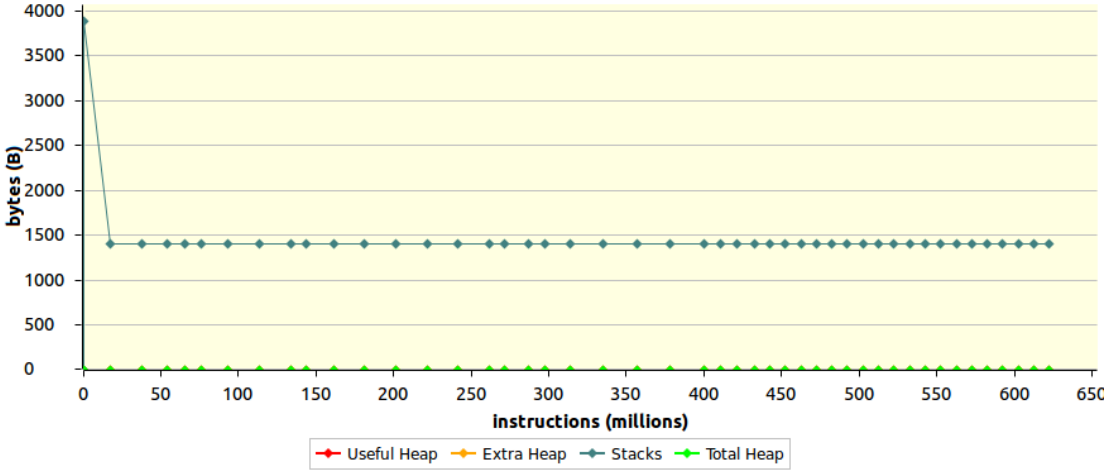


Figure 3.7. RAM Consumption for 512-bit ICMetric Strong Key using SHA-256

Figure 3.8 presents a memory profile graph for the generation of a 1024 bit strong ICMetric key using SHA-256. It is evident from the graph that the maximum memory consumed for the generation of 1024 bit strong ICMetric key is also around 4 KB.

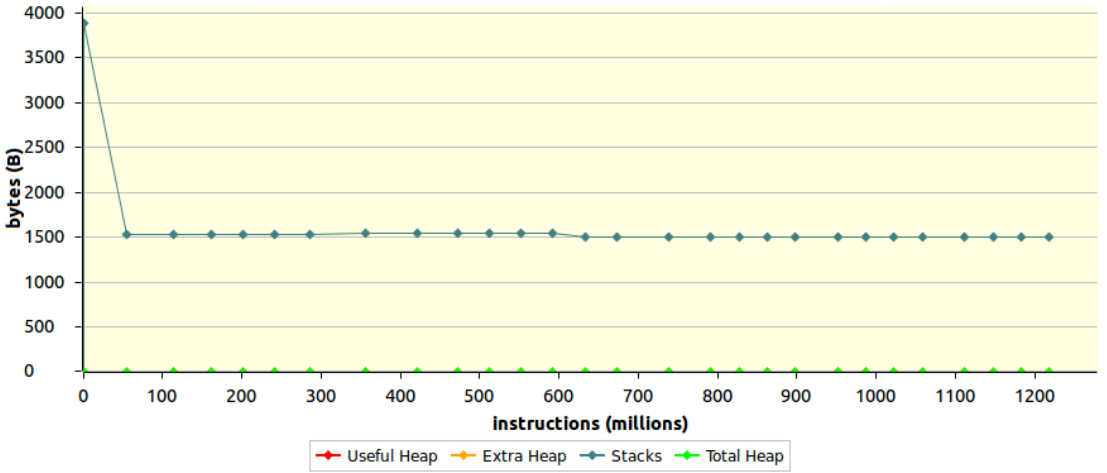


Figure 3.8. RAM Consumption for 1024-bit ICMetric Strong Key using SHA-256

Figure 3.9 shows the memory profile graph for the generation of a 2048 bit strong ICMetric key using SHA-256. It is again visible from the graphical results that

the maximum memory consumed for the generation of 2048-bit key is around 4 KB, which is very suitable for resource constrained devices.

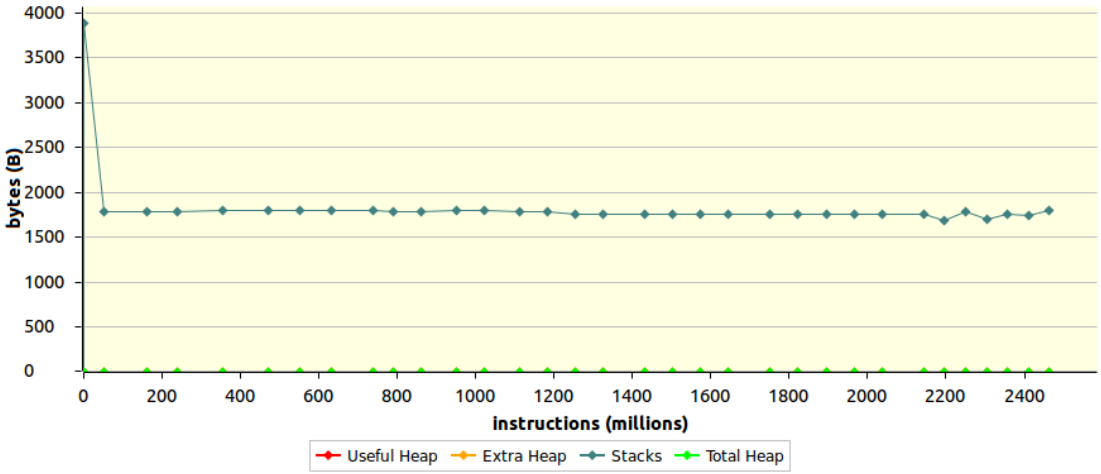


Figure 3.9. RAM Consumption for 2048-bit ICMetric Strong Key using SHA-256

Moreover, it is evident from all the SHA-256 based graphical results above that the RAM consumption increases only slightly with the increase in the required strong ICMetric key. Hence the proposed strong ICMetric key generation scheme is able to provide high levels of security at the cost of very small increase in memory consumption. Moreover, there is slight jitter observed in the graphs, which is not directly associated with our scheme and is due to other CPU related operations being performed.

3.4.2 ICMetric Strong Key Generation Scheme using SHA-512

This section evaluates the performance of the ICMetric strong key generation scheme using SHA-512. The scheme aims to generate strong ICMetric keys for carrying out cryptographic operations by supporting four strong key variants; namely 256 bit, 512 bit, 1024 bit and 2048 bit ICMetric strong key. Therefore, this section analyses these variants in terms of time and memory.

Firstly, the evaluation of the proposed ICMetric strong key generation scheme using SHA-512 take place based on the time requirements. A 128-bit device ICMetric and two 128-bit salt values as input for testing all the variants of the SHA-512 based prototype.

To prove the scalability of the strong key generation scheme, the time taken for the generation of keys of varying sizes has been considered. Figure 3.10 shows the experiment number versus time taken graph for computing various key variants namely 256, 512, 1024 and 2048 bits. It is evident from the graph that increase in the key size brings a change to the time performance with all the SHA-512 variants.

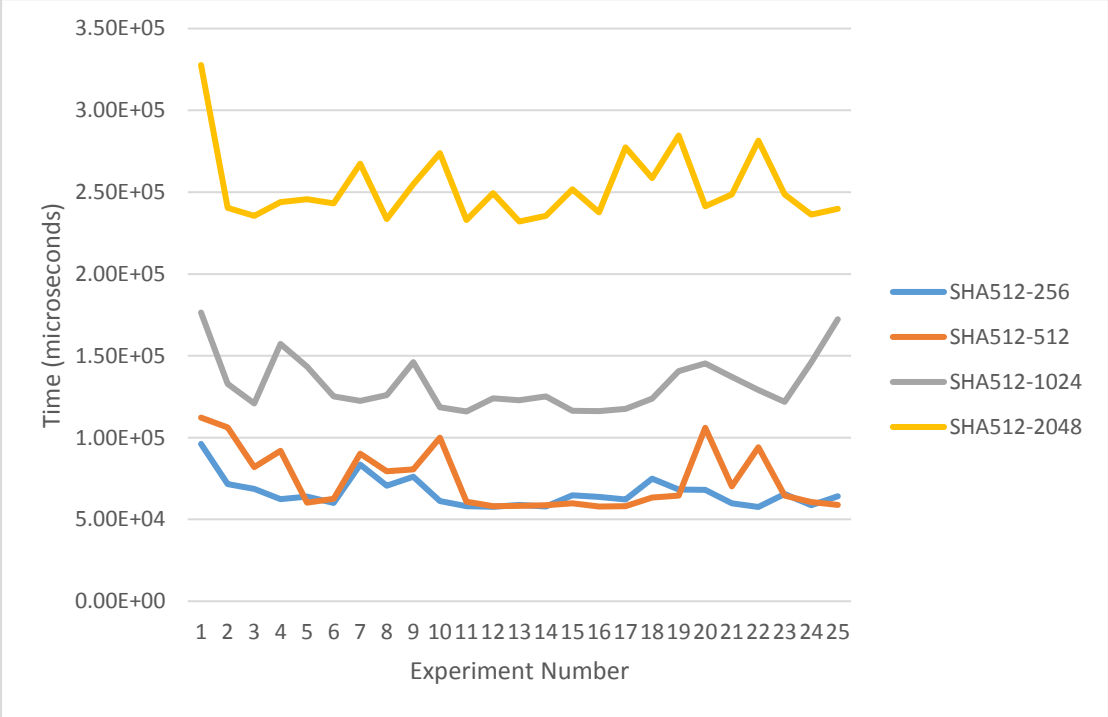


Figure 3.10. Execution Time for ICMetric Strong Key Variants using SHA-512

Table 3.3 shows the average execution time for the generation of each of the ICMetric strong key variants using SHA-512. It is clear from the graph that the average time required for the generation of strong ICMetric key increases with an increase in the required size of the ICMetric key.

Table 3.3. Average Exec Time for ICMetric Strong Key Variants using SHA-512

ICMetric Key Variant	SHA512-256	SHA512-512	SHA512-1024	SHA512-2048
Average Time (microseconds)	66186.04	74374.92	132929.16	252909.68

This section evaluates the memory performance of the strong ICMetric key generation scheme using SHA-512. Once again, a 128 bit ICMetric is used as input for testing all the variants of the proposed scheme. The memory performance of the proposed scheme is evaluated using Valgrind. The generated graphs depict memory profile of the variants by presenting the program lifecycle and the memory (bytes) consumed.

Figure 3.11 shows a memory profile graph for the generation of a 256 bit ICMetric strong key using SHA-512. It is evident from the graph that the maximum memory consumed for the generation of a 256 bit ICMetric key is around 4KB.

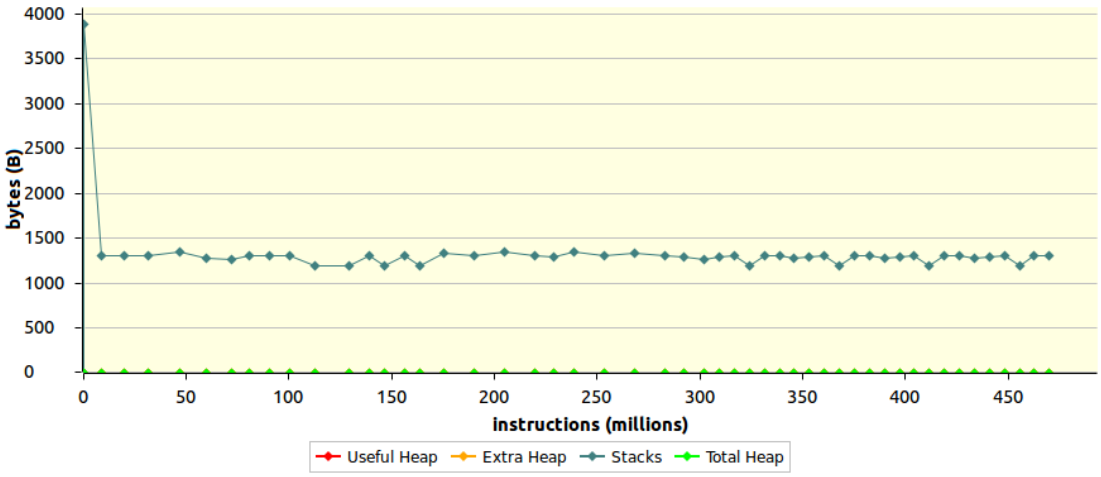


Figure 3.11. RAM Consumption for 256-bit ICMetric Strong Key using SHA-512

Similarly, Figure 3.12 presents a memory profile graph for the generation of a 512-bit strong ICMetric key using SHA-512, and it is clear from the graph that the maximum memory consumed is around 4KB.

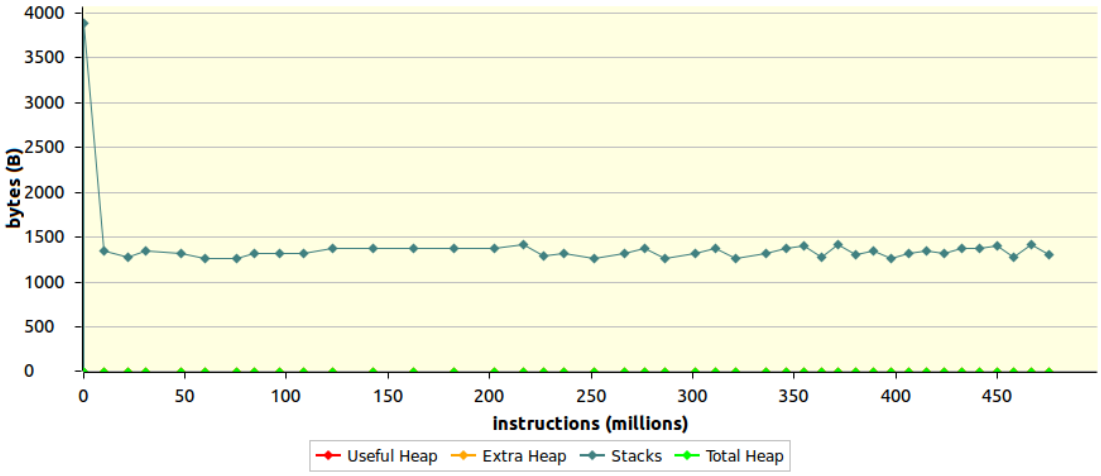


Figure 3.12. RAM Consumption for 512-bit ICMetric Strong Key using SHA-512

Figure 3.13 presents a memory profile graph for the generation of a 1024 bit strong ICMetric key using SHA-512 and shows the maximum memory consumed is again around 4 KB.

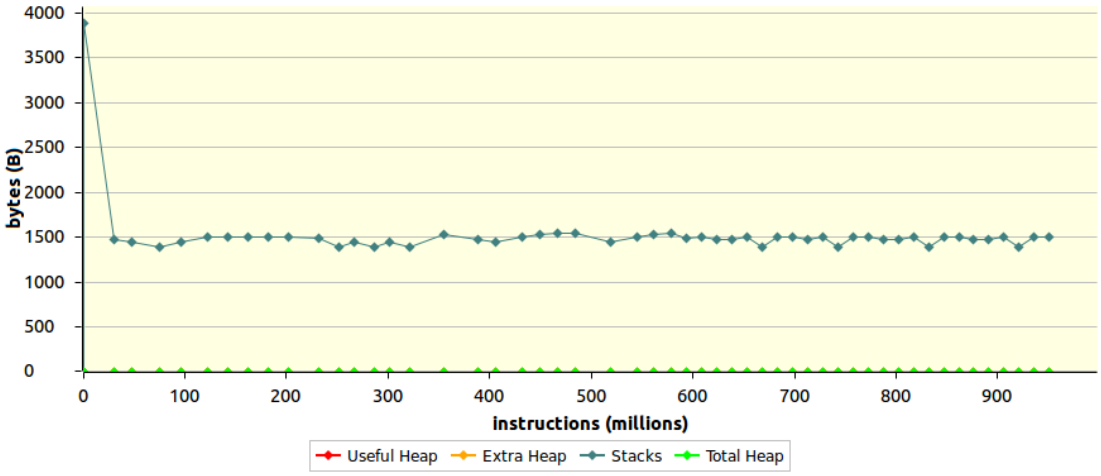


Figure 3.13. RAM Consumption for 1024-bit ICMetric Strong Key using SHA-512

Figure 3.14 shows the memory profile graph for the generation of a 2048 bit strong ICMetric key using SHA-512. It is visible from the graph that the maximum memory consumed for the generation of the 2048 bit strong ICMetric key is 4 KB.

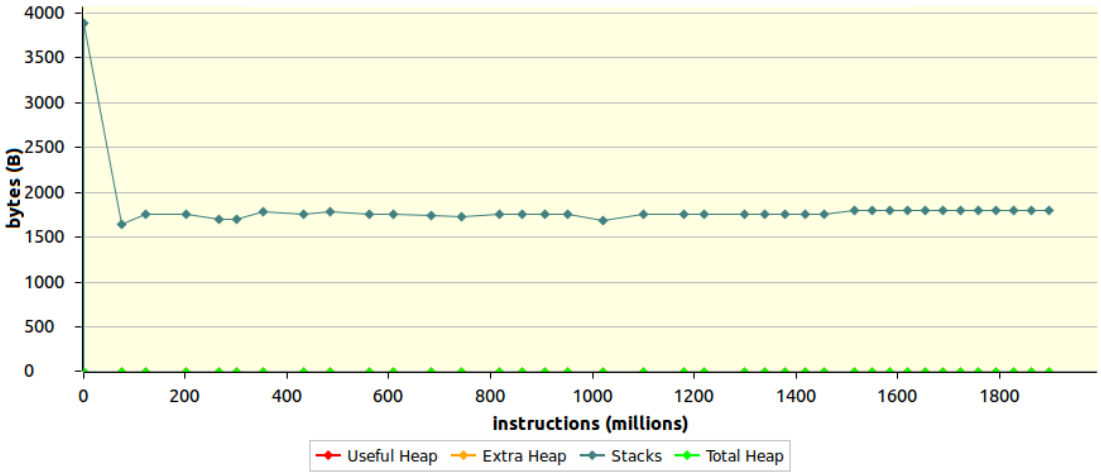


Figure 3.14. RAM Consumption for 2048-bit ICMetric Strong Key using SHA-512

It is evident from SHA-512 based results of the proposed scheme that the memory consumption over the lifetime of the strong ICMetric key generation increased only slightly with an increase in the required length of the strong ICMetric key. This proves that the proposed strong ICMetric key generation scheme can provide high levels of security at the cost of very small increase in memory consumption.

3.4.3 Performance Comparison

This section does a performance comparison between the proposed strong ICMetric key generation scheme and the standard PBKDF2 based on the execution time and RAM consumption. The performance comparison is carried out based on all the key variants of the strong ICMetric key generation scheme. This includes four different sized ICMetric strong keys namely 256, 512, 1024 and 2048 bits. Each variant has been further evaluated based on either SHA-256 or SHA-512.

The performance comparison between the proposed strong ICMetric key generation scheme and PBKFD2 is firstly carried out based on the time taken for the generation of a strong ICMetric key over several experiments.

Figure 3.15 performs a time performance comparison between the two variants of the proposed ICMetric strong key generation scheme and PBKDF2, for generating a 256-bit strong ICMetric key. These two variants namely SHA256-256 and SHA512-256 are compared with PBKDF2 to generate a 256-bit strong ICMetric key. 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.

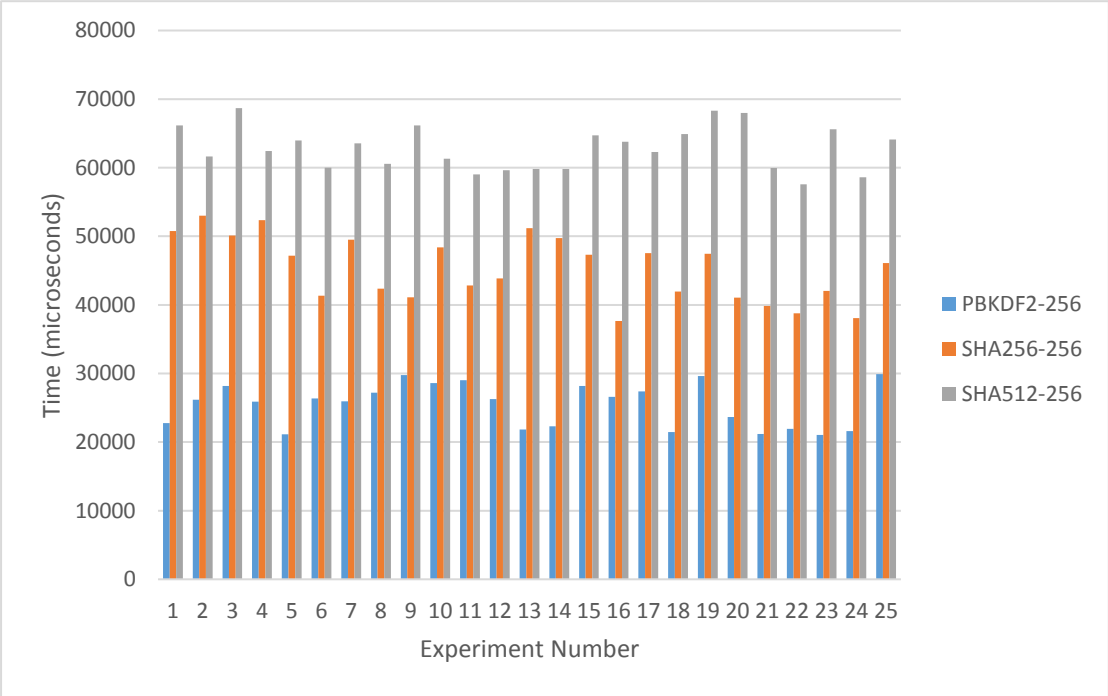


Figure 3.15. Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 256-bit Strong Keys

Table 3.4 shows the average execution time comparison between the two variants of the proposed ICMetric strong key generation scheme and PBKDF2 for

generating a 256-bit strong key. It is evident from the results in the table that the average time consumption for PBKDF2 is less than that of the ICMetric variants, but the security advantages of the ICMetric schemes far outweigh the security of PBKDF.

Table 3.4. Average Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 256-bit Strong Keys

Key Generation Scheme	PBKDF2-256	SHA256-256	SHA512-256
Average Time (microseconds)	25364.08	45266.16	62826.04

Figure 3.16 presents a time performance comparison between the two variants of the proposed ICMetric strong key generation scheme versus PBKDF2 for generating a 512-bit strong key.

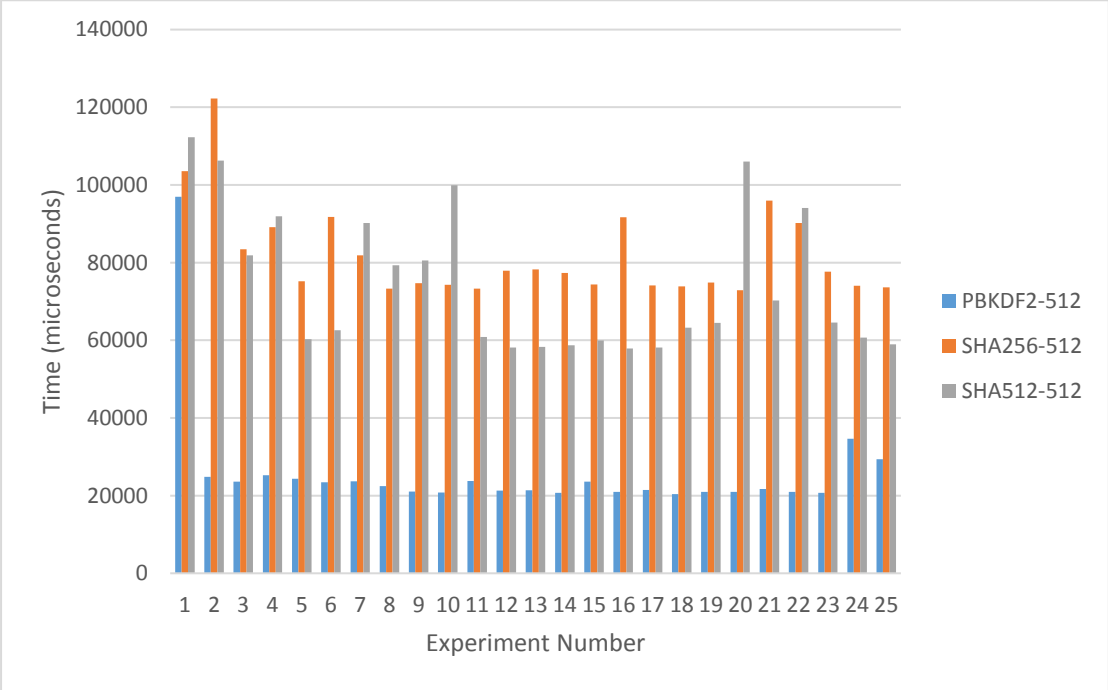


Figure 3.16. Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 512-bit Strong Keys

These two variants namely SHA256-512 and SHA512-512 are compared with PBKDF2 in the above graph to generate a 256-bit strong ICMetric key. The graph shows the number of the experiment versus the time taken in microseconds by each experiment to generate a 512-bit strong key.

Table 3.5 shows the average execution time performance comparison between the two variants of the proposed ICMetric strong key generation scheme and the PBKDF2 for generating a 512-bit strong key. It is evident from the graph in Figure 3.16 that the PBKDF2 takes 0.025 seconds to generate the strong key, whereas the strong ICMetric key generation scheme based on SHA-256 takes 0.082 seconds and the same based on SHA-512 takes 0.078 seconds.

Table 3.5. Average Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 512-bit Strong Keys

Key Generation Scheme	PBKDF2-512	SHA256-512	SHA512-512
Average Time (microseconds)	25966.4	81988.52	74374.92

Figure 3.17 presents a time performance comparison between the two variants of the proposed ICMetric strong key generation scheme versus PBKDF2 for generating a 1024 bit strong ICMetric key. These two variants namely SHA256-1024 and SHA512-1024 are compared with PBKDF2 to generate a 1024 bit strong ICMetric key. The graph in Figure 3.17 shows the number of the experiment versus the time taken in microseconds by the experiment for the generation of strong ICMetric keys. There is

an increase in the execution time for the proposed scheme, however the security advantages of the strong key scheme far outweigh the traditional PBKDF2.

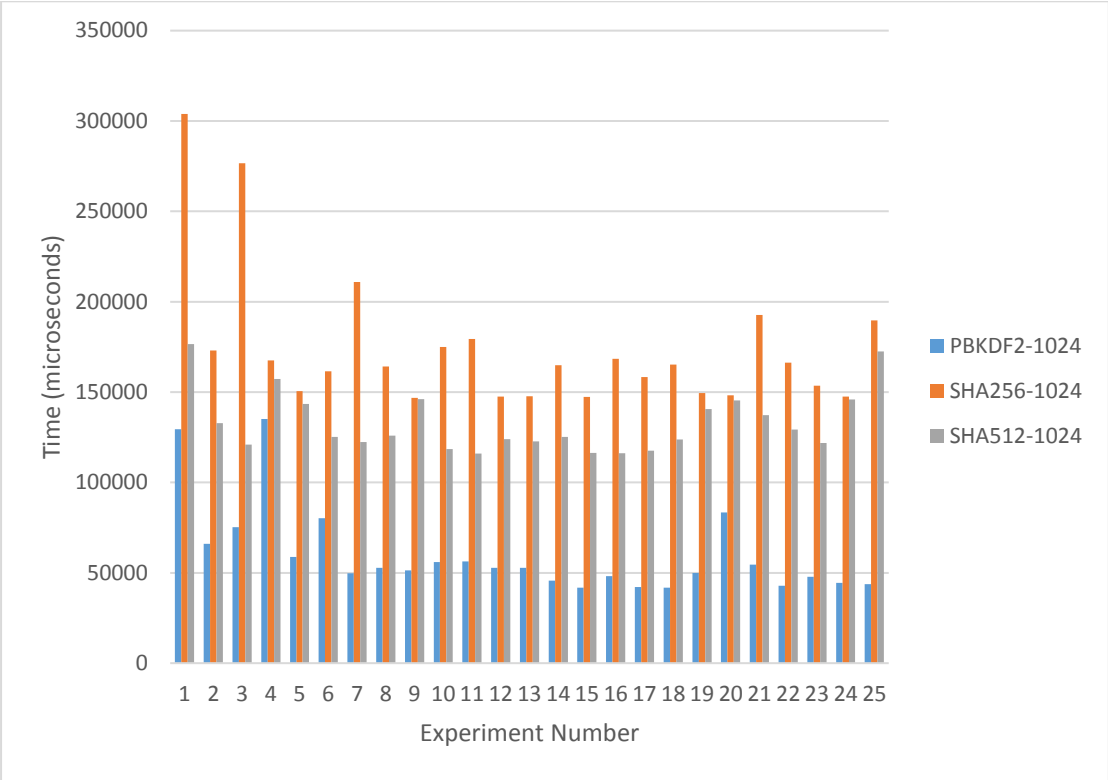


Figure 3.17. Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 1024-bit Strong Keys

Table 3.6 performs an average execution time comparison between the two variants of the proposed ICMetric strong key generation scheme and PBKDF2 for generating a 1024-bit strong key.

Table 3.6. Average Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 1024-bit Strong Keys

Key Generation Scheme	PBKDF2-1024	SHA256-1024	SHA512-1024
Average Time (microseconds)	60109.08	174228.04	132929.2

These two variants namely SHA256-1024 and SHA512-1024 are compared with PBKDF2 to generate a 1024 bit strong ICMetric key and can be seen to have a higher time consumption but at the same time provide more security advantages.

Figure 3.18 performs a performance comparison between the variants of the proposed ICMetric strong key generation scheme for generating a 2048 bit strong key. These results help assess the resource consumption of the strong key generation scheme.

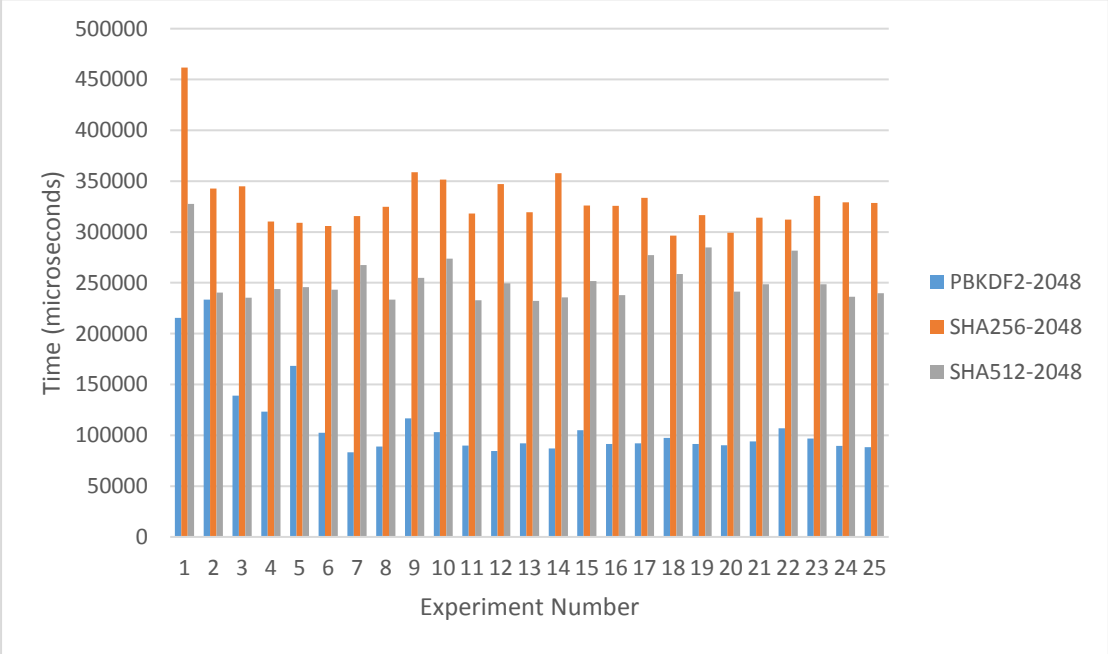


Figure 3.18. Execution time comparison of the ICMetric strong key scheme variants with PBKDF2 for the generation of 2048 bit strong keys

These two variants namely SHA256-2048 and SHA512-2048 in Figure 3.18 are compared with PBKDF2 to generate a 2048-bit strong ICMetric key. The above graph shows the experiment number versus the time taken in microseconds by the experiment for the generation of the 2048 bit strong key.

Further analysis of the obtained results for the generation of the 2048 bit ICMetric key are used in performing a performance comparison between the two variants of the proposed ICMetric strong key generation scheme for generating a 2048-bit strong key. These two variants namely SHA256-2048 and SHA512-2048 are compared with PBKDF2 to generate a 2048-bit strong ICMetric key. Table 3.7 shows that the average time taken by PBKDF2 to generate a 2048-bit key is 0.11 seconds, and by SHA-256 and SHA-512 based variants of the scheme is 0.33 seconds and 0.25 seconds respectively.

Table 3.7. Average Execution Time Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 2048 bit Strong Keys

Key Generation Scheme	PBKDF2-2048	SHA256-2048	SHA512-2048
Average Time (microseconds)	110795.4	331397.68	252909.68

In this section, the performance comparison between the proposed strong ICMetric key generation scheme and PBKFD2 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.

Figure 3.19 performs an average RAM consumption comparison between the two variants of the proposed ICMetric strong key generation scheme versus the PBKDF2 for generating a 256-bit strong key. These two variants namely SHA256-256 and SHA512-256 are compared with PBKDF2 to generate a 256-bit strong ICMetric key. The graph shows the name of the scheme versus the average RAM consumption

in bytes. It is evident from the graph that the SHA-512 bit variant of the proposed scheme has the maximum average RAM consumption. A comparison of strong ICMetric key generation based on SHA-256 and SHA-512 shows that, there is a 1.19% 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.

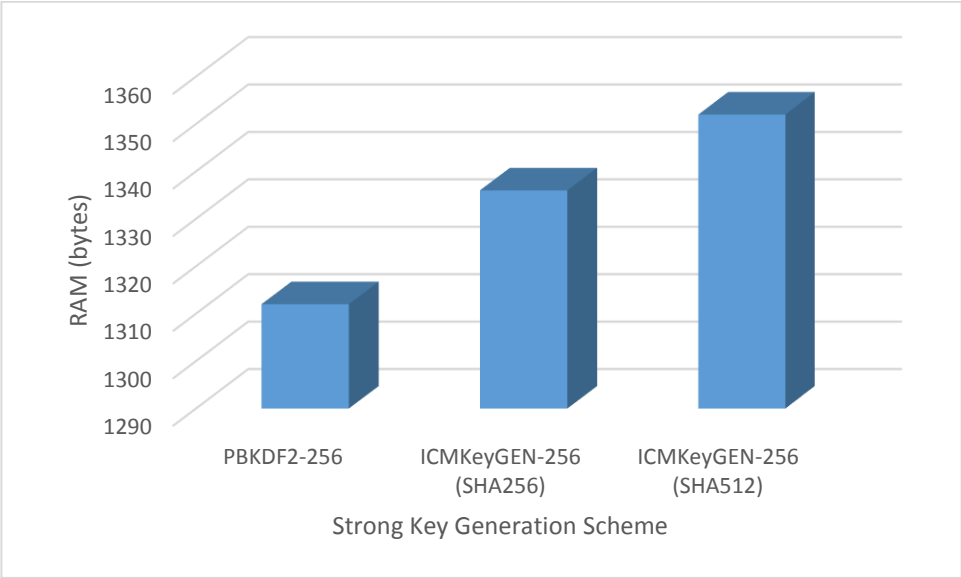


Figure 3.19. Average RAM Consumption Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 256-bit Strong Keys

Figure 3.20 shows an average RAM consumption comparison between the two variants of the proposed ICMetric strong key generation scheme for generating a 512-bit strong key, this is based on the number of the experiment versus the time taken in microseconds by that experiment. These two variants namely SHA256-512 and SHA512-512 are compared with PBKDF2 to generate a 512-bit strong ICMetric key.

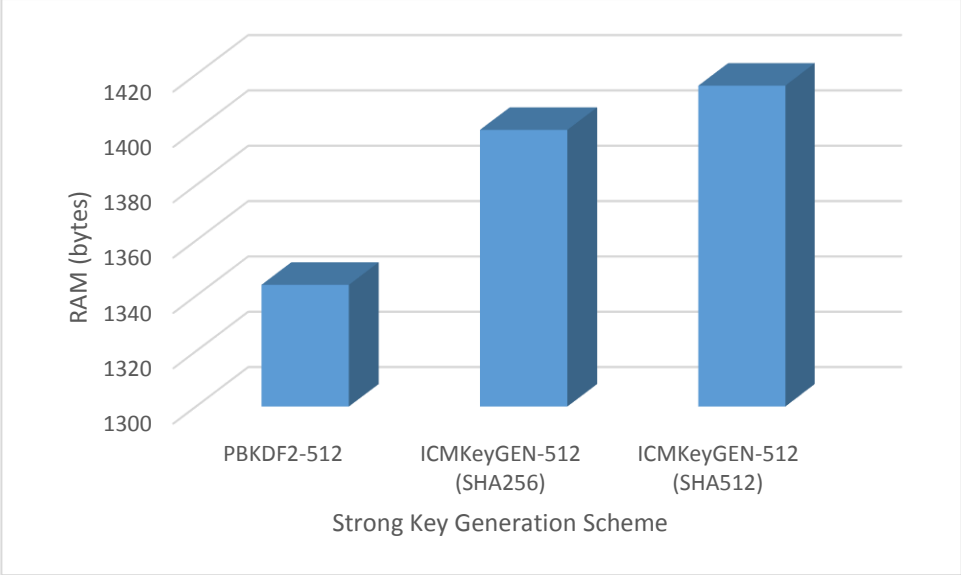


Figure 3.20. Average RAM Consumption Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 512-bit Strong Key

Figure 3.21 presents a performance comparison between the two variants of the proposed ICMetric strong key generation scheme for generating a 1024-bit strong key. These two variants namely SHA256-1024 and SHA512-1024 are compared with PBKDF2 to generate a 256-bit strong ICMetric key.

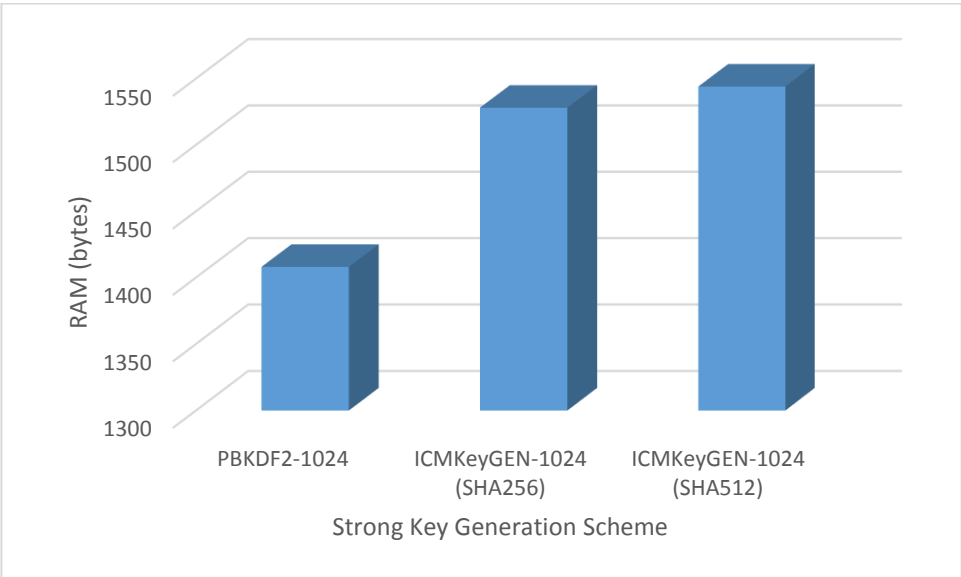


Figure 3.21. Average RAM Consumption Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 1024-bit Strong Key

Figure 3.22 gives a performance comparison between the two variants of the proposed ICMetric strong key generation scheme for generating a 2048-bit strong key. These two variants namely SHA256-2048 and SHA512-2048 are compared with PBKDF2 to generate a 2048-bit strong ICMetric key. The graph in Figure 3.22 shows the number of the experiment versus the time taken in microseconds by the experiment.

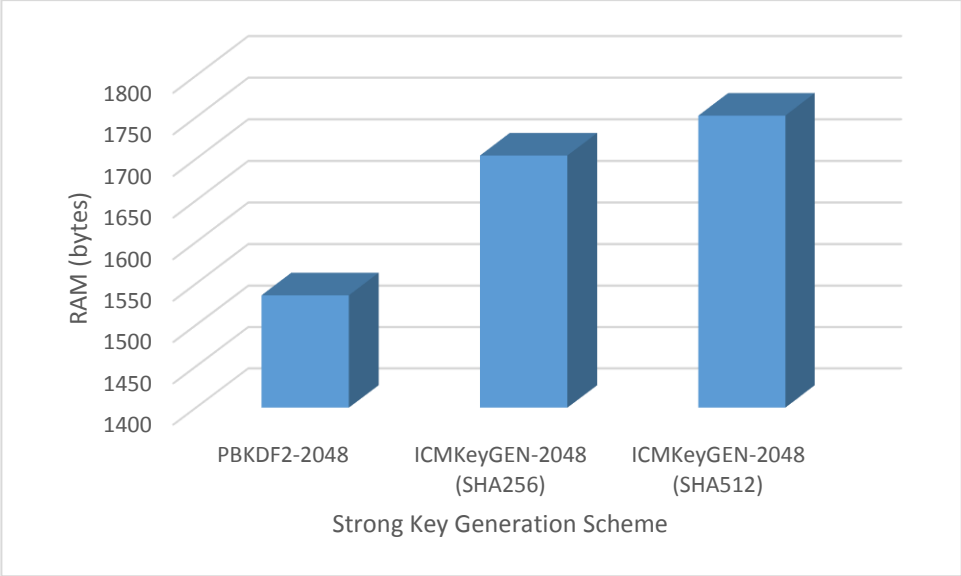


Figure 3.22. Average RAM Consumption Comparison of the ICMetric Strong Key Scheme Variants with PBKDF2 for the Generation of 2048-bit Strong Key

There is very slight difference in the RAM consumed by the four variants of the strong ICMetric key generation scheme 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. Further analysis shows that the percentage RAM increase between SHA-256 and SHA-512 for the generation of a strong ICMetric key is 1.03%.

3.5 STRENGTH OF THE ICMERTIC DERIVED KEY

To test the proposed key generation scheme that 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. They are analysed whether the keys generated through the designed scheme 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.

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 scheme 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'* [87], 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 [88].

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-bit bytes and the reported statistics reflect this property of the bit-stream. The detailed results for the three key lengths are given in Figure 3.23, that show the entropy of each generated strong ICMetric key in comparison with the rivalling key generation methods.

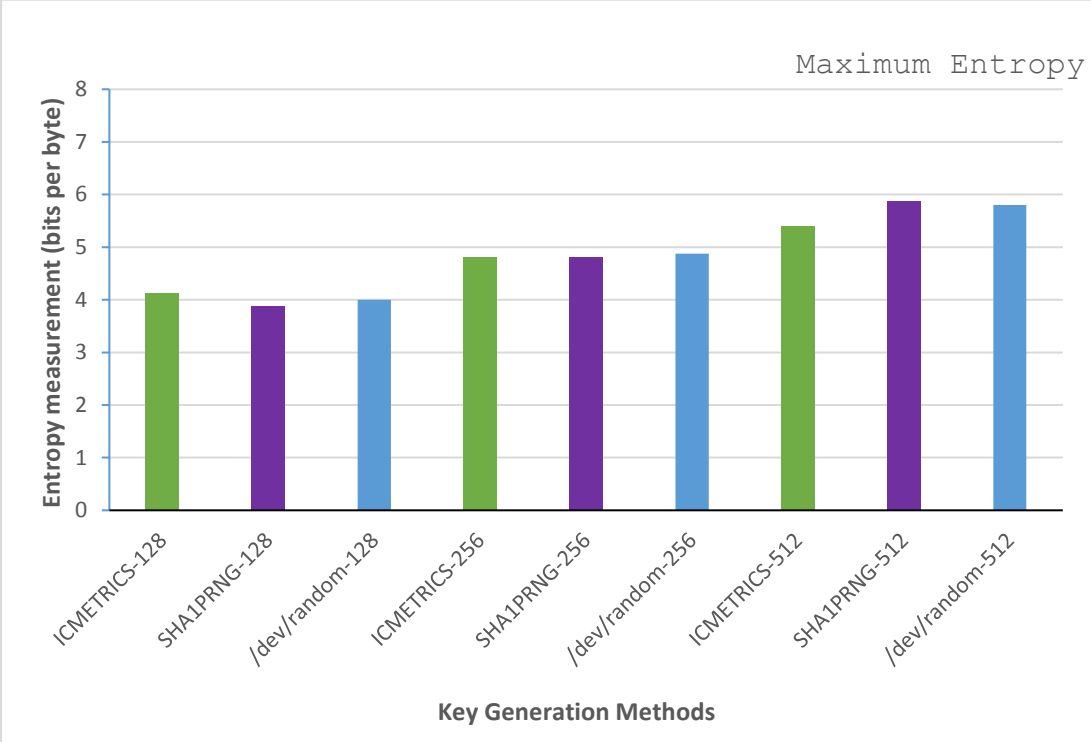


Figure 3.23. Entropy Measurements for Strong Key Variants

It is clear from the above Figure 3.23, 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 bit keys, around 4.8 bits per

byte for 256 bit keys and around 5.8 bits per byte for 512 bit keys. Therefore, the ICMetric based key generation scheme is faring as well as some other common and widely used cryptographically secure key generation schemes.

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 bit-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 3.24.

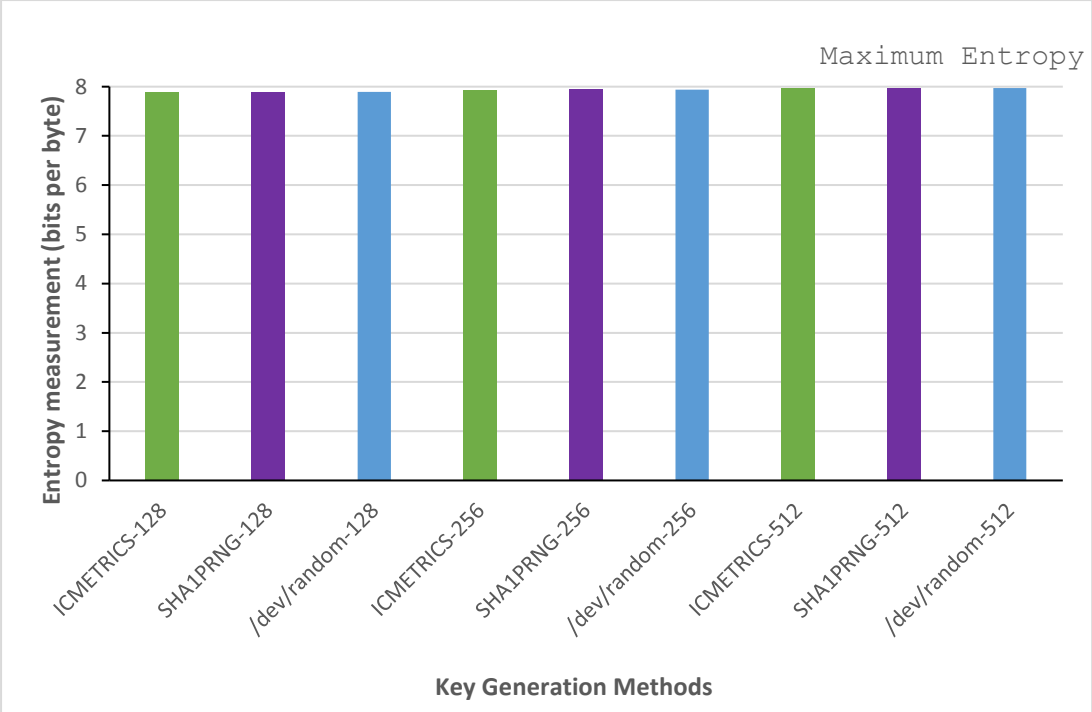


Figure 3.24. Entropy Measurements of Strong Key Variants based on Larger Inputs

As is clear from the above Figure 3.24, 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.

3.6 SECURITY ANALYSIS OF THE SCHEME

In the following section, we perform a security analysis of the designed ICMetric strong key generation scheme with respect to the security goals accomplished. In table 3.8, a summary overview of the goals accomplished/ unaccomplished by the designed ICMetric strong key generation scheme is presented. The table also compares the proposed ICMetric strong key generation scheme and PBKDF2 in terms of these security goals for the secure working of a strong key generation scheme.

Table 3.8. Comparison between Security Goals Accomplished by PBKDF2 and ICMetric Strong Key Generation Scheme

Security Goals							
	Brute-force Attacks	Dictionary Attacks	Rainbow table Attacks	Length	Entropy	Deters ICMetric capture	Adaptability
PBKDF2	✓	Partial	Partial	✓	✓	✗	✓
ICMetric Strong Key Generation	✓	✓	✓	✓	✓	✓	✓

The proposed strong ICMetric key generation scheme 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. The first strength of the scheme is that it safeguards the ICMetric keys from brute force attacks with two random salt values assigned to each entity. These two salt values are associated with

the hashing operation at two tiers of the proposed scheme, for the generation of a longer derived ICMetric key. The use of two salt values, and several key derivation function rounds make it more time consuming to crack the strong ICMetric key. The idea is to slow down the key generator function so brute force cracking of the key becomes prohibitively slow.

The point of computer security is never to make it (mathematically) impossible, just mathematically and physically impractical. The repetition of the hashing process multiple times, algorithmically slows down the hashing process which makes brute force attacks against the strong passwords much harder. This means that fewer passwords can be tested at once. As processors get faster, the number of iterations used to create the strong key are constantly increased, which means the key derivation function can scale with Moore's law [86]. The strong ICMetric key generation scheme produces a stretched high entropy key bringing an increase in both the entropy and length of the key thus safeguarding against pre-computed attacks. These features of length, entropy and randomness directly translate into more time being taken to break the key, and making brute force attack an impossible task.

The designed strong ICMetric key generation scheme is based on hash function computations. A major problem associated with schemes making use of hashing is the possibility of pre-computed attacks. These pre-computed attacks were of major concern for the proposed scheme and have been dealt with by the two-tier model of the designed scheme. The proposed scheme can defeat dictionary attacks; by passing the device ICMetric through two tiers of the scheme based on two salt values, thereby generating a strong ICMetric key that is not a simple variant of a word found in the dictionary. The two tier design of the scheme thwarts pre-computed dictionary attacks since an

adversary cannot create a password list by simply looking at the linear relationship between the output strong ICMetric key and the device ICMetric.

The proposed scheme safeguards against rainbow table attacks, owing to its two-tier architecture. An adversary makes use of two 128-bit salt 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 salts is to make the rainbow table a user specific element thereby increasing the complexity of the attack. A second salt value is used to provide a second layer of security with a random string in addition to the salt already being used. Doing so increase 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-bit random values at run time. The proposed scheme follows a two-tier approach; where in the first stage the output of thousands of hash operations on the device ICMetric and a salt value become input to another huge number of hashing operations with a second salt 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 scheme increases the time and complexity thereby making the strong ICMetric key difficult to break.

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 [15]. The strong ICMetric key generation scheme makes use of key derivation functions on the device ICMetric coupled with a salt, to generate strong ICMetric key variants of sufficient length. These strong ICMetric key variants namely; 256 bit ICMetric key, 512 bit ICMetric key, 1024 bit ICMetric key and 2048

bit ICMetric key can now serve as cryptographic keys in various cryptographic applications.

The proposed ICMetric strong key generation scheme generates keys that have sufficient entropy to be securely used in cryptographic applications. The inclusion of two random 128 bit salt values to the key derivation function, at two stages of the scheme 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.

The ICMetric technology is an innovative concept, designed to deter key theft. The proposed strong ICMetric key generation scheme also deters all forms of key capturing. The proposed scheme also 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 scheme is able to hinder an attacker's ability to get hold of the ICMetric at any point of the working of the scheme.

The proposed strong key generation scheme is very adaptable to the requirements of other cryptosystems and security applications. It has four strong ICMetric key variants i.e., a 256 bit ICMetric key, 512 bit ICMetric key, 1024 bit

ICMetric key and a 2048 bit strong ICMetric key. The length of each ICMetric strong key variants has been selected in accordance with the key requirements of well-known symmetric and asymmetric cryptosystems. The proposed strong ICMetric key generation scheme can form the foundation of any application domain wishing to make use of the ICMetric technology, and acts as a bridge between ICMetric and the cryptosystem making use of this technology. It converts an input ICMetric of any size to one of the strong ICMetric key variants.

3.7 SUMMARY

This chapter has presented a strong cryptographic key generation scheme based on the ICMetric technology. The two-tier approach followed by the scheme aims to strengthen the weak ICMetric key, thereby safeguarding it from pre-computed attacks and ICMetric capture. The proposed scheme acts to bridge the gap between the ICMetric technology and its use in security applications. It proposes four strong ICMetric key variants i.e., 256 bit strong ICMetric key, 512 bit strong ICMetric key, 1024 bit strong ICMetric key and a 2048 bit strong ICMetric key. The aim of the proposed strong ICMetric key variants is to make the scheme adaptable to the requirements of different cryptosystems.

This chapter then further goes on to do a comparative performance analysis between a state of the art key derivation function and the proposed strong key generation scheme. The results obtained from performance evaluation and security analysis lead to the conclusion that at the expense of additional time and memory, the proposed scheme makes the ICMetric technology very viable for deployment in various applications. Therefore, the next chapter showcases a protocol that can create a link

between ICMetric keys and symmetric key applications while keeping in mind the design principles of the ICMetric technology.

4

The ICMetric Symmetric Key Protocol

4.1 INTRODUCTION

Password Authenticated Key Exchange (PAKE) is the most important topic in cryptography. It aims to address a practical security problem of how to establish secure communication between two parties solely based on a shared password without requiring a Public Key Infrastructure (PKI); since maintaining a PKI is expensive. This chapter presents a symmetric key protocol for applications based on the ICMetric technology, referred to as the ICMetric symmetric key protocol. The designed protocol aims to bridge the gap between the ICMetric technology and its use in symmetric key applications. The ICMetric symmetric key protocol is based on a symmetric cryptosystem. This protocol aims to establish a symmetric key between two parties on a network, typically to protect further communication. A conventional shared secret authentication protocol is not suited for use with the ICMetric technology. Therefore,

the proposed design integrates zero knowledge password proofs (ZKPP), into the protocol design of symmetric key applications based on the ICMetric technology. ZKPP have proved to be particularly appropriate for ICMetric based applications due to the design principles of the ICMetric technology itself. The ICMetric of an entity cannot be communicated during cryptographic operations, therefore verifier based zero knowledge password proofs [89][90] are employed to authenticate participating entities and thereby establish a symmetric ICMetric key. The established symmetric key is used for symmetric cryptosystems based on the ICMetric technology for achieving confidentiality of data.

4.1.1 Symmetric Ciphers

Symmetric key cryptography is the oldest and best-known technique used for the encryption of data. The classical cryptographic protocols are based extensively on the utilization of this technique for developing secure communication protocols. They are characterised by the idea of using the same key to encrypt and decrypt the secret message. A secret key is required to be shared among the communicating parties prior to the secure communication and is kept hidden from the adversaries [98]. The Advanced Encryption Standard (AES) was chosen in 2001 as the winner of a 5-year contest to replace the then outdated and insecure DES. AES is a version of the Rijndael algorithm [99] designed by Joan Daemen and Vincent Rijmen. AES is also an iterated block cipher, with 10, 12, or 14 rounds for key sizes 128, 192, and 256 bits, respectively. AES provides high performance symmetric key encryption and decryption. Although multitudes of cryptographers have examined it over the last 10 years or so, no substantial attacks against the algorithm itself have been published.

4.1.2 Authentication Protocols

Password authentication protocols come in many flavours [91], but they all solve the same problem; one party must somehow prove to another party that it knows some secret usually set in advance.

A secure authentication protocol is expected not to leak any information about the password to eavesdroppers. In the simplest of all password authentication protocols, Carol (the user or client) sends Steve (the host or server) her username and her plaintext password, and Steve verifies the password, either by comparing it directly to his version of Carol's password or applying a one-way hash function first and checking against a database of stored hashes. Since Carol's password is immediately exposed to any eavesdropping attack, this method is unacceptable.

The authentication mechanisms that require a server to store a copy of the user's password are known as plaintext equivalent mechanisms [92]. The authentication mechanism that require a verifier to be stored at the server are called verifier based mechanisms [92][17]. Verifier based mechanisms have a significant advantage over the mechanisms that are plaintext equivalent. A system that uses plaintext-equivalent authentication becomes instantly compromised once the password database is compromised, since every user's password is in the database. A database of verifiers on the other hand can be protected just as easily and effectively as a database of plaintext equivalent passwords, except that failure of said protection is not as catastrophic if only verifiers are compromised [93].

Secure authentication protocols are expected not to leak information about the password to eavesdroppers, protocols classified as zero knowledge don't leak any

information about the password to the server. This subset of verifier based protocols is strong indeed.

4.1.3 Zero Knowledge Password Proof

A zero-knowledge password proof (ZKPP) is an interactive method for one party (the prover) to prove to another party (the verifier) that it knows a value without revealing anything other than the fact that it knows that password to the verifier. A ZKPP is secure against off-line dictionary attacks and prevents any party from verifying guesses for the password without interacting with a party that knows it; and in the optimal case provides exactly one guess in each interaction. A common use of a zero-knowledge password proof is in authentication systems where one party wants to prove its identity to a second party using a password but doesn't want the second party or anybody else to learn anything about the password. This property makes zero knowledge password proof particularly suitable for ICMetric application.

4.1.4 Cryptographic Password Authentication

The notion of PAKE was introduced by Bellovin and Merritt [94]. The first and maybe best known PAKE protocols include SPEKE by Jablon [95] and EKE by Bellovin and Merritt [96]. PAKE allows two parties, holding low-entropy keys, to negotiate a common authenticated session key. Despite the key exchange functionality, it authenticates the two parties. They aim to protect against offline dictionary attacks but require restrictions on the number of failed password trials as all password-based protocols to preserve security against online dictionary attacks. The standard model of PAKE does not require any PKI. Therefore, PAKE protocols solve the problem of potential password leakage, inherent to the approach based on secure channels.

A password authenticated key agreement method is an interactive method for two or more parties to establish cryptographic keys based on one or more party's knowledge of a password. An important property is that an eavesdropper or man in the middle cannot obtain enough information to be able to brute force a password without further interactions with the parties for each (few) guesses. This means that strong security can be obtained using weak passwords.

A challenge intrinsic to all PAKE protocols is the issue of server compromise. It has been mentioned from the first PAKE protocols by Bellovin and Merritt [94] describing the threats arising from stolen password databases. Servers store passwords in databases to retrieve them when necessary. To offer better protection against password database compromise servers store only the randomised password hash and the random salt that was used. Bellovin and Merritt [94] first described how password authenticated key-exchange can be performed while the server stores only a verifier of the actual password. Their sketched idea resembles the concept of Verifier-based PAKE (VPAKE). This concept is also known as augmented PAKE [97].

The reason for protecting passwords on a server is rather simple. Computing a verifier from a password using only a one-way function allows an attacker to precompute a list of verifiers such that inverting the one-way function can be performed rather efficiently considering human password choice. Adding a small amount of true randomness to the input of the one-way function alleviates these attacks as rainbow tables for all possible random values would have to be created, which leads to a time/memory trade-off up to a point where storing those tables becomes inefficient. Nonetheless, VPAKE is an interesting primitive that deserves more investigation

because it gives stronger security guarantees than PAKE and models a real-world need where servers store only password verifiers.

4.2 SYMMETRIC KEY PROTOCOL

The proposed security framework details an ICMetric centered symmetric key protocol that can be used with ICMetric based symmetric key applications. The proposed scheme is an idea to improve the security of the ICMetric based applications by providing authentication and confidentiality of the data in combination with the ICMetric symmetric keys.

4.2.1 Adversary Model

Embedded systems consist of dedicated devices that aim to sense and monitor data. The security of embedded system devices has become particularly important, since the risk of data theft has been understood/ recognized. With the prevalence in attacks on embedded systems [20]; professional stakeholders like device manufacturers, administration agencies and researchers have started paying extra emphasis on the implementation of security in embedded system applications. Attacks on an embedded system can target the client, server or network. Attack improvisations can result in an attack that targets different parts of the system at the same time. Some attacks on embedded systems can be launched physically and do not require the attacker to possess specialized equipment. For instance, attacks based on social engineering, shoulder surfing and device theft are common in all domains of computing. Although these attacks are common they can be deterred by creating user awareness. Similarly, owing to the lack of awareness, people often leave logged in terminals unattended which can lead to data theft and penetration of specialized systems.

A threat facing embedded applications is the possibility of data leakage while it is being transmitted over the channel. Eavesdropping is a major threat to user privacy [100], the adversary can easily read unencrypted messages being transmitted over the network. The captured messages can reveal important information to the attacker about the users, which can pose a serious threat to user privacy. User's data can be safeguarded from these attacks by employing stringent security measures in the system that provide privacy to the sensed user data.

4.2.2 Security Goals

Ensuring security and privacy are at the core of ICMetric based applications. There are several security and privacy challenges that ICMetric based applications generally try to address. The proposed symmetric key protocol based on ICMetric, aims to fulfil the following security and privacy challenges. To secure an ICMetric based network of entities, the following attributes are considered.

A fundamental goal of the scheme is to generate strong ICMetric symmetric keys. This goal ensures generation and use of keys with high entropy and adequate size, so that key guessing attacks are not possible. The strong key generations are based on two tier ICMetric key derivation function fed with a salt value, that takes time stamp as a seed value to provide true randomization; thereby generating a strong ICMetric symmetric key.

A crucial security goal specific to our proposed scheme and the health domain is secure admission control. The purpose of this security goal is to ensure that, the joining entities are who they claim they are. Fulfilling this security goal means that the entities that are part of the system are trusted and authentic. Secure admission control ensures that impersonating and fabricated entities do not gain access to the system.

An essential security goal of the scheme is to authenticate entities. Authentication ensures that only authenticated entities can take part and any impersonating entities are eliminated at source. The proposed scheme fulfills this security goal by preserving the secrecy of the ICMetric. The ICMetric centered authentication scheme uses a verifier based Zero Knowledge Proof (ZKP) that prevents the actual ICMetric from being transmitted onto the channel for authentication of the participant.

A vital security goal of the scheme is to maintain confidentiality of data. Preserving the secrecy of data is of utmost importance, so that the data is not leaked to adversaries. The proposed scheme uses an ICMetric symmetric key based encryption/decryption scheme to provide confidentiality of data.

A security goal of the ICMetric symmetric key protocol is that it doesn't leak any information that allows a passive/active attacker to perform offline exhaustive search of the password. It prevents a disclosed session from affecting the security of other established session keys.

4.3 THE ICMETRIC SYMMETRIC KEY SCHEME

The proposed ICMetric based symmetric key scheme acts to bridge the gap between the ICMetric technology and its use in symmetric key cryptosystems. The proposed details a scheme which makes the ICMetric technology compatible for use with symmetric key applications.

The proposed scheme is intended for networked environments and its design is based on the following assumptions:

- The communication links between the server and entities are all unprotected. Therefore, the data being transmitted over the communication channel should be protected.
- Each entity (sensor, device, etc.) needs to be registered with the server.

The sever is responsible for assigning all the network specific configurations to successfully join a network. All entities have already decided to trust the server, either without authentication or based on server authentication via SSL. The following section details the steps involved in the secure functioning of the symmetric key scheme based on ICMetric.

4.3.1 Admission Control

The requirement and importance of secure admission control is obvious as key management and secure communication schemes are effective only after the entities join the network in a secure admission process. The admission control scheme is used only once when the entity registers for the first time. The registration of an entity is a separate process that requires both manual and electronic data collections. The process is initiated when an entity requests to register by supplying necessary credentials. The entity registration consists of the following unique activities.

- When an entity wishes to register with the server, its necessary credentials are forwarded to a server where the registration is digitized.
- Once the entity is registered, the server is updated with a unique identification and a 128-bit random per-entity value (so called "salt").
- A copy of the identification and salt is also provided to the entity for storage.

Each entity that is part of the system generates an ICMetric (ICM) as outlined in chapter 2 by extracting feature values. The registered entity computes $'h'$ based on the ID assigned id, salt s and the ICMetric as below

$$h = T(ICM + s + id) \tag{4.1}$$

where T is the two-tier key derivation scheme outlined in chapter 3.

The computed value h is used by the registered entity in the computation of the verifier

$$verifier = g^h mod\ n \tag{4.2}$$

The entity computes its verifier, and discards the ICMetric and the computed h; to safeguard from pre-computed attacks. The entity sends its verifier and salt value to the server. The server keeps the entity's verifier and salt value secret and uses it to perform mutual authentication in future. This ultimately requires the client to trust the server to process and store the received verifier in a secure way.

4.3.2 Symmetric Key Generation

This section provides the architectural details of symmetric key cryptographic module in the design of the scheme. The symmetric cryptographic module provides security functionalities that facilitate secure transmission of data between the server and the entity, which might be a sensor or a device. The proposed scheme is an idea to improve the security of the ICMetric by generating a symmetric session key between the entity and the server, which is resilient against pre-computed attacks. The generated ICMetric symmetric key is then used for securely communicating the data between the authenticated parties.

The process of symmetric key generation and authentication is carried out between the entity and the server to establish symmetric key for secure data communication between them.

As mentioned above in section 4.3.1, the server stores certain entries for each registered entity (sensor, device, etc.) in the form of a tuple. The entity ID is the identifying attribute for an entity assigned by the server at the time of registration. The salt s is generated by the server as a random number 128-bit number at the time of registration. The server already has a verifier v received and stored for every entity, which it uses now during the process of symmetric key generation and authentication. The entity generated its verifier v at registration time and sent it to KGC as

$$v = g^x mod\, n \tag{4.3}$$

Where g is a generator of the multiplicative group and n is a safe prime as described in Table 4.1.

The design of the ICMetric symmetric key generation and authentication scheme is based on the Secure Remote Password Protocol [17][18]. SRP is particularly suitable for our application, since its properties and design works very well with the constraints of ICMetric technology of not transmitting the ICMetric. SRP doesn't require the exchange of ICMetric information between parties for authentication and key generation.

To formally start the ICMetric based symmetric key generation process each registered entity that wishes to communicate generates its ICMetric number based on the procedure outlined in chapter 2.

$$x = \text{ICMetric of the entity} \tag{4.4}$$

Table 4.1. Mathematical Notations

Symbol	Meaning
n	A large prime number. All computations are performed modulo n
g	A primitive root modulo n (often called a generator)
$salt_A$	A random string used as the entity's salt
V	The verifier of the entity
Id_A	ID of entity
x	ICMetric of the entity
a, b	Ephemeral private keys, generated randomly and not publicly revealed
$h( )$	Two-tier ICMetric key derivation function
u	Scrambling parameter-A and B concatenated and hashed
K_A, K_S	Session keys

The authentication process for sending data readings to the server is initiated by the entity, when it sends an initiation request containing its id to the server asking for the assigned salt value. On contacting the server, each entity receives the salt stored on the server under its entity ID. Now the entity generates number a

$$a = h\,(x_B || Salt_B || Id_B) \tag{4.5}$$

where the function $h( )$ is the two-tier ICMetric key derivation function explained in chapter 3.

The client side now calculates A and sends the result to the server.

$$A = g^a \bmod n \tag{4.6}$$

The server does a similar operation to calculate b

$$b = h\,(x_s || Salt_s || Id_s) \tag{4.7}$$

thereby calculating B and also adds the public verifier to it and finally sending B to the sensor.

$$B = (kv + g^b) \bmod n \tag{4.8}$$

Where $k = h(N||g)$

Both sides compute a random scrambling parameter $u = h\,(A \,||\, B)$ based on the exchanged $A$and B. Now both sides construct the shared session key. The entity constructs it as follows:

$$\begin{aligned} S_B &= (B - k \cdot v)^{a+ux} \bmod n \\ &= (kv + g^b - k \cdot v)^{a+ux} \bmod n \\ &= (g^b)^{a+ux} \\ K_A &= h\,(S_A) \end{aligned} \tag{4.9}$$

The server constructs it as follows

$$\begin{aligned} S_B &= (A \cdot v^u)^b \\ &= (g^{a+xu})^b \\ K_B &= h\,(S_B) \end{aligned} \tag{4.10}$$

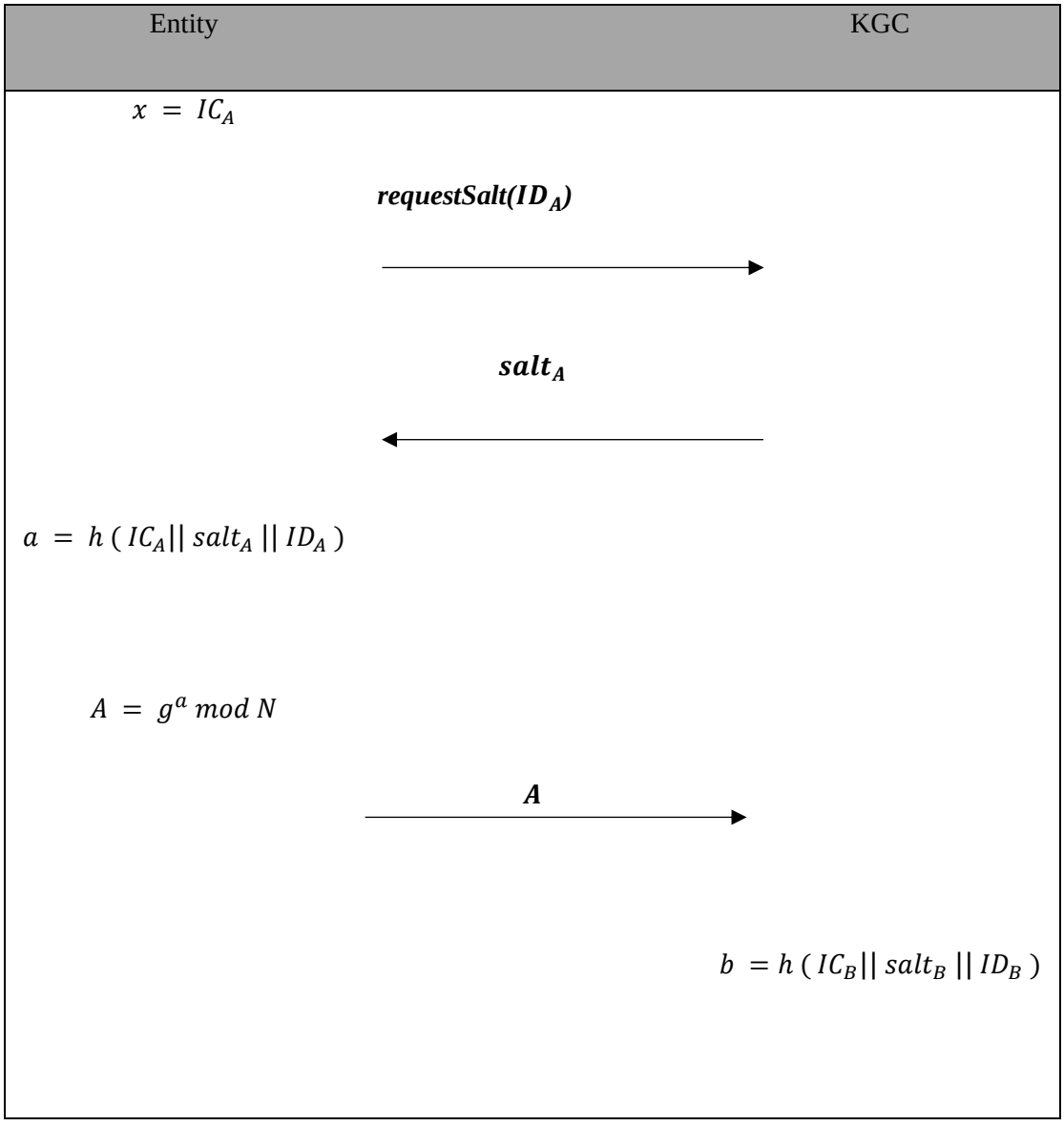
Both sides now possess the same secure session key K.

4.3.3 Symmetric Key Authentication

To complete the authentication, now the entity needs to prove to the server that its key is identical. To do so, the entity constructs the message M_1 and sends it to the server,

$$M_1 = h\,(\,N \,||\, g \,||\, ID_A \,||\, salt_A \,||\, A \,||\, B \,||\, K_A\,) \qquad (4.11)$$

The server will calculate M_2 using its own K_S and compare it against the message received from the sensor. If both keys don't match, the authentication fails resulting in refusal of communication request. If the sensor request is authenticated, the data can safely be relayed to the server. The summarized protocol is given in Figure 4.1. After an entity is authenticated and joins the network, its session key is kept in a secure cache and is valid for a set time.



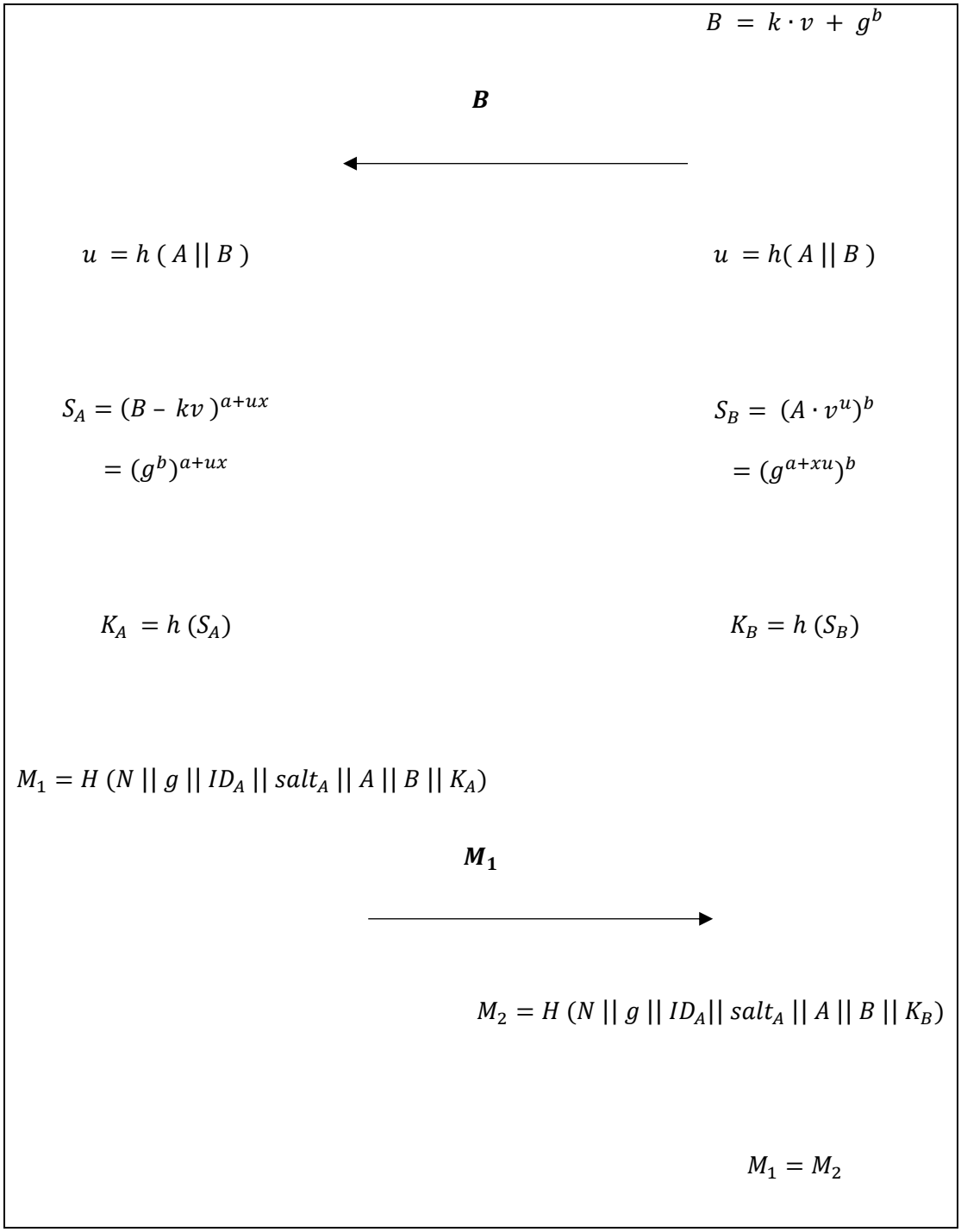


Figure 4.1. ICMetric based Symmetric Cryptographic Module

Mutual authentication can also be accomplished in the proposed scheme by server sending M_2 to client, and then the client comparing M_1 and M_2 to see if the server is authentic.

4.3.4 Symmetric Key Encryption/ Decryption

Once the session key is established, secure communications are carried out using any well-known symmetric key algorithms. In the proposed design AES has been used to securely send data between the entity and the server. So, in this step of the design all the individual ICMetric based components are integrated with the symmetric encryption/ decryption scheme to achieve data confidentiality between the parties.

4.4 IMPLEMENTATION AND EVALUATION

The working prototype of the protocol has been implemented using C on Linux. The scheme has been implemented on a first-generation Intel Corei3 3.2 GHz processor with 6 GB RAM. The authentication and key generation module of the design has been implemented using the OpenSSL cryptographic library. For the implementation of symmetric encryption/ decryption the CyaSSL [101] library has been employed. Although there is support for AES in OpenSSL, but CyaSSL library has been used for implementing the encryption/ decryption counterpart of the design. The CyaSSL is a lightweight SSL library that specifically targets resource constrained devices. CyaSSL is 20 times smaller than the standard OpenSSL. CyaSSL also takes advantage of Intel AES-NI support [102]. This helps boost the performance of AES, since the instructions run directly from the chip instead of running the algorithm from software, thereby making it run 5-10 times faster. The design and implementation choices are greatly influenced by the fact that embedded platforms possess limited resources. Detailed results in the upcoming section also show that the proposed scheme can achieve high levels of security without resource demand while deployment on a real sensor test bed.

The authentication and key generation module of the system implementation has five variants namely 160, 224, 256, 384, 512 bits; with a 128-bit device ICMetric as input. To offer flexibility the encryption/ decryption module of our scheme has been implemented to facilitate two AES variants namely AES-128 and AES-256.

The following section, evaluates the efficiency of the design by measuring the RAM consumption and execution time of the implemented scheme. For evaluating the execution time of the system, the programs runtime itself has been employed; whereas for evaluating the memory consumed by the prototype Valgrind is used, also detailed in section 3.9.

4.4.1 ICMetric Key Generation and Authentication Module

This section evaluates the performance of the ICMetric based key generation and authentication module that is part of the proposed symmetric key protocol.

Firstly, the time requirements for authentication and the key generation counterpart are evaluated, which is responsible for the authentication and the key generation of an ICMetric based symmetric key. A 128-bit ICMetric is taken as input for testing all the variants of the prototype. To prove the scalability of the key generation scheme, the time taken for the generation of keys of varying key sizes are studied. Figure 4.2 shows the key size versus time taken graph for computing various key variants namely 160, 224, 256, 384, 512 bits. It is evident from the graph that increase in the key size doesn't bring a drastic change to the time performance of the application. Therefore, the proposed scheme provides higher levels of security without substantial time performance overheads.

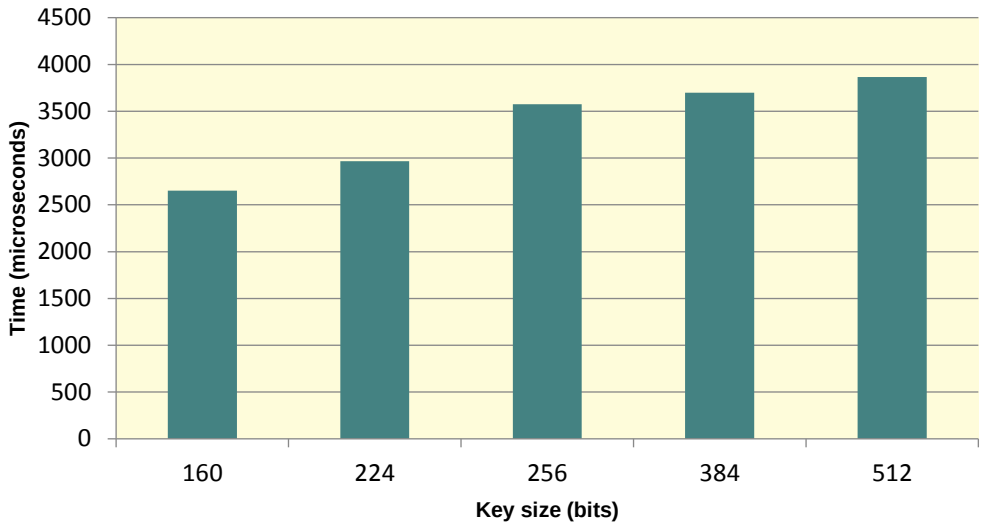


Figure 4.2. Execution Times for ICMetric based Symmetric Key Variants

This section evaluates the memory performance of authentication and the key generation counterpart, that is responsible for the authentication and the key generation of an ICMetric based symmetric key. Once again, a 128-bit ICMetric is taken as input for testing all the variants of the prototype. The memory performance of the ICMetric based authentication and key generation counterpart is evaluated using Valgrind. The generated graphs depict memory profile of the variants by presenting the program lifecycle and the memory (bytes) consumed.

Figure 4.3 shows a memory profile graph for the authentication and key generation of ICMetric. It is evident from the graph that the maximum total memory consumed for the generation and authentication using 160-bit key is around 11KB, which is very suitable for resource constrained devices. Therefore, the proposed scheme can provide higher levels of security at very optimum amounts of memory consumption.

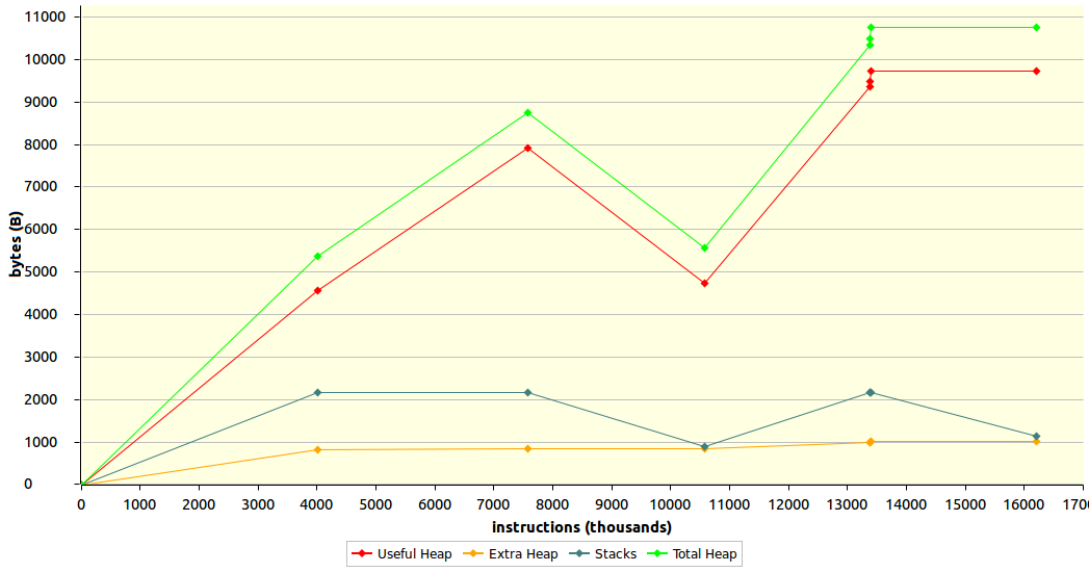


Figure 4.3. RAM Consumption for 160-bit ICMetric based Symmetric Key

Figure 4.4 presents a memory profile graph for the authentication and key generation of 224-bit ICMetric key.

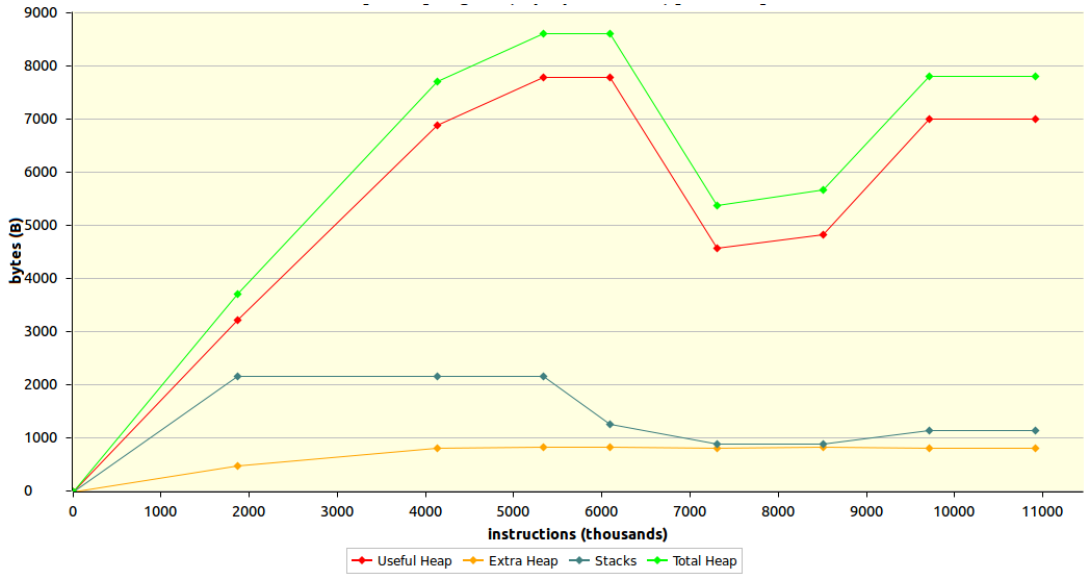


Figure 4.4. RAM Consumption for 224-bit ICMetric based Symmetric Key

It is evident from the graph above that the maximum total memory consumed for the generation and authentication using 224-bit key is around 8.5 KB, which is very

suitable for resource constrained devices. This proves that the proposed scheme can provide high levels of security at the cost of very small amount of memory.

Figure 4.5 presents a memory profile graph for authentication and key generation of 256-bit ICMetric key. It is evident from the graph that the maximum total memory consumed for the generation and authentication using 256-bit key is around 9 KB, which is very suitable for resource constrained devices. This proves that the proposed scheme can provide high levels of security at the cost of very small amount of memory.

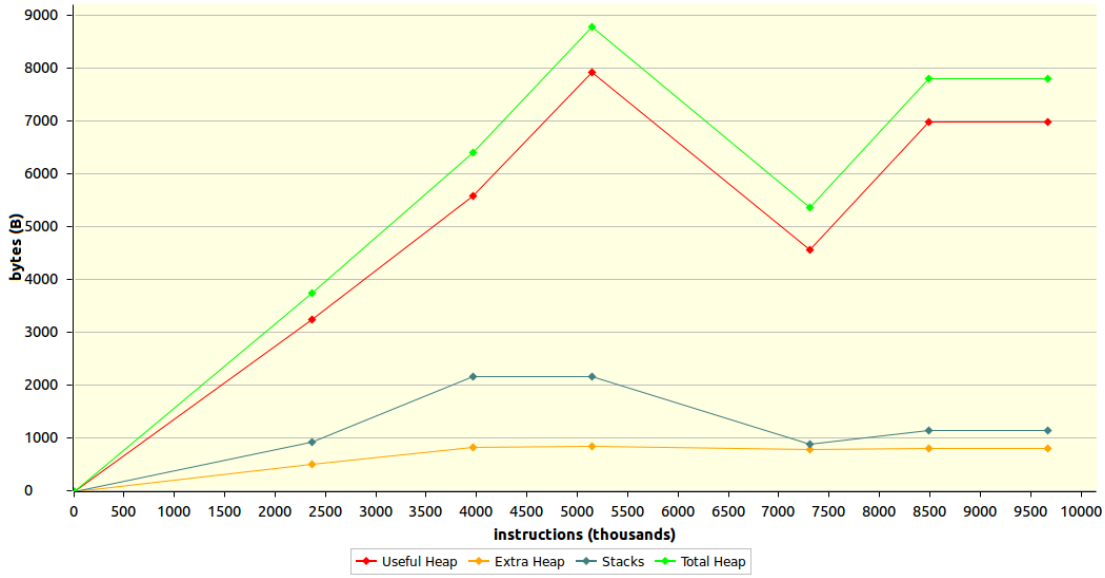


Figure 4.5. RAM Consumption for 256-bit ICMetric based Symmetric Key

Figure 4.6 shows the memory profile graph for the authentication and key generation of a 384 bit ICMetric key. It is evident from the graph that the maximum total memory consumed for the generation and authentication using 384-bit key is around 9 KB, which is very suitable for resource constrained devices. This proves that the proposed scheme provides high levels of security at the cost of very small amount of memory.

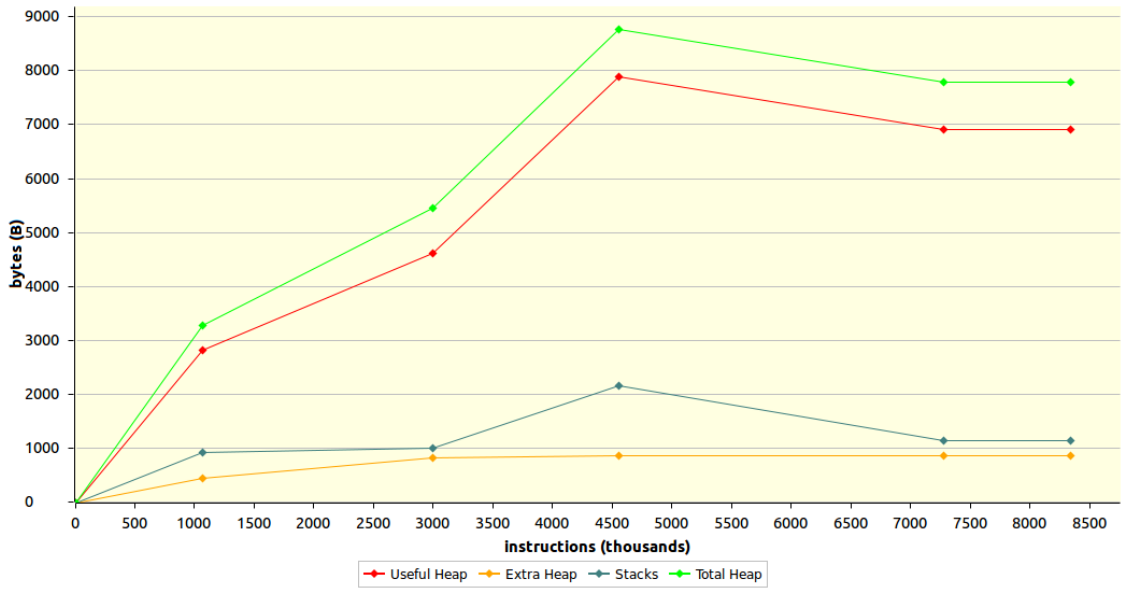


Figure 4.6. RAM Consumption for 384-bit ICMetric based Symmetric Key

Figure 4.7 shows the memory profile graph for the authentication and key generation using 512 bit ICMetric key.

Figure 4.7. RAM Consumption for 512-bit ICMetric based Symmetric Key

It is evident from the graph in Figure 4.7 that the maximum total memory consumed for the generation and authentication using 512-bit key is around 11 KB, which is very suitable for resource constrained devices. This proves that the proposed scheme provides high levels of security with a very small memory footprint.

4.4.2 ICMetric based AES Encryption/ Decryption Module

This section evaluates the performance of the ICMetric based encryption/ decryption counterpart of the implemented scheme. This section evaluates the time performance of encrypt/ decrypt module that is responsible for the encryption/decryption of data using the ICMetric symmetric key.

Figure 4.8 shows a bar graph of the 128 and 256 bit AES key variants versus the time taken.

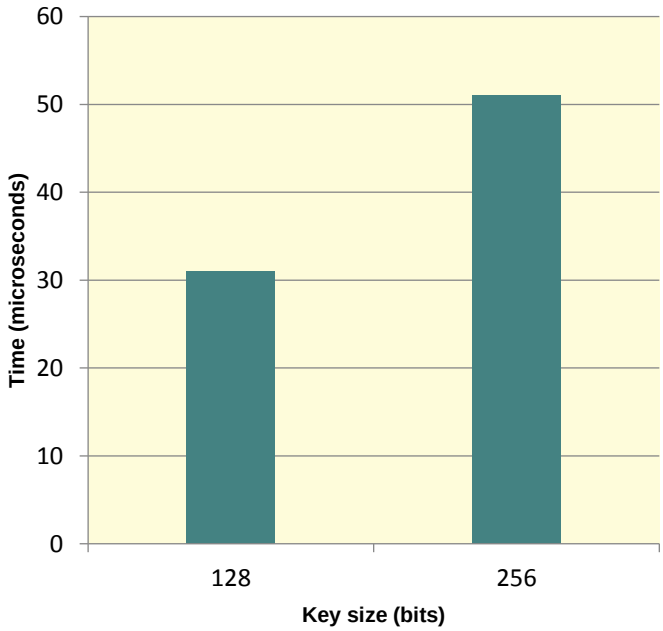


Figure 4.8. Execution Times for ICMetric based AES variants

It is evident from the graph that an increase in the key size has a very small effect on the time performance of the application. Therefore, the proposed scheme provides higher levels of security without substantial time performance overheads.

Figure 4.9 presents a memory profile graph used for the 128-bit AES encryption/ decryption using ICMetric symmetric key. It is evident from the graph that the maximum total memory consumed for the encryption/ decryption using 128-bit key is 3.8 KB, which is very suitable for resource constrained devices, especially since CyaSSL takes advantage of the Intel AES-NI. This proves that the proposed scheme provides high levels of security at the cost of very small amount of memory.

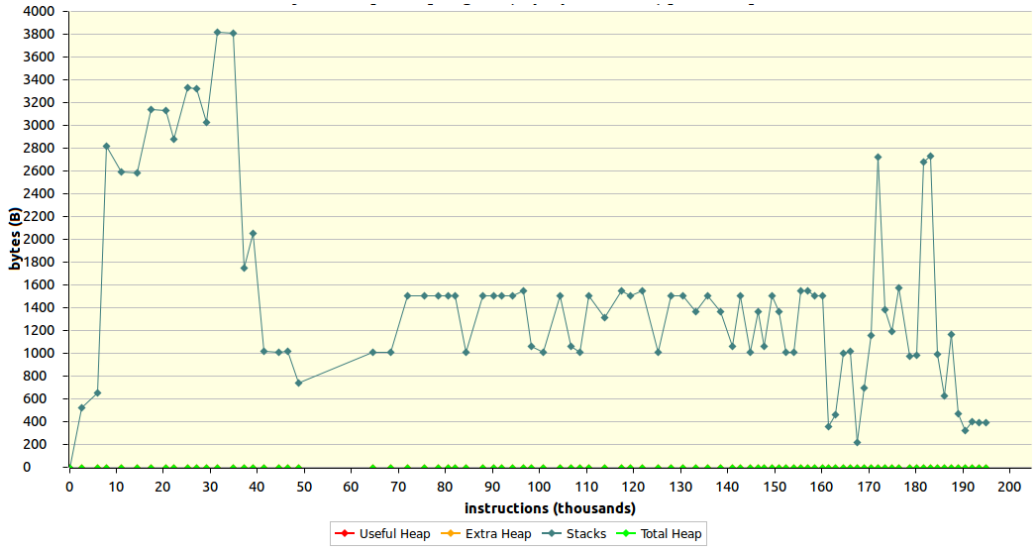


Figure 4.9. RAM Consumption for ICMetric based 128-bit AES Encryption

Figure 4.10 shows a memory profile graph for the 256 bit AES encryption/ decryption using ICMetric symmetric key. It is evident from the graph that the maximum total memory consumed for the encryption/ decryption using 256 bit key is 3.8 KB, which is again very suitable for resource constrained devices, and as mentioned above CyaSSL takes advantage of the Intel AES-NI. This proves that the proposed scheme provides high levels of security at the cost of very small amount of memory.

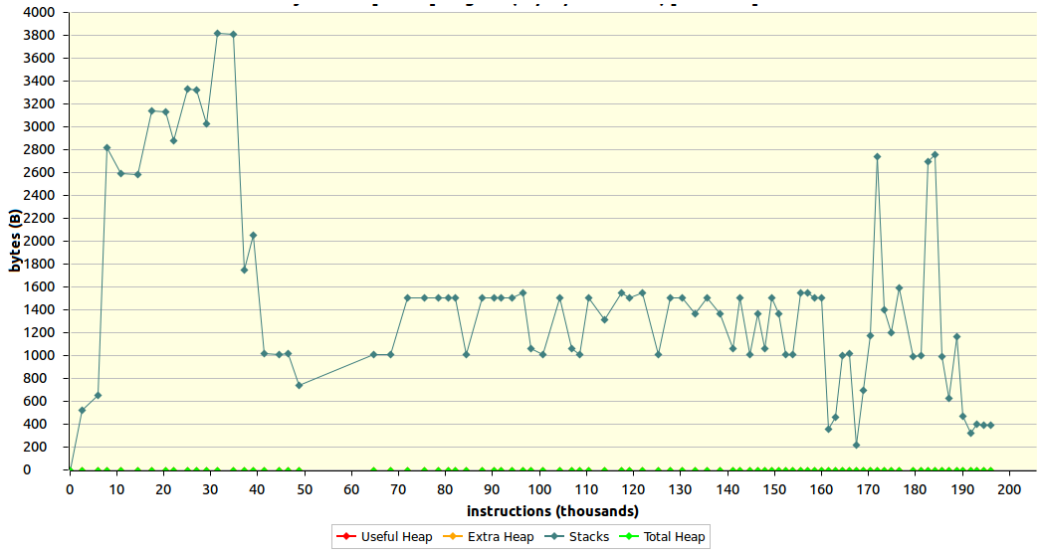


Figure 4.10. RAM Consumption for ICMetric based 256-bit AES Encryption

4.5 Security Analysis

This section performs a security analysis of the designed ICMetric symmetric key scheme; based on the security goals accomplished and the limitations of the scheme. Table 4.2 gives a summary of the goals accomplished/unaccomplished by individual system modules of the proposed symmetric key scheme.

Table 4.2. System Modules and Goals Accomplished

		System Goals				
		Authentication	**Confidentiality**	**Access Control**	**Non Repudiation**	**Precomputed Attacks**
Modules	**ICMetric**			✓	✓	
	Strong Key Generation					✓
	Symmetric Module	✓		✓		✓
	AES		✓			

The ICMetric technology is an innovative concept, designed to deter key theft. The designed scheme relies on generating the ICMetric from a range of features that are unique to a computation device. These features are extracted when required and an ICMetric is computed. This resulting ICMetric is used to generate cryptographic keys. When the keys are no longer required both the ICMetric and the cryptographic keys 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.

The ICMetric is generated using unique hardware features of a computation device. This implies that each ICMetric is associated with only a single entity. Since the used features of the entity cannot be forgotten, lost, stolen or fabricated, this prevents an entity from denying a particular action performed by their device. Here non repudiation is an inferred security goal, which is obtained when the ICMetric technology is used for the provision of enhanced security.

The originally generated ICMetric is weak; and is of insufficient length and entropy, to serve as a key for cryptographic operations. The symmetric key generation scheme in security framework uses key derivation functions on ICMetric number coupled with a salt, to generate a strong symmetric ICMetric key.

Our security framework effectively conveys a zero-knowledge password proof. This property is particularly important, since at no point during our scheme can the ICMetric be transmitted over the channel. The ICMetric serves as a fingerprint for an entity, so it cannot be transmitted over the channel during security operations, such as authentication and access control in particular. The zero-knowledge password proof ensures that the entity can prove to the KGC about the knowledge of the ICMetric number, without transmitting/ communicating the ICMetric to the verifying party.

The authentication and access control in the proposed scheme is based on the generated strong ICMetric key of each entity. A major advantage provided by the proposed symmetric key scheme is that of authenticating entities without the need for human intervention. This is particularly important from the perspective of applications where human intervention is not always possible for authentication, therefore the authentication functionality is automatically carried out based on the entities' ICMetric keys. To generate the symmetric key, the generating entity/ device must have the knowledge of the assigned salt value and must then generate its ICMetric. Knowing only one of them does not allow the generation of symmetric key. This safeguards the network from attackers, since only authenticated entities that have been registered/ assigned a salt by the server can form part of the symmetric key generation process.

Each device concatenates its ICMetric with a random per-user value (so called "salt") and stores the hash value of the result along with the salt. This makes certain kinds of brute-force attacks and dictionary attacks more difficult. The security features of the design also prevent the possibility of man in the middle attack. The scrambling parameter ensures that each entity only gets one verification attempt at impersonation, resulting in the ruling out the possibility of man-in the middle attack. The data sent by each entity is encrypted using AES based on the ICMetric symmetric key, so that health data is not leaked to unauthorized parties. Therefore, only authenticated parties that are in possession of a symmetric key can read the health data.

4.6 Summary

This chapter has presented a symmetric protocol for ICMetric key applications. The proposed ICMetric symmetric key protocol acts to bridge the gap between the ICMetric technology and its use in cryptosystems. The proposed symmetric key

protocol for ICMetric based applications is based on a zero-knowledge proof password and works very well with the design principles of the ICMetric technology. This chapter then goes on to do a performance evaluation of the proposed ICMetric symmetric key protocol. The results obtained from the performance evaluation and security analysis lead to the conclusion that at the expense of additional time and memory, the proposed scheme makes the ICMetric technology very viable for providing security in symmetric key applications. This chapter concludes by showcasing how the proposed ICMetric based symmetric protocol can be used in practice, by presenting a secure ICMetric based remote patient monitoring system for the health industry.

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 cryptosystem for ICMetric based applications is of utmost importance. The next chapter is a step in this direction and aims to address the very issue of designing an asymmetric key protocol that can create a link between the ICMetric applications and their security with asymmetric cryptosystems.

5

The ICMetric Asymmetric Key Cryptosystem

5.1 INTRODUCTION

Keys in cryptography are either symmetric or asymmetric. Symmetric key cryptography uses identical keys to perform encryption and decryption operations. Besides encryption/decryption asymmetric key cryptography has applications in digital signatures, SSL, PGP, etc. The asymmetric keying uses a public-private key pair to establish secure communications in cryptography. The generation of asymmetric keys is a computationally intensive operation, but this keying approach is favoured owing to the benefits it offers compared to symmetric keying. Asymmetric key generation is purely based on algorithmic intractability to ensure that the key pairs operate independently and it is computationally impossible for an adversary to extract a private key if the public key has been provided. The studies [9][38][39] presented in earlier

chapters of the thesis show that the adversaries will not always target algorithmic intractability to capture the keys. Therefore, there is need for a solution where the private key is never stored and is generated only when required. Thus, the asymmetric key is generated using the ICMetric technology as a strong root of trust, where keys are generated using unpredictable system features.

The previous chapter studied the generation of a symmetric key by using the ICMetric of a device. This chapter studies the design and implementation of a public key cryptographic algorithm based on the ICMetric technology. The proposed scheme uses the ICMetric of a device along with the RSA algorithm to generate a strong public-private key pair for ICMetric based applications. A combination of RSA and ICMetric technology is a novel concept that promises higher levels of security without excessive resource demand.

5.1.1 Asymmetric Ciphers

Asymmetric-key algorithms (or public-key algorithms) are a unique concept, owing to the way keys are held by the individual parties. The use of a public-private key pair was first explored by Diffie et al [104], where he laid the foundations for a range of applications and research in public key cryptography. An asymmetric key is composed of a public key and a private key. The private key is kept secret while the public key can be widely distributed. Owing to their unique design asymmetric keys possess the property that it is computationally infeasible to extract the private key if the public key is provided. Hence asymmetric keys are generated and utilized believing in the mathematical intractability of the key generation algorithm.

Public key encryption allows a communicating party A holding public key pk of some party B to encrypt a message m in ciphertext c such that only the party holding the corresponding private key sk (B usually) to decrypt the ciphertext c to m. It is formally defined as follows:

A public key encryption scheme $PKE = (KGenPK, Enc, Dec)$ consists of the following three algorithms:

- $KGenPK(\lambda)$ generates a private/public key-pair (sk, pk) on input of the security parameter λ.
- $Encpk(m, r)$ encrypts message m with public key pk and randomness r, and outputs ciphertext c.

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

There are two methods by which encryption is carried out using asymmetric keys. The two methods are discussed below:

- The sender uses a public key to encrypt the message while the decryption is carried out by the holder of the corresponding private key. This ensures that the message can only be decrypted by the owner of the private key. Secondly this scenario allows any party holding the intended recipients public key to carry out encryption. Hence there is no need for a formal key exchange as the public keys can be shared with minimum overhead.
- The sender uses his private key to encrypt the message while the decryption is carried out using the public key. This method of encryption ensures non repudiation since the decrypting parties are given guarantee that only the holder of the private key can encrypt the message. This method of encryption should

be used with caution as the message can be decrypted by all the parties that hold the public key.

Besides encryption asymmetric key cryptography can also be used in digital signatures; the message is signed with the sender's private key and anyone who has access to the public key can verify that the message was indeed sent by the owner of the private key. Common examples of asymmetric ciphers are RSA [105], Diffie Hellman [104], and Elliptic curve cryptography (ECC) [106].

5.1.2 RSA Cryptosystem

The RSA algorithm is one of the oldest and well established asymmetric cryptographic schemes. First published [105] in 1978, the RSA algorithm is considered the de facto asymmetric keying algorithm. Years of cryptanalysis has not been able to prove or disapprove the security of this algorithm [107]. Hence even today it is believed that the RSA algorithm is secure and cannot be broken. The RSA algorithm is composed of two modules i.e. asymmetric key generation and RSA encryption/ decryption.

The RSA key generation algorithm is mathematically intractable since its design is based on the integer factorization problem [108]. Thus, the problem states that given a large positive integer x, find the two factors that constitute the number. This problem is unsolvable in polynomial time owing to which it is used as a basis for key generation. This problem becomes even harder to solve in RSA, as the algorithm uses two large primes to form the large positive integer x. The RSA algorithm uses the Euler totient function as an integral part of key generation. The Euler totient function has unique properties associated with primes.

Definition 5.1 The Euler totient function [106] denoted by $\varphi(n)$ gives the number of integers less than n that are coprimes to n, where n is an integer greater than 1.

The Euler totient function is multiplicative and has the property that it is difficult to compute for non-primes, but is easy to compute for prime numbers. Hence if p is a prime number then the Euler totient function $\varphi(p)$ is equal to $p - 1$. The RSA key generation algorithm is composed of the following steps:

Step 1 Produce two large random primes p and q.

Step 2 Compute the RSA modulus $N = p * q$.

Step 3 Compute the Euler's totient function φ, to generate the private exponent.

$$\begin{aligned} \varphi\ (N) &= \varphi(p)\ \varphi(q) \\ &= (p-1)(q-1) \end{aligned} \tag{5.1}$$

Step 4 Choose an integer e such that $1 \leq e \leq \varphi(N)$ i.e. e and $\varphi(N)$ are coprimes.

Step 5 Compute the public exponent d.

$$d \equiv e^{-1}(mod(N)) \tag{5.2}$$

The public key exponent d is released while the private component φ is kept secret. A popular choice for the public e in the key generation algorithm is 65573 [109][110].

5.1.3 Security Goals

The proposed ICMetric public key cryptosystem aims to provide high levels of security in end-to-end communication environments using generated ICMetric public/private key pairs. The proposed scheme fulfils the following security goals:

- Confidentiality – A major goal of the proposed ICMetric based public key cryptosystem is to provide confidentiality of data. The proposed scheme uses an extended RSA encrypt/ decrypt algorithm for the provision of data secrecy.
- Non repudiation – Another goal of the proposed ICMetric based public key cryptosystem is non repudiation of data. The proposed ICMetric public key scheme uses the features of a system to create an ICMetric. The system's ICMetric is then used to create a public/private key pair. The ICMetric private key is used to perform encryption; and decryption is carried out using the corresponding ICMetric public key, thereby providing non repudiation. The key generation is based on the ICMetric of the entity, therefore the resulting cipher text can be traced back to a single source.

5.2 THE PROPOSED ICM-RSA CRYPTOSYSTEM

The proposed ICM-RSA cryptosystem details an ICMetric centred asymmetric key scheme that can be used with asymmetric key applications based on the ICMetric technology. The proposed scheme 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 scheme.

5.2.1 Key Generation

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

1. Choose two distinct prime numbers p and q.
    - The integers p and q are random. Prime integers can be efficiently found using a primality test.
2. Compute $n = pq$.

- n is used as the modulus for both the public and private keys.

3. Compute

$$\varphi(n) = lcm\big(\varphi(p), \varphi(q)\big) = lcm(p-1, q-1)$$

where φ is Carmichael's totient function. This value is kept private.

4. Set d as the ICMetric of the entity.
5. Check if d is prime, if not, set d as the next nearest prime number p such that there is no prime q such that $d < q < p$; using Miller–Rabin algorithm [111]
6. Check if d is $1 < d < \varphi(n)$ and $gcd\big(d, \varphi(n)\big) = 1$; i.e., d and $\varphi(n)$ are coprime.
7. Determine e as $e \equiv d^{-1}(mod\ \varphi(n))$; i.e., e is the modular multiplicative inverse of $d(modulo\ \varphi(n))$.

- e is released as the public key exponent.
- d is kept as the private key exponent.

The pair $\big(d, \varphi(n)\big)$ is the private key, while the pair (n, e) is the public key.

5.2.2 Encryption

The message m is represented as an integer in the interval $[0 \dots n-1]$ 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.

5.2.3 Decryption

To recover the message m from the ciphertext c. The entity can recover message as

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

5.3 SECURITY PROOF

This section presents the corollary and lemmas that help validate the security of the proposed ICM-RSA protocol against the standard RSA. This will lead to the overall security proof of the cryptosystem.

Corollary - 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.

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

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.

Proof of Sketch

To prove the security of ICM-RSA, we firstly prove that the above assumptions are strictly met in the proposed scheme. For this, we firstly assume that $\varphi(n)$ is known, so how can it lead to solving the integer factorization problem [108]. Solving the corresponding equations (1) and (2) 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 [108][106]. 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 scheme and hence ICM-RSA is secure.

5.4 IMPLEMENTATION AND EVALUATION

This section evaluates the efficiency of the proposed extended ICM-RSA design. It starts by discussing the implementation aspects of the cryptosystem and then goes on to evaluate the performance of the ICM-RSA key generation module and the encryption/decryption module. The implementation of the ICMetric based public key cryptosystem has been done using JAVA on Linux. The scheme has been implemented on a first-generation Intel Core i3 3.2 GHz processor with 6 GB RAM. Detailed results in the upcoming section show that the designed ICMetric public key cryptosystem can achieve high levels of security without extensive resource demands.

5.4.1 ICM-RSA Key Generation

The evaluation of the proposed ICM-RSA key generation scheme is done by comparing it with the performance of the standard RSA key generation algorithm. As key sizes are increasing, therefore bigger key sizes have been simulated as a baseline. The simulated key sizes for RSA based asymmetric keys are 1024, 2048, 3072, 4096 bits respectively. 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. Figure 5.1 is a

comparison of the standard RSA and the ICMetric based RSA (ICM-RSA) key generation algorithm when generating 1024 bit keys. It depicts the range of time taken for generating 1024 bit keys. 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 (5) and (6) of the ICM-RSA key generation scheme. At the same time, it must be stated that the use of the ICM-RSA algorithm is justified owing to the benefits it offers.

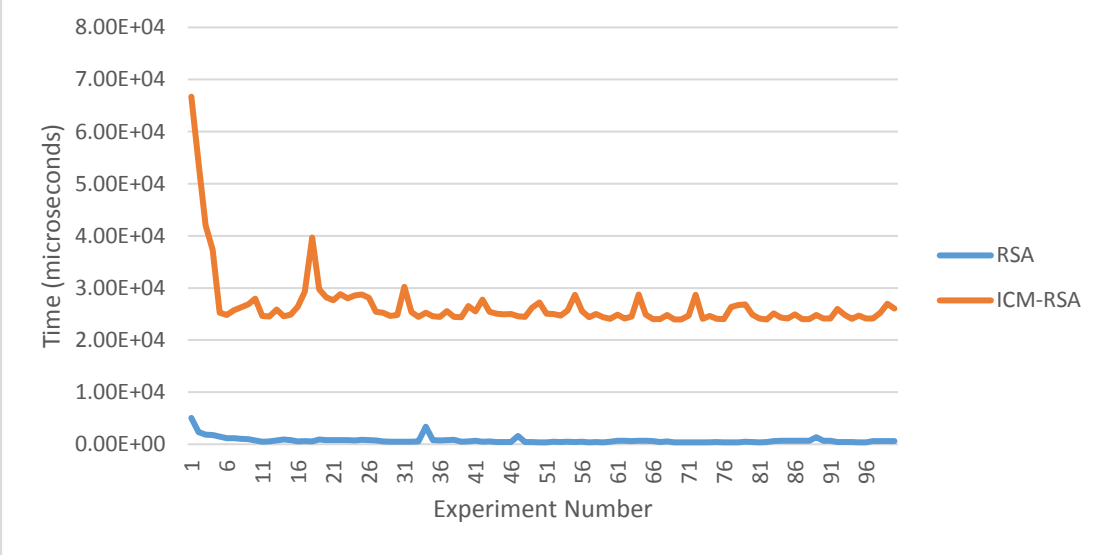


Figure 5.1. Execution Time Comparison of the RSA Key Generation with Extended RSA Key Generation for a 1024 bit Key

Minor fluctuations were observed when executing the key generation algorithm. The spikes are present because of the probabilistic nature of the scheme, and therefore the execution time may vary for each run of the program. Therefore, the average time for the ICM-RSA and RSA was computed as an indicator of expected performance. Table 5.1 provides the average time consumption of the proposed algorithm and the standard RSA. There is a notable difference in execution time but the ICMetric based scheme is justified owing to security rewards.

Table 5.1. Average Execution Time Comparison of the RSA Key Generation with ICMetric-RSA for the Generation of 1024 bit Key

Key Generation Scheme	RSA	ICM-RSA
Average Time (microseconds)	718	26648

When studying the 2048-bit key generation it is evident that the standard RSA algorithm outperforms ICM-RSA. The ICMetric based asymmetric key generation algorithm takes on average 119253.7 microseconds. 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. It must also be noted that the execution time for the first run of the program is quite high because of the additional resources JAVA takes for the initialization of the JVM.

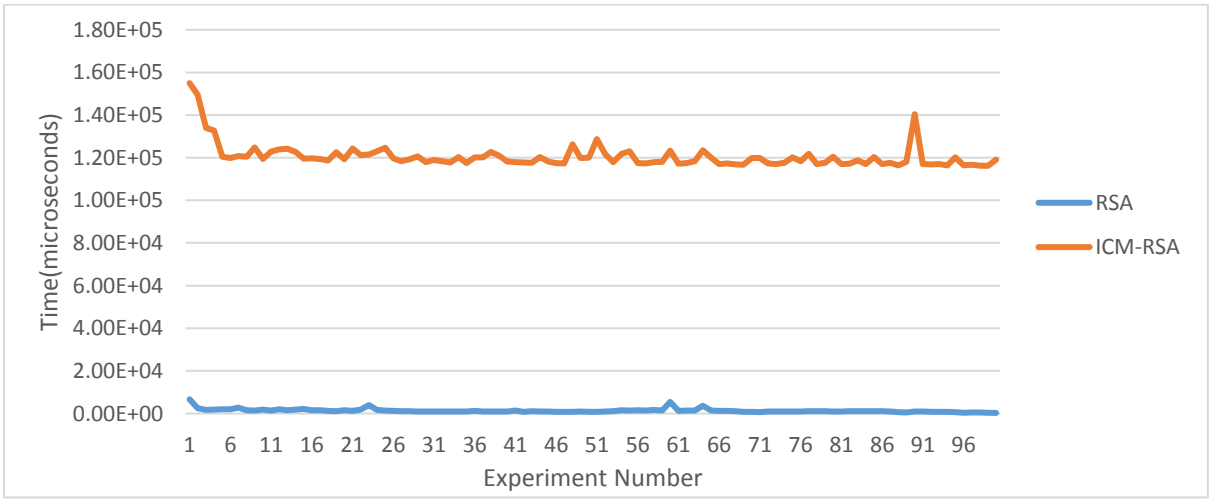


Figure 5.2. Execution Time Comparison of the RSA Key Generation with Extended RSA key Generation for a 2048 bit Key

Therefore, the average execution time is computed as an indicator of the algorithms performance. Table 5.2 gives the average execution time of both the RSA and ICM-RSA algorithm for the 2048 bit key.

Table 5.2. Average Execution Time Comparison of the RSA Key Generation with ICMetric-RSA for the Generation of 2048 bit Key

Key Generation Scheme	RSA	ICM-RSA
Average Time (microseconds)	1326.398	119253.7

The 3072 bit key size is simulated to determine how the ICMetric technology would influence futuristic applications where the key sizes can be 3072 bit 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. Figure 5.3 gives the graph of 3072-bit key generation for both ICMetric implementations. When generating the 3072-bit key the average execution time for RSA is 1588.023 microseconds while that of ICM-RSA is 332561.1 microseconds. Once again as mentioned earlier, there is a sharp drop in the execution time after the first run of the program. This trend is observed in all the upcoming graphs and is because of the time taken by JAVA in initialization of JVM; and the additional resources required by JAVA for converting the byte code to machine code during the first run of the program.

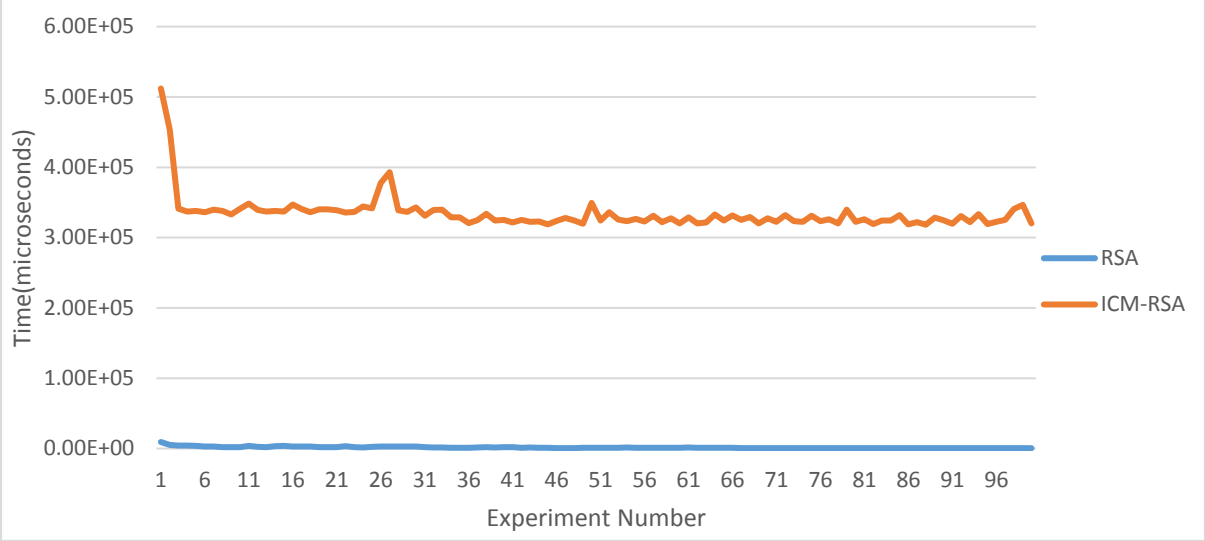


Figure 5.3. Execution Time Comparison of the RSA Key Generation with Extended RSA Key Generation for a 3072 bit Key

The computed average time for key generation using RSA is very moderate while the ICM-RSA schemes average execution time is slightly elevated. Table 5.3 gives the average execution time comparison for both the RSA and ICM-RSA algorithms.

Table 5.3. Average Execution Time Comparison of the RSA Key Generation with ICMetric-RSA for the Generation of 3072 bit Key

Key Generation Scheme	RSA	ICM-RSA
Average Time (microseconds)	1588.023	332561.1

Like the previous key sizes the RSA algorithm outperforms the ICM-RSA when considering the 4096-bit key. 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. Figure 5.4 gives a graph showing the performance comparison of the proposed and rivalling scheme when generating 4096 bit keys.

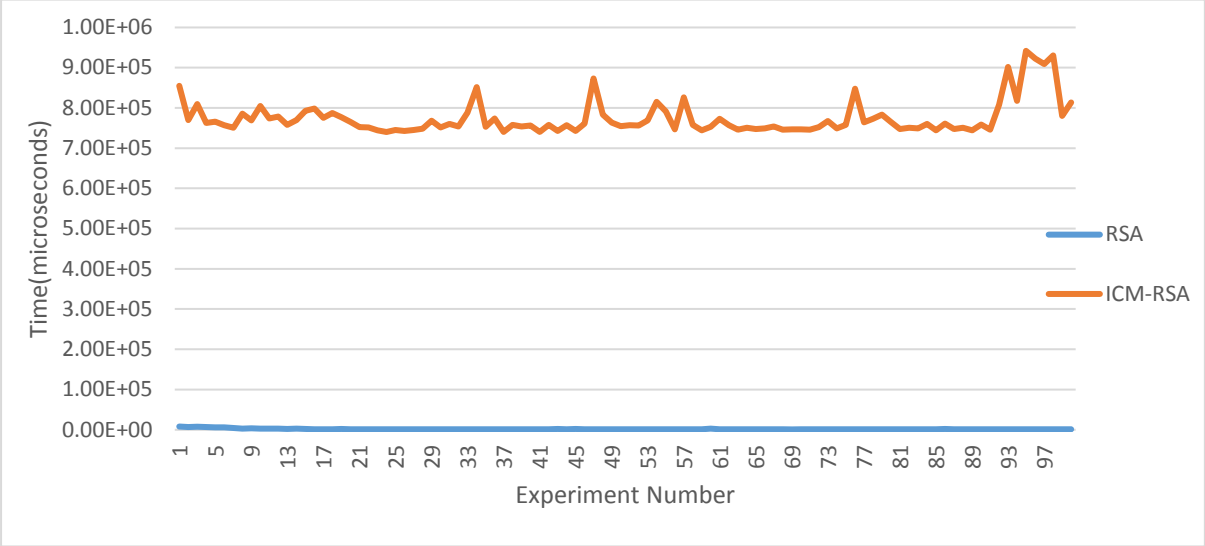


Figure 5.4. Execution Time Comparison of the RSA Key Generation with ICMetric-RSA Key Generation for a 4096 bit Key

Considerable fluctuations in execution time are analysed by computing the average execution time. After a large number of iterations, the average is computed for both the proposed scheme 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. Table 5.4 gives a summary of the average execution time for both competing schemes.

Table 5.4. Average Execution Time Comparison of the RSA Key Generation with ICMetric-RSA for the Generation of 4096 bit Key

Key Generation Scheme	RSA	ICM-RSA
Average Time (microseconds)	1704.008	773117.61

5.4.2 ICM-RSA Encryption/Decryption

To understand the effect of key size on encryption and decryption operations, the 1024 bit key is tested using both the proposed and the standard RSA schemes. It is evident from the superimposed graph in Figure 5.5 that the standard RSA is faster while performing encryption/ decryption.

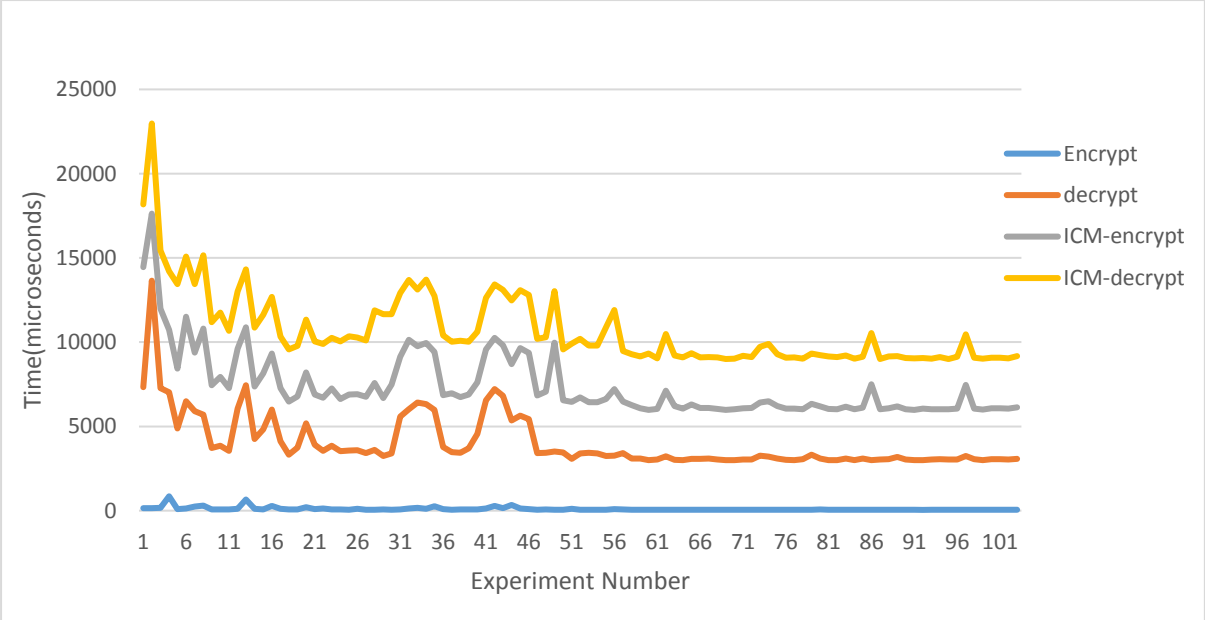


Figure 5.5. Execution Time Comparison of RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 1024 bit Key

Considerable fluctuations in execution time have been seen when comparing the proposed scheme and the basic RSA encryption/ decryption using 1024-bit key. After a fairly large number of iterations, the average is computed for both the proposed scheme and the rivalling scheme. It can be concluded that the standard RSA algorithm is faster than the proposed scheme but both are competitively close in average execution time presented in Table 5.5.

Table 5.5. Average Execution Time Comparison of the RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 1024 bit ICMetric Key

Scheme	RSA Encrypt	RSA Decrypt	ICM-RSA Encrypt	ICM-RSA Decrypt
Average Time (microseconds)	97.08019	3925.508	3388.564	3327.587

A comparison of RSA encryption/ decryption with the proposed ICMetric based encryption/ decryption shows very competitive results. Figure 5.6 gives a graph of the proposed scheme performance comparison with the standard RSA using a 2048 bit key. Both schemes are very competitive in performance although the standard RSA outperforms the proposed scheme.

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.

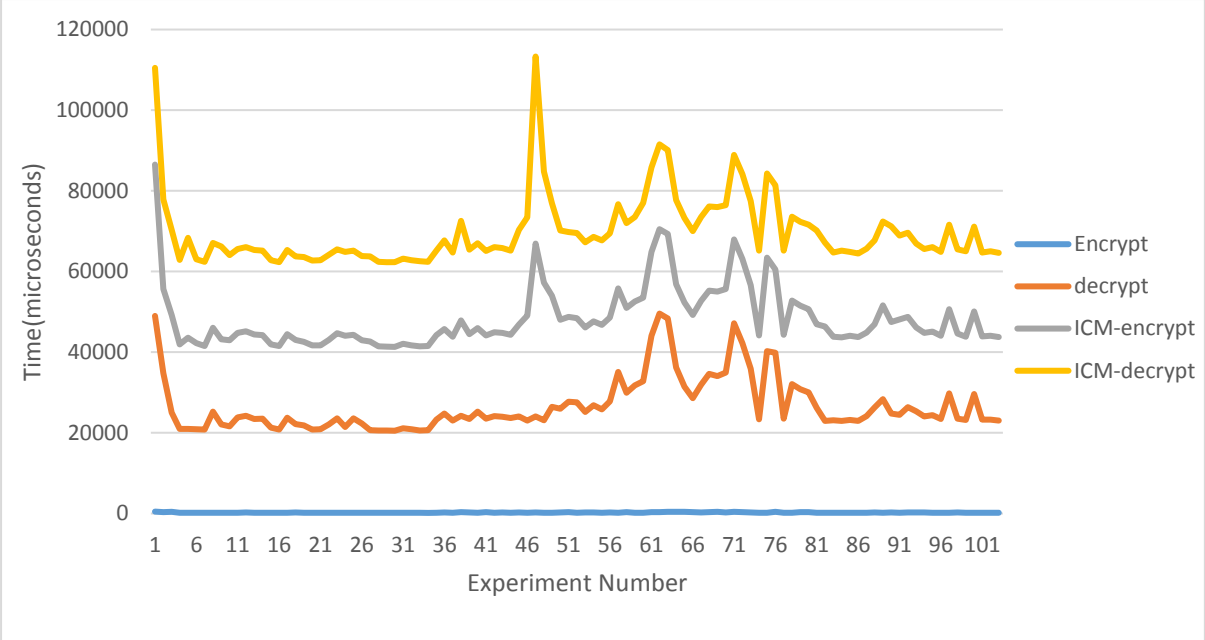


Figure 5.6. Execution time Comparison of RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 2048 bit Key

As there are frequent fluctuations in execution time therefore the average execution time is computed. Results in Table 5.6 show that the 2048-bit encryption/ decryption algorithm is very competitive whether carried out using the standard RSA or the proposed ICMetric based RSA encryption/ decryption.

Table 5.6. Average Execution Time Comparison of the RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 2048-bit ICMetric Key

Scheme	RSA Encrypt	RSA Decrypt	ICM-RSA Encrypt	ICM-RSA Decrypt
Average Time (microseconds)	237.1884	26479.31	21702.7721	21547.24376

The 3072-bit key is used to see its effect on encryption/ decryption using the standard RSA and the proposed ICMetric based scheme. The basic RSA algorithm used for encryption/ decryption is faster in operations as compared to the proposed ICMetric

based scheme. Figure 5.7 is a superimposed graph showing the encryption/ decryption operation carried out using two rivalling schemes executed using 3072 bit keys.

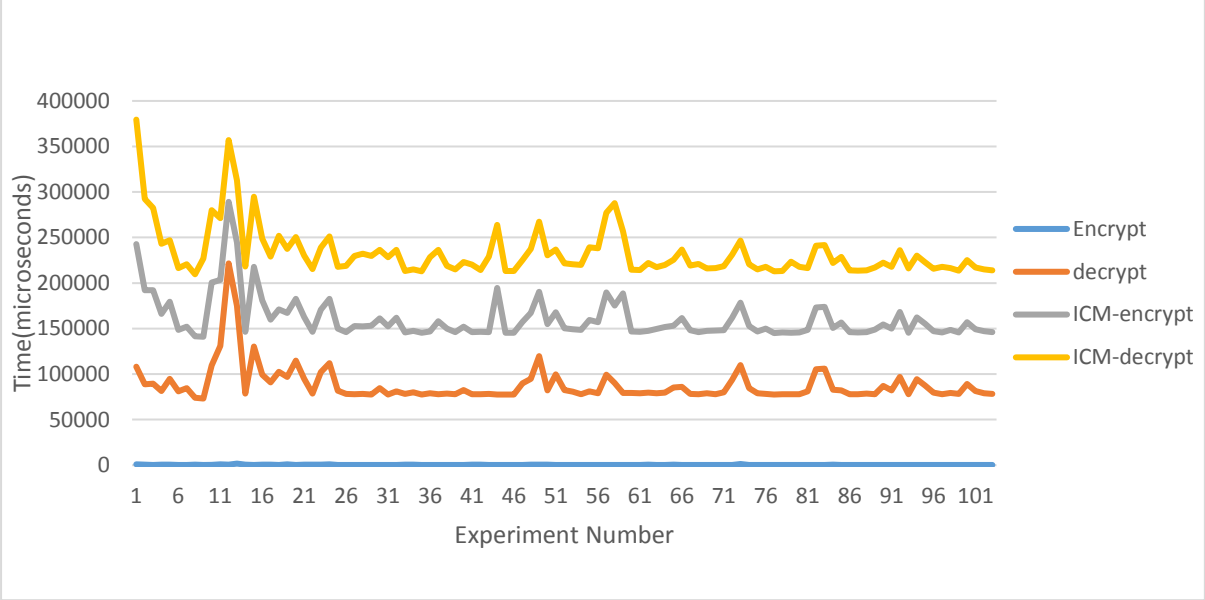


Figure 5.7. Execution time Comparison of RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 3072-bit ICMetric Key

Table 5.7 gives the average time required for both the RSA and ICMetric based encryption/ decryption using the 2048 bit key.

Table 5.7. Average Execution Time Comparison of the RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 3072-bit ICMetric Key

Scheme	RSA Encrypt	RSA Decrypt	ICM-RSA Encrypt	ICM-RSA Decrypt
Average Time (microseconds)	476.077	87518.54	72815.14992	72580.17339

While performing the encryption/ decryption using the 3072 key generation algorithm fluctuations were seen. Therefore, the average execution time is computed as an indicator of the algorithms performance.

The 4096 bit key is used to simulate the encryption/ decryption using the standard RSA and the ICMetric based schemes. The basic RSA confidentiality algorithm is closely related in terms of time efficiency compared to the proposed ICMetric based scheme. Figure 5.8 gives the graph of 4096 bit key for both ICMetric based scheme and the standard RSA encryption/ decryption. Both rivalling schemes are competitively placed when considering their performance. Here it can be said that system designers should not hesitate in choosing the ICMetric based scheme for encryption/ decryption.

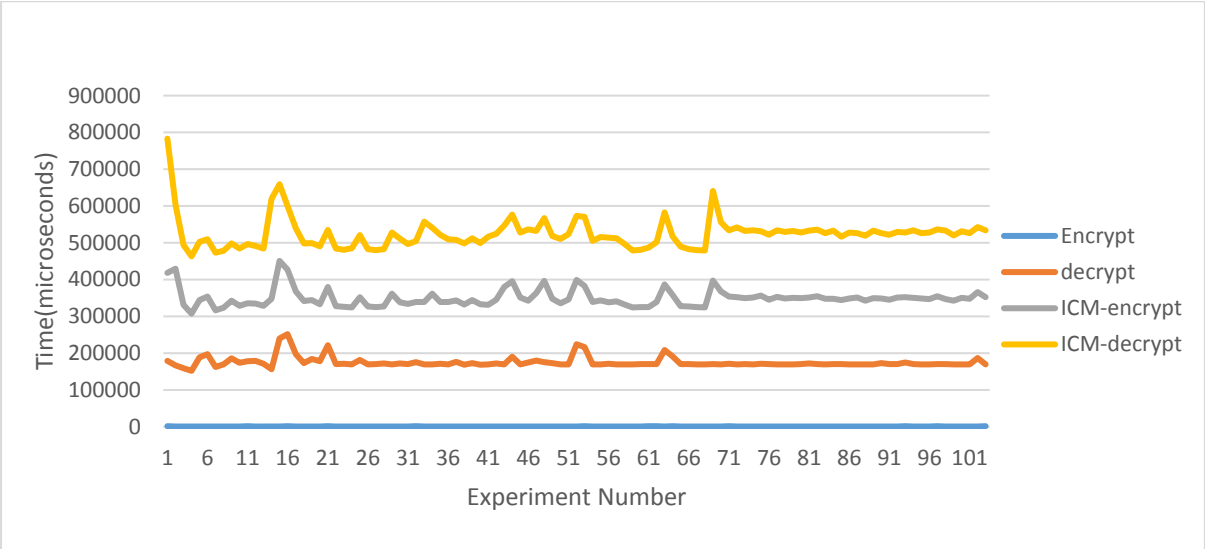


Figure 5.8. Execution Time Comparison of RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 4096-bit ICMetric Key

The computed average time for performing encryption/ decryption using RSA is better than the proposed ICMetric based confidentiality scheme. Table 5.8 gives the average execution time comparison for both the RSA and the RSA based encryption/ decryption scheme using 4096 bit key. Individual readings show that the RSA algorithm outperforms but not by a great difference.

Table 5.8. Average Execution Time Comparison of the RSA Encryption/Decryption with ICMetric-RSA Encryption/Decryption using a 4096-bit ICMetric Key

Scheme	RSA Encrypt	RSA Decrypt	ICM-RSA Encrypt	ICM-RSA Decrypt
Average Time (microseconds)	750.5248	174473.9	174108.2848	175274.6582

5.5 SUMMARY

Asymmetric key cryptography is a powerful tool which can provide guarantees that are not possible with symmetric key cryptography. Asymmetric key cryptography uses a unique combination of public and private key for the provision of security. The public key is shared unrestricted but the private key is kept secret. Like all cryptographic algorithms asymmetric keys rely on mathematical intractability for the generation of the public-private key pair. Thus, the mathematical intractability is the basis for the inability of an adversary to extract the private key from the public key. A popular asymmetric algorithm is the RSA. This RSA key generation uses two large primes for computing a modulus. Although this is computationally hard to break it must be acknowledged that adversaries do not always rely on algorithmic intractability to capture the keys. Owing to system weaknesses it may be easier for an adversary to capture the stored keys.

This chapter demonstrates the use of the ICMetric technology for asymmetric key generation. The key generation scheme uses the ICMetric of a system to create a public and private key pair. The proposed scheme is based on the RSA key generation which is well recognized owing to its high levels of security. Since the RSA key generation algorithm is highly stable therefore efforts have been made to ensure that

the algorithm itself is not excessively modified, but has been very carefully tuned to work with ICMetric technology.

When performing, encryption using asymmetric keys there are two possible methods. The method that provides non repudiation along with encryption is private key encryption followed by public key decryption. By using the ICMetric technology with RSA one can say that a stronger non repudiation guarantee is provided since the ICMetric of a device cannot be predicted or cloned. Hence the ICMetric technology can be used for non-repudiation which is not a feature of Physically Unclonable Functions.

The proposed cryptosystem has been simulated and tested to show that ICMetric based asymmetric keys can be generated without excessive resource demand. The scheme has been tested by generating keys of various sizes i.e. 1204, 2048, 3072, 4096. The generated keys are then used to perform encryption and decryption of data. The generated key sizes using the ICMetric based RSA have been compared with the standard RSA key generation algorithm. 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.

A technology loses its purpose if its application is not demonstrated in real environments. The next chapter is a step in this direction and showcases the use of the ICMetric technology for the Docker containers employed by a leading cloud service provider for hosting their cloud services. It focuses on deriving keys from the operating characteristics of Docker containers and creating a secure Docker environment via the ICMetric technology.

6

Case Study: Docker Security Using ICMetrics

Docker provides a way to run almost any application securely isolated in a container. The isolation and security allow the running of many containers simultaneously on a server. The use of Docker containers [112] enables global cloud service providers to run their cloud provisioning services across multiple platforms. However, for cloud service providers to use Docker safely for managing their containers, there are potential security issues that need to be taken care of. The security in Docker containers poses many challenges in creating a stable, secure and reliable system that can be used to ensure such security. A major challenge associated with using Docker containers safely is to detect any change in the working of the container due to introduction of malicious or abnormal behaviour. This chapter investigates the feasibility of employing the ICMetric technology for deriving keys from the operating characteristics of Docker containers. It studies enabling a secure

Docker environment via the ICMetric technology. The ICMetric technology creates a runtime profile of the running container to detect any change in the runtime behaviour of the Docker container being used. The technique effectively allows the use of the distinct characteristics that are generated even for identical containers. A major novelty and security enhancing feature of the ICMetric system is that the measured characteristics need not remain constant, but are free to vary within deduced parameters, thus allowing the software to operate in several states and on a variety of platforms, whilst still ensuring that any illegal clone or malware infecting the software is detected because of unacceptable changes in the operating parameters. This acts as a security enhancing feature, in that an attacker cannot easily derive the information to reproduce the characteristics to break the ICMetric by observing the system, especially as characteristics whose acceptable values vary over time are typically employed.

ICMetric identification for Docker containers represents a novel concept of regulating access to services and can produce assurance-providing protection at especially the vulnerable points. Evaluation and resilience in the presence of malware on a Docker container is a novel concept, and ICMetric has been proposed as a process for managing these adverse consequences effectively. Malware on a Docker container can be detected due to the change reflected in the software constitution, thereby changing the underlying feature values which are used to generate the ICMetric.

A major contribution of this chapter is a discussion on the operation of the ICMetric technology in Docker containers. As the security of an ICMetric based system is founded on its features, therefore a comprehensive discussion on ICMetric features for Docker containers has been presented in the chapter. The chapter starts off by discussing and analysing Docker container namespaces and cgroups features. Then it goes on to do a comprehensive analysis of these container namespaces and cgroups

features. The last part of the chapter uses the selected Docker container features in the formation of ICMetric for identification of Docker containers.

6.1 DOCKER CONTAINER

Containers [113] are used to provide a virtualized computation environment over a single Linux kernel. They are implemented through the use of two kernel functionalities namely, namespaces and cgroups. Docker uses Linux cgroups and namespaces, to control and run its containers. Features in these namespaces and cgroups can potentially be used in the generation of ICMetric key for identification of a Docker container.

6.1.1 Namespaces

Namespaces [113] wrap a global system resource in an abstraction. This abstraction gives processes within the namespace an illusion that they have their own isolated instance of the global resource. Currently Linux implements six types of namespaces namely; mount, PID, user, IPC, UTS and Net. Docker uses four of the six Linux namespaces; namely PID, IPC, UTS, and Net namespace. Some of the namespaces that Docker Engine uses on Linux are:

- The Process ID (PID) namespace manages process isolation.
- The Network (Net) namespace manages network interfaces and provides isolation of the system resources associated with networking, e.g., network devices, IP addresses, IP routing tables, port numbers, and so on.
- The inter-process communication (IPC) namespace is responsible for managing access to inter-process communication resources.

- The UTS namespace is responsible for isolating kernel and version identifiers such as hostname and domain name.

6.1.2 Cgroups

Docker engine also makes use of cgroups or control groups to track groups of processes; and to expose metrics about CPU, memory and block input/ output usage [113]. Control groups allow the Docker engine to share available hardware resources to containers, while setting up limits and constraints. Some of the subsystems managed by cgroups are:

- blkio - this subsystem sets limits on input/output access to and from block devices such as physical drives (disk, solid state or USB).
- cpu - this subsystem uses the scheduler to provide cgroup tasks access to the CPU, while also keeping track of system CPU time.
- cpuacct - this subsystem generates automatic reports on CPU resources used by tasks in a cgroup.
- cpuset - this subsystem assigns individual CPUs (on a multicore system) and memory nodes to tasks in a cgroup.
- devices - this subsystem allows or denies access to devices used by tasks in a cgroup.
- memory - this subsystem sets limits on memory used by tasks in a cgroup and generates automatic reports on memory resources used by those tasks in the container.
- freezer - this subsystem suspends or resumes tasks in a cgroup.

6.2 Methodology

The ICMetric system allows the use of the distinct characteristics that are generated for a container in the Docker environment. Example characteristics are the features found in the Docker namespaces and Docker cgroups. Features of the Docker containers form the foundation of the approach to create an ICMetric for the Docker container. There are certain aspects of each feature that are considered to determine their adequacy. When considering the employment of features, the individual features are thoroughly examined to ensure that they can provide useful data. The feature data is tested to validate, that the varying results produced are viable to contribute to the key generation. If a feature provides enough inter-sample variance it is selected for further examination. The selected features are used in the formation of an ICMetric for identification of a Docker container.

The analysis of Docker containers, allows the possibility of finding features which can provide an adequate dynamic range, obfuscation and variance. The analysis requires the collection of data from multiple Docker containers, to generate a diverse range for each feature value. If the selected feature provides enough inter-sample variance it is selected for further examination. The selected features are further analysed to find a high correlation with other container features.

6.2.1 Feature Data Collection

The ICMetric technology uses distinct characteristics that are generated for a running Docker container by monitoring the runtime behaviour of a container being used. Example characteristics are CPU usage, memory usage, input/ output usage and network attributes. Runtime Docker features are the foundation of the approach to

creating secure ICMetric keys. There are certain aspects of each feature that need to be considered to determine their adequacy. When considering the employment of new features, they must be thoroughly examined to ensure that they can provide useful data and then tested to validate that the varying results they produce are viable to contribute to the generation of an ICMetric key.

6.2.2 Feature List

Docker containers are implemented using two kernel functionalities namely, namespaces and cgroups. The features from Docker namespaces and cgroups are narrowed down via their observed variations in value from a large collection of candidate features belonging to a range of namespace and cgroup categories. Table 6.1 lists all the Docker container features that showed good variation and are considered potentially useful for ICMetric key generation.

Table 6.1. Docker Container Features

1. precpu_stats/cpu_usage/6_usage	2. precpu_stats/cpu_usage/percpu_usage/0
3. precpu_stats/cpu_usage/percpu_usage/1	4. precpu_stats/cpu_usage/percpu_usage/2
5. precpu_stats/cpu_usage/percpu_usage/3	6. precpu_stats/cpu_usage/usage_in_usermode
7. precpu_stats/cpu_usage/usage_in_kernelmode	8. precpu_stats/system_cpu_usage
9. cpu_stats/cpu_usage/6_usage	10. cpu_stats/cpu_usage/percpu_usage/0
11. cpu_stats/cpu_usage/percpu_usage/1	12. cpu_stats/cpu_usage/percpu_usage/2
13. cpu_stats/cpu_usage/percpu_usage/3	14. cpu_stats/cpu_usage/usage_in_kernelmode
15. cpu_stats/cpu_usage/usage_in_usermode	16. cpu_stats/system_cpu_usage
17. memory_stats/stats/pgfault	18. memory_stats/stats/pgpgin

19. memory_stats/stats/pgpgout	20. memory_stats/max_usage
21. memory_stats/stats/cache	22. memory_stats/stats/inactive_file
23. memory_stats/stats/mapped_file	24. memory_stats/stats/pgmajfault
25. blkio_stats/io_service_byte_recursive/0/value	26. blkio_stat/io_service_byte_recursiv/3/value
27. blkio_stats/io_service_bytes_recursive/4/value	28. blkio_stats/io_serviced_recursive/0/value
29. blkio_stats/io_serviced_recursive/3/value	30. blkio_stats/io_serviced_recursive/4/value
31. networks/eth0/tx_bytes	32. networks/eth0/tx_packets

Utilizing the captured data shown in the table above, two different types of analysis were completed: 1) inter-sample and 2) intra-sample. The main motivation here was to identify those features that show high inter-sample and low intra-sample variation. The tables below show a selection of these features analysed in terms of their intra-sample and inter-sample variance, thereby deciding on their viability for ICMetric generation. CPU features provide information on the CPU statistics of the Docker container. These values are stable within the same container, and therefore produce low intra-sample variance. The read values indicated that the obtained values exhibited low intra-sample variance and high inter-sample variance, which is desirable, as shown in Table 6.2 below.

Table 6.2. Analysis of Selection of CPU based Docker Container Features

Parameter	**Intra-sample**	**Inter-sample**	**Comments**
cpu_stats/system_cpu_usage	stable	varying	Potentially useful
cpu_stats/cpu_usage/percpu_usage/1	stable	varying	Potentially useful

precpu_stats/system_cpu_usage	stable	varying	Potentially useful

Memory features provide information on memory related statistics of the Docker container. Some obtained memory features vary within the same container and remain same among other containers, hence potentially not useful for the ICMetric generation. However, some of the obtained memory feature values were stable after multiple runs within the same container, whereby leading to a low intra-sample variance and high inter-sample variance, as evident from Table 6.3.

Table 6.3. Analysis of Selection of Memory based Docker Container Features

Parameter	**Intra-sample**	**Inter-sample**	**Comments**
memory_stats/max_usage	varying	varying	Potentially not useful
memory_stats/stats/active_file	varying	varying	Potentially not useful
memory_stats/stats/6_inactive_file	stable	varying	Potentially useful
memory_stats/stats/pgpgin	stable	varying	Potentially useful

The networks category has data on network related statistics of the Docker container. In a controlled environment, these otherwise potentially variable values are considered static offering low intra-sample variance, as shown in Table 6.4.

Table 6.4. Analysis of Selection of CPU based Docker Container Features

Parameter	**Intra-sample**	**Inter-sample**	**Comments**
networks/eth0/rx_bytes	stable	varying	Potentially useful

networks/eth0/tx_packets	stable	varying	Potentially useful
network settings/sandboxed	varying	varying	Potentially not useful

6.2.3 Feature Correlation Analysis

Correlation is the measure of inter-relationship between two features. Correlation is defined as the average of the product of the difference between the feature values and their respective means. For T_i samples of a Docker container it is mathematically represented as:

$$\frac{1}{T_i}\sum_{h=1}^{T_i}(x_{1h}-\mu_1)\,(x_{2h}-\mu_2)$$

The feature correlation analysis was carried out by looking at correlations among the thirty-two cgroup features which showed variance. Figure 6.1 depicts the 56 x 56 correlation matrices for the cadvisor Docker container. Each row of this 56 x 56 correlation matrix gives the correlation results of a specific feature with all the other features.

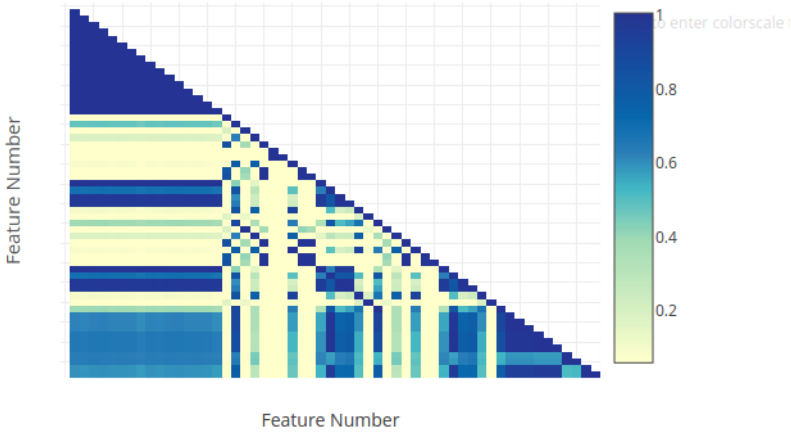


Figure 6.1. Visualisation of 56x56 Correlation Matrices for Docker containers

For example, the first row shows the correlation results for the Feature 1 (precpu_stats/cpu_usage/6_usage in this case) with all the other 55 features and itself. Similarly, the last row depicts the correlation results for the Feature 52 (blkio_stats/io_serviced_recursive/4/value in this case) with all the other 55 features and itself. A significant correlation can be found between many of the features as shown by the areas in darker blue shades.

Correlated features are a combination of singular features as they provide a greater level of obfuscation than only singular features. This adds an extra level of protection when recreating the values, as they must be generated and can't be read directly from a device. Each correlated feature can itself be used as a feature, which has the benefit of increasing the entropy of the key, generated by the ICMetric algorithm. Additionally, the correlated feature is likely to be more stable than the singular feature as they represent a relationship rather than coming from a specific range, meaning that there is less intra-sample variance thus increasing reproducibility of the generated key. Another significant aspect of correlated features is their ability to help distinguish containers. Singular features have a higher chance of having an overlap when the possible range for the feature is analysed across multiple Docker containers. The analysis we have completed looks at correlations between two features.

Table 6.5. Docker Container Feature List for Correlation Analysis

1.precpu_stats/cpu_usage/6_usage	2.precpu_stats/cpu_usage/percpu_usage/0
3.precpu_stats/cpu_usage/percpu_usage/1	4.precpu_stats/cpu_usage/percpu_usage/2
5.precpu_stats/cpu_usage/percpu_usage/3	6.precpu_stats/cpu_usage/usage_in_kernelmode
7.precpu_stats/cpu_usage/usage_in_usermode	8.precpu_stats/system_cpu_usage

9.cpu_stats/cpu_usage/6_usage	10.cpu_stats/cpu_usage/percpu_usage/0
11.cpu_stats/cpu_usage/percpu_usage/1	12.cpu_stats/cpu_usage/percpu_usage/2
13.cpu_stats/cpu_usage/percpu_usage/3	14.cpu_stats/cpu_usage/usage_in_kernelmode
15.cpu_stats/cpu_usage/usage_in_usermode	16.cpu_stats/system_cpu_usage
17.memory_stats/usage	18.memory_stats/max_usage
19.memory_stats/stats/active_anon	20.memory_stats/stats/active_file
21.memory_stats/stats/cache	22.memory_stats/stats/hierarchical_memory_limit
23.memory_stats/stats/hierarchical_memsw_limit	24.memory_stats/stats/inactive_anon
25.memory_stats/stats/inactive_file	26.memory_stats/stats/mapped_file
27.memory_stats/stats/pgfault	28.memory_stats/stats/pgmajfault
29.memory_stats/stats/pgpgin	30.memory_stats/stats/pgpgout
31.memory_stats/stats/rss	32.memory_stats/stats/rss_huge
33.memory_stats/stats/swap	34.memory_stats/stats/6_active_anon
35.memory_stats/stats/6_active_file	36.memory_stats/stats/6_cache
37.memory_stats/stats/6_inactive_anon	38.memory_stats/stats/6_inactive_file
39.memory_stats/stats/6_mapped_file	40.memory_stats/stats/6_pgfault
41.memory_stats/stats/6_pgmajfault	42.memory_stats/stats/6_pgpgin
43.memory_stats/stats/6_pgpgout	44.memory_stats/stats/6_rss
45.memory_stats/stats/6_rss_huge	46.memory_stats/stats/6_swap
47.blkio_stats/io_service_bytes_recursive/0/value	48.blkio_stats/io_service_bytes_recursive/3/value
49.blkio_stats/io_service_bytes_recursive/4/value	50.blkio_stats/io_serviced_recursive/0/value
51.blkio_stats/io_serviced_recursive/3/value	52.blkio_stats/io_serviced_recursive/4/value
53.networks/eth0/rx_bytes	54.networks/eth0/rx_packets

55.networks/eth0/tx_bytes	56.networks/eth0/tx_packets

In Figure 6.1, the colour of each pixel/small block shows how a specific feature is correlated to another feature. For example, the feature 1 (precpu_stats/cpu_usage/6_usage) is positively correlated with feature 10 (cpu_stats/cpu_usage/percpu_usage/0) and is shown by bright blue colour. On the other hand, feature 23 (memory_stats/stats/hierarchical_memsw_limit) has zero correlation with feature 19 (memory_stats/stats/active_anon) and is therefore shown by lime colour. The aim of these visualisation matrices is to find reliable correlations. By observing the visualisation matrices, the following pairs of features showed a strong positive correlation with a Pearson correlation coefficient over 0.8, thus indicating that they have a sufficient correlation to be considered. Memory_stats/stats/swap (33) and memory_stats/max_usage (18) are directly correlated with each other and have an average correlation coefficient 0.89; similarly, cpu_stats/cpu_usage/usage_in_kernelmode (14) and precpu_stats/cpu_usage /percpu_usage/3 (5) are strongly correlated with a coefficient of 0.99. Table 6.6 shows the analysed and selected Docker container features with their entropies.

Table 6.6. Feature Entropy for Docker Containers

Feature	**Entropy (in bits)**
precpu_stats/cpu_usage/percpu_usage/0	35
precpu_stats/cpu_usage/percpu_usage/1	34
precpu_stats/cpu_usage/percpu_usage/2	35
precpu_stats/cpu_usage/percpu_usage/3	35
precpu_stats/cpu_usage/usage_in_usermode	36

precpu_stats/cpu_usage/usage_in_kernelmode	35
precpu_stats/system_cpu_usage	46
cpu_stats/cpu_usage/percpu_usage/0	35
cpu_stats/cpu_usage/percpu_usage/1	34
cpu_stats/cpu_usage/percpu_usage/2	35
cpu_stats/cpu_usage/percpu_usage/3	35
cpu_stats/cpu_usage/usage_in_kernelmode	35
cpu_stats/cpu_usage/usage_in_usermode	36
cpu_stats/system_cpu_usage	46
memory_stats/stats/pgpgin	25
memory_stats/stats/pgpgout	24
memory_stats/max_usage	24
memory_stats/stats/cache	20
memory_stats/stats/inactive_file	10
memory_stats/stats/mapped_file	16
memory_stats/stats/pgmajfault	16
blkio_stats/io_service_bytes_recursive/0/value	25
blkio_stats/io_service_bytes_recursive/3/value	25
blkio_stats/io_service_bytes_recursive/4/value	25
blkio_stats/io_serviced_recursive/0/value	11
blkio_stats/io_serviced_recursive/3/value	11
ID	256
MACaddress	41
blkio_stats/io_serviced_recursive/4/value	11
networks/eth0/tx_bytes	11
networks/eth0/tx_packets	4
network settings/sandboxid	256

network settings/sandboxkey	256
network settings/EndpointID	256
Network settings/Networks/bridge/NetworkID	256
Network settings/Networks/bridge/EndpointID	256
Network settings/Networks/bridge/MacAddress	256

6.3 ICMETRIC KEY GENERATION

The ICMetric key generation is a lightweight process that involves the gathering of feature values from the Docker containers. The features are combined and go through the ICMetric normalisation algorithm, this obfuscates and normalises each feature thereby producing an ICMetric key [50] [51]. The generated ICMetric key is thereby used in performing identification in Docker.

A proof of concept demonstrator was also developed to convey how ICMetric technology can be used in a practical scenario to add security. The demonstration has been created to explicitly illustrate how ICMetric generated keys can be used to provide identification in Docker containers. For testing, three completely different types of containers were used to prove the successful working of the ICMetric technology for Docker container security; namely cadvisor, tcpdump and python. The demonstrator implements ICMetric for the generation of private keys and therefore does not require the storage of the said keys.

The demonstrator has been designed to showcase the following aspects of a practical implementation of ICMetrics:

1. A Docker container utilising ICMetric can successfully identify itself.
2. The integration of ICMetrics for Docker container identification.

These points focus on ICMetric innate ability to confirm the identity of the Docker container themselves, rather than confirming the validity of a stored private key. This provides the inherent ability to implicitly detect any tampering of the image associated with the Docker container via the inclusion of spyware or similar malware software, since this will cause the Docker container to produce a varied ICMetric key.

In the calibration phase of the ICMetric generation process, the Docker container stats data is periodically gathered, in this case every minute. The collected stats can be used for further analysis and ICMetric generation. The stats values for all containers are aggregated in such a manner that only the values that are common to all the containers are kept for the ICMetric key generation for Docker containers. The ICMetric is generated based on the selected container data samples and its feature values.

Now the ICMetric is generated for all three Docker containers and can be used for identification of Docker containers. The technique used for ICMetric identification is the multivariate normal distribution [114][115] and was the core of the demonstrator for security of Docker containers. The demonstrator was able to show that the ICMetric technology can be used to provide security in Docker containers.

The final stage, involved introducing some random data to the Docker container classes. This was done purposely to prove that malicious activity can be detected by the designed system based on the calculated ICMetric values. This step of the demonstrator proved that malicious activity can be detected since the ICMetric values did not match. The demonstrator shows that transparent ICMetric system provides advantages in the detection of tampering. The generated ICMetric key for the Docker containers can be used for providing other security functionalities in the Docker environment.

6.4 SECURITY ANALYSIS

Specifically, ICMetric possess the following significant potential to be used for identification of the Docker containers:

- ICMetric of the Docker container helped identify the container through direct generation of ICMetric from the internal behavioral and operating characteristics of Docker containers. Therefore, no keys or characteristic templates were stored as the ICMetric is regenerated when required; thereby preventing the possibility of key capture used for container security. Also, the compromise of a system does not release sensitive ICMetric data, which would allow unauthorized access to the container.
- The ICMetric technology is able to perform identification and detect malware based tampering of the Docker container, since introduction of any malware causes the ICMetric key to vary. Tampering with the constitution of the Docker container causes its behavior to change, potentially causing the features underlying the ICMetric to change, perhaps dramatically, thus causing the generated ICMetric to change. Consequently, a maliciously tampered Docker container will be autonomously prevented from being accessible, as the regenerated keys will differ from those created before its integrity was compromised. Therefore, ICMetric security in Docker containers will fail securely and provide a very high immunity from tampering in Docker containers.
- The ICMetric technology delivers another major advantage, as the security provided by the system may be implemented transparently to the user. That is, the proposed validation system generates an ICMetric key from the behavioral

characteristics of the Docker container, without any action being required from the user.

6.5 SUMMARY

This work showed the feasibility of the ICMetric system being used in the identification of Docker containers on the cloud, thereby establishing the authenticity of the container in question. Namespace and cgroup features can potentially be used to generate ICMetric key for Docker container identification. An extensive analysis of Docker container features led to the success of establishing features for ICMetric identification in Docker containers. The strength of the ICMetric technology lies in the formulation of a system that is comprised of diverse and robust features. The results of the various feature tests conducted, exhibited a wide range of potential features, with many of the tested features having produced very promising results. The use of feature correlations can further strengthen these results as extra values with higher levels of obfuscation were derived from these comparisons.

After carrying out the analysis and testing of features, the ability to use the ICMetric system in a practical environment test bed looked highly promising. A major challenge was the appropriate selection of features, that can work together consistently in a single system to generate a stable key. In conclusion, this case study has shown the potential of the technology to generate an ICMetric for Docker container identification.

7

Conclusion and Future Directions

Computation systems have already transitioned from standalone devices to highly capable networked systems that share data and information. As users increasingly rely on networked devices, the need for security cannot be denied. A comprehensive security solution should provide at least a set of basic services like confidentiality, authentication and non-repudiation. A security implementation that provides insufficient security or poorly implemented security provides a false assurance of security. Some systems being manufactured today possess design flaws or have limited to no security provision [116]. Adversaries can exploit these design flaws to gain illegitimate access to a system. This thesis is an effort to improve the security of end-to-end communication using the ICMetric technology. The proposed schemes aim to provide confidentiality, authentication and integrity using the ICMetric technology as a novel root of trust. This chapter presents a summary of contributions made to improve the security of systems in the end-to-end environment.

7.1 SUMMARY OF THESIS

Availability of high speed communications has resulted in the creation of environments that facilitate information and resource sharing. Users are increasingly using computation systems for banking, e-commerce, healthcare, education, communication, etc. With such diverse applications, the importance of security cannot be denied. Conventional cryptographic systems have relied on stored keys to provide security services. A cryptographic algorithm is made public while steps are taken to ensure that the keys are kept secret. The sharing of the cryptographic algorithm does not expose a system to the adversary as the algorithms are designed using mathematical or algorithmic intractability. Although increasing key sizes has been successful in deterring pre-computed attacks [117], it has also forced adversaries to look for other methods to attack the cryptosystems. Adversaries now attempt to exploit weaknesses in design or implementation to capture a system. This research has first shown that there are numerous methods by which an adversary can capture a system. What is interesting is that the adversaries can use side channel attacks [7][38] which are highly effective and do not require a lot of time to execute. To counter side channel attacks cryptographers are now implementing hardware entangled cryptographic schemes by using Physically Unclonable Functions [118]. Chapter 2 presents detailed literature pertaining to common attacks and their remedy by incorporating an alternative root of trust like Physically Unclonable Functions. This chapter presents a detailed discussion on the novel ICMetric technology as a basis for cryptographic services. The ICMetric technology uses the features of a device to create an identification called the device ICMetric. The ICMetric technology has two purposes i.e. as a key theft deterrent and a root of trust for cryptographic key generation. The features used for ICMetric

generation play an important role in the security of the technology, therefore a study on possible features has also been presented. The purpose of incorporating the ICMetric technology into secure communications is to reduce problems associated with key theft by eliminating the need for stored keys. The cryptographic keys are generated only when required and discarded after use. This means that the keys are never stored on the system.

The ICMetric technology is used as an alternative to stored keys, therefore schemes have been presented that demonstrate how key can be generated using the ICMetric of a device. However, the generated ICMetric keys could be weak [15], hence chapter 3 presents a two-tier scheme for the generation of strong ICMetric based symmetric keys. The proposed scheme aims to safeguard the actual ICMetric as well as the generated ICMetric keys from pre-computed attacks.

Many cryptographic algorithms use the symmetric keys for the provision of security. Chapter 4 has established that symmetric keys can be generated by using the ICMetric of a device, without transmitting the ICMetric over the channel. The proposed scheme has been simulated and results show that the ICMetric technology can be used to generate symmetric keys without excessive resource demand. The chapter has also used AES to show the effectiveness and successful working of the ICMetric based symmetric keys.

Asymmetric keys provide a powerful way of providing security owing to the way in which the keys are held by the communicating parties. Chapter 5 has demonstrated that the ICMetric of a device can be used for creating asymmetric keys. Perhaps the greatest advantage of this scheme is that it provides non repudiation services as the holder of a private key cannot deny that the encrypted message was sent

from his device. The proposed scheme 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.

Chapter 6 is a practical case study which addresses to solve a major problem cloud service providers face with hosting its cloud services in Docker containers. Evaluation and resilience in the presence of malware on a Docker container is a novel concept that has effectively been managed by ICMetric. The chapter investigates enabling a secure Docker environment via the ICMetric technology, thereby detecting change in software constitution of the containers due to the introduction of malicious or abnormal behaviour. The ICMetric technology creates a runtime profile of the running container, to detect change in the runtime behaviour of the Docker.

7.2 LIST OF PUBLICATIONS

In the following section the publication activity carried out during this PhD are enlisted:

[1] H. Tahir, R. Tahir, K. McDonald-Maier, "On the Security of Consumer Wearable Devices in the Internet of Things", PLOS One.

[2] R. Tahir, H. Tahir, A. Sajjad, K. McDonald-Maier, “ICMApen: An ICMetric Based Security Framework for Sleep Apnea Monitoring”, ISI International Journal of Advanced and Applied Sciences, August 2017.

[3] R. Tahir, H. Tahir, A. Sajjad, K. McDonald-Maier, "A Secure Cloud Framework for ICMetric Based IoT Health Devices", 2nd International Conference on Internet

of Things, Data and Cloud Computing (ICC 2017), Cambridge, UK, 22-23 March, 2017.

[4] R. Tahir, H. Tahir, K. McDonald-Maier, A. Fernando, "A Novel ICMetric Based Framework for Securing the Internet of Things", IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 7-11 January 2016, pp. 469-470.

[5] R. Tahir, H. Tahir, K. McDonald-Maier, "Securing Health Sensing Using Integrated Circuit Metric", MDPI Sensors Journal 2015, October 2015, 15(10), pp 26621-26642.

[6] H. Tahir, R. Tahir, K. McDonald-Maier, "Securing MEMS Based Sensor Nodes in the Internet of Things", 6th International Conference on Emerging Security Technologies (EST), Technische Universitaet Braunschweig, Germany, 3-5 September, 2015.

[7] H. Tahir, R. Tahir, K. McDonald-Maier, "A Group Secure Key Generation and Transfer Protocol Based on ICMetrics", 9th IEEE International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP 14), Manchester, UK, 23-25 July, 2014. pp 733-738.

[8] H. Tahir, R. Tahir, K. McDonald-Maier, "A Novel Private Cloud Document Archival System Based on ICMetrics", 4th International conference on Emerging Security Technologies (EST), Cambridge, 09-11 Sept, 2013. pp 102-106.

[9] R. Tahir, H. Hu, D. Gu, G. Howells, K. McDonald-Maier, "A Scheme for the Generation of Strong ICMetrics Session Key Pairs for Secure Embedded System Applications", 7th IEEE Symposium on Security and Multimodality in Pervasive Environment, Barcelona, Spain, March 26-29, 2013.

[10] R. Tahir, H. Hu, D. Gu, G. Howells, K. McDonald-Maier, "Resilience against Brute Force and Rainbow Table Attacks using Strong ICMetrics Session Key Pairs", 1st IEEE International Conference on Communications, Signal Processing, and their Applications (ICCSPA'13), Sharjah, UAE, Feb 12-14, 2013.

[11] R. Tahir, H. Hu, D. Gu, G. Howells, K. McDonald-Maier, "A Scheme for the Generation of Strong Cryptographic Key Pairs based on ICMetrics", 7th IEEE International Conference on Internet Technology and Secured Transactions (ICITST'12), London, UK, Dec 12-14, 2012.

[12] Tahir, R., Maier, K. M., "Improving Resilience against Node Capture Attacks in Wireless Sensor Networks using ICMetrics", 3rd International Conference on Emerging Security Technologies, September 5-7, 2012, Lisbon, Portugal.

[13] Tahir, R., Maier, K. M., "An ICMetrics based Lightweight Security Architecture using Lattice Signcryption", 3rd International Conference on Emerging Security Technologies, September 5-7, 2012, Lisbon, Portugal.

7.3 FUTURE DIRECTIONS

The ICMetric technology has been designed as a strong design basis upon which further systems can be built. There are many research areas which could benefit from the advantages the ICMetric technology has to offer. This section provides some possible research directions based on the ICMetric technology.

Previously the ICMetric technology has not been studied in combination with high speed industrial networks like Profinet [119]. The purpose of Profinet is to provide a stable high-speed network where I/O devices (sensors, monitoring systems, cameras, etc.) can send and receive large amounts of data under tight time constraints (1ms). The

ICMetric technology can be used to look for unique features in the I/O devices, thereby improving security of the devices and the network.

As self-driving vehicles become increasingly common it is important to ensure that the vehicles and their occupants are secure. A typical autonomous vehicle is equipped with many sensors and components which could be compromised by adversaries remotely. The ICMetric technology can be used to prevent adversaries from compromising an embedded component by ensuring stringent security methods.

Cryptocurrencies [120] are fundamentally digital assets that are exchanged like conventional currencies. Since cryptocurrencies do not physically exist therefore all transactions take place with the help of cryptography. The ICMetric technology can be incorporated into cryptocurrency implementation to provide improved authentication, confidentiality and integrity. It would be particularly interesting to see how the security of blockchains can be improved with ICMetric. This work can also be extended to the permissioned blockchains where problem exists in verification and integrity of transactions found in many domains; ranging from financial transactions to infrastructure such as telecommunications networks, healthcare data and access governance [120]. This goal can be achieved by combining ICMetric device authentication and blockchains.

The ICMetric technology could have a profound impact in Searchable Encryption [121][122]. Searchable Encryption is a technique that has evolved with the use of Cloud to store large amounts of data. To preserve the privacy of the documents, the documents need to be encrypted prior to being outsourced to the cloud. Searching over the encrypted documents was not possible before Searchable Encryption was introduced. Searchable encryption allows a client to generate search queries and

delegate the search to the Cloud. One of the key features of Searchable Encryption are that only an authorised person should be able to perform the search and generate meaningful trapdoor. With the use of ICMetric coupled with Searchable Encryption, neither the key nor the documents need to be stored on the client side which helps resist the key theft and data theft.

An emerging area of research which will greatly impact our daily lives is the Internet of Things (IoT) [123]. There are many forms of devices in IoT. Some devices will possess adequate resources while others will be limited in their capabilities. Devices in the IoT make many promises but there are hurdles [124] in their adoption. One major hurdle in the wide adoption of the IoT are security concerns. The ICMetric technology can mitigate many of the security concerns that plague devices in the IoT.

REFERENCES

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

[2] Y. Dodis, "The Cost of Cryptography," 2013. [Online]. Available: http://nautil.us/issue/7/waste/the-cost-of-cryptography. [Accessed: 02-Jul-2017].

[3] C. Graham, "NHS cyber attack: Everything you need to know about 'biggest ransomware' offensive in history," *The Telegraph*, 13-May-2017.

[4] L. J. Camp and M. E. Johnson, *The Economics of Financial and Medical Identity Theft*. Boston, MA: Springer US, 2012.

[5] W. Rankl and W. Effing, *Smart Card Handbook*. Wiley, 2000.

[6] M. R. Garey and D. S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness*, 1st edition. W.H. Freeman, 1979.

[7] K. Schramm, K. Lemke, and C. Paar, "Embedded Cryptography: Side Channel Attacks," in *Embedded Security in Cars*, Berlin/Heidelberg: Springer-Verlag, 2006, pp. 187–206.

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

[9] D. Genkin, L. Pachmanov, I. Pipman, A. Shamir, and E. Tromer, “Physical key extraction attacks on PCs,” *Commun. ACM*, vol. 59, no. 6, pp. 70–79, Jun. 2016.

[10] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten, “Lest We Remember: Cold-Boot Attacks on Encryption Keys,” *Communications of the ACM*, vol. 52, no. 5, ACM, p. 91, 01-May-2009.

[11] I. Kizhatov, “Physical Security of Cryptographic Algorithm Implementations,” Universite Du Luxembourg, 2011.

[12] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, “Lightweight Cryptography for Embedded Systems – A Comparative Analysis,” in *8th International Workshop on Data Privacy Management and Autonomous Spontaneous Security*, 2013, pp. 333–349.

[13] S. H. Weingart, “Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defenses,” Springer Berlin Heidelberg, 2000, pp. 302–317.

[14] S. Tahir and I. Rashid, “ICMetric-Based Secure Communication,” in *Innovative Solutions for Access Control Management*, vol. 36, IGI Global, 2016, pp. 263–293.

[15] R. Tahir, H. Huosheng Hu, D. Dongbing Gu, K. McDonald-Maier, and 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)*, 2013,

pp. 1–6.

[16] D. Merli and R. Plaga, “Physical unclonable functions: devices for cryptostorage,” in *Proceedings of the 3rd international workshop on Trustworthy embedded devices - TrustED ’13*, 2013, pp. 1–2.

[17] T. Wu, “The Secure Remote Password Protocol,” in *Proceedings of the Symposium on Network and Distributed Systems Security NDSS 98*, 1998, vol. 4, pp. 97–111.

[18] T. Perrin, T. Wu, N. Mavrogiannopoulos, and D. Taylor, “Using the Secure Remote Password (SRP) Protocol for TLS Authentication,” 2007.

[19] P. Koopman, “Embedded system security,” *Computer (Long. Beach. Calif).*, vol. 37, no. 7, pp. 95–97, Jul. 2004.

[20] D. Papp, Z. Ma, and L. Buttyan, “Embedded systems security: Threats, vulnerabilities, and attack taxonomy,” in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, 2015, pp. 145–152.

[21] T. Noergaard, *Embedded systems architecture : a comprehensive guide for engineers and programmers*, Second Edition. Newnes, 2013.

[22] P. Marwedel, *Embedded system design : embedded systems foundations of cyber-physical systems*, 2nd ed. Springer, 2011.

[23] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, “Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT),” in *International Conference on Network Security and Applications*, 2010, pp. 420–429.

[24] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded*

Applications, 2nd ed. Springer Publishing Company, 2011.

[25] R. Zurawski, "Networked embedded systems in industrial automation," in *6th IEEE International Conference on Industrial Informatics*, 2008, pp. 3–8.

[26] K. Qian, D. den. Haring, and L. Cao, *Embedded software development with C*. Springer, 2009.

[27] K. R. Fowler and C. L. Silver, *Developing and managing embedded systems and products : methods, techniques, tools, processes, and teamwork*. Newnes, 2015.

[28] C. H. Gebotys, *Security in Embedded Devices*. Springer, 2010.

[29] S. Parameswaran and T. Wolf, "Embedded systems security—an overview," *Des. Autom. Embed. Syst.*, vol. 12, no. 3, pp. 173–183, Sep. 2008.

[30] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and Apporaches for Distributed Sensor Network Security," 2000.

[31] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "Analyzing the energy consumption of security protocols," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.*, 2003, pp. 30–35.

[32] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wirel. Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[33] B. Schneier, "The Internet of Things Is Wildly Insecure — And Often Unpatchable," *Wired*, Jun-2014.

[34] I. Zeifman, D. Bekerman, and B. Herzberg, “Breaking Down Mirai: An IoT DDoS Botnet Analysis,” *Imperva incapsula Blog*, 2016. [Online]. Available: https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html.

[35] S. Yu, *Distributed denial of service attack and defense*. Springer-Verlag, 2014.

[36] J. Leyden, “Strange Mirai botnet brew blamed for powerful application layer attack,” *The Register*, 2017. [Online]. Available: https://www.theregister.co.uk/2017/03/29/mirai_variant/.

[37] N. Dhanjani, *Abusing the Internet of things: blackouts, freakouts, and stakeouts*. O’Reilly, 2015.

[38] W. Hnath and J. Pettengill, “Differential Power Analysis Side-Channel Attacks in Cryptography,” Worcester Polytechnic Institute, 2010.

[39] P. C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in *Advances in Cryptology — CRYPTO ’96*, Springer, Berlin, Heidelberg, 1996, pp. 104–113.

[40] J. R. Vacca, *Network and system security*. Syngress, 2014.

[41] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, “Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation,” 2015.

[42] M. J. Atallah, E. D. Bryant, J. T. Korb, and J. R. Rice, “Binding software to specific native hardware in a VM environment,” in *Proceedings of the 1st ACM workshop on Virtual machine security - VMSec ’08*, 2008, p. 45.

[43] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical one-way

functions.," *Science (80-.).*, vol. 297, no. 5589, pp. 2026–30, Sep. 2002.

[44] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of PUF-based 'unclonable' RFID ICs for anti-counterfeiting and security applications," in *2008 IEEE International Conference on RFID (Frequency Identification), IEEE RFID 2008*, 2008, pp. 58–64.

[45] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual conference on Design automation - DAC '07*, 2007, pp. 9–14.

[46] M. Cortez, A. Dargar, S. Hamdioui, and G.-J. Schrijen, "Modeling SRAM start-up behavior for Physical Unclonable Functions," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2012, pp. 1–6.

[47] G.-J. Schrijen and 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*, 2012, pp. 1319–1324.

[48] Intrinsic-ID Inc, "SRAM PUF: The secure silicon fingerprint," 2016.

[49] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-Proof Hardware from Protective Coatings," in *Proceedings of the 8th International Conference on Cryptographic Hardware and Embedded Systems*, 2006, pp. 369–383.

[50] E. Papoutsis, G. Howells, A. Hopkins, and K. McDonald-Maier, "Integrating Feature Values for Key Generation in an ICmetric System," in *2009 NASA/ESA*

Conference on Adaptive Hardware and Systems, 2009, pp. 82–88.

[51] E. Papoutsis, "Investigation of the Potential of Generating Encryption Keys for ICMETRICS," University of Kent, 2009.

[52] A. N. Kataria, D. M. Adhyaru, A. K. Sharma, and T. H. Zaveri, "A survey of automated biometric authentication techniques," in *2013 Nirma University International Conference on Engineering (NUiCONE)*, 2013, pp. 1–6.

[53] X. Zhai, K. Appiah, S. Ehsan, H. Hu, D. Gu, K. McDonald-Maier, W. M. Cheung, and G. Howells, "Application of ICmetrics for Embedded System Security," in *2013 Fourth International Conference on Emerging Security Technologies*, 2013, pp. 89–92.

[54] Y. Kovalchuk, K. McDonald-Maier, and G. Howells, "Overview of ICmetrics Technology – Security Infrastructure for Autonomous and Intelligent Healthcare System," *Int. J. u- e- Serv. Sci. Technol.*, vol. 4, no. 3, pp. 49–60, 2011.

[55] Y. Kovalchuk, H. Hu, D. Gu, K. McDonald-Maier, D. Newman, S. Kelly, and G. Howells, "Investigation of Properties of ICmetrics Features," in *2012 Third International Conference on Emerging Security Technologies*, 2012, pp. 115–120.

[56] R. Tahir, H. Tahir, and K. McDonald-Maier, "Securing health sensing using integrated circuit metric," *Sensors (Switzerland)*, vol. 15, no. 10, pp. 26621–26642, 2015.

[57] Maxim Integrated Products, "DS2411 Silicon Serial Number with VCC Input," Sunnyvale, 2011.

[58] C. Martin, A. Kadry, and G. Abu-Shady, “Quantifying the financial impact of it security breaches on business processes,” in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, 2014, pp. 149–155.

[59] O. Kara and A. Atalay, “Preimages of hash functions through rainbow tables,” in *2009 24th International Symposium on Computer and Information Sciences*, 2009, pp. 304–309.

[60] K. B.Adedeji and J. O. Famoriji, “Investigating the Effects of varying the Key Size on the Performance of AES Algorithm for Encryption of Data over a Communication Channel,” *Int. J. Appl. Inf. Syst.*, vol. 7, no. 8, pp. 6–10, Sep. 2014.

[61] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[62] E. T. Jaynes and G. L. Bretthorst, *Probability Theory : The Logic of Science*. Cambridge University Press, 2003.

[63] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M. Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST Special Publication, 2010.

[64] G. Marsaglia, “Marsaglia’s Random Number CDROM Files including the DIEHARD Battery of Tests of Randomness.” Florida State University, 1995.

[65] *Security Requirements for Cryptographic Modules*. NIST, 2001.

[66] R. W. Hamming, *Coding and information theory*. Prentice-Hall, 1980.

[67] D. E. Knuth, *The art of computer programming : Seminumerical algorithms*, 3rd ed. Boston: Addison-Wesley Pub. Co, 1997.

[68] W. J. Morokoff and R. E. Caflisch, "Quasi-Monte Carlo Integration," *J. Comput. Phys.*, vol. 122, no. 2, pp. 218–230, Dec. 1995.

[69] A. K. Lenstra and R. V. Eric, "Selecting Cryptographic Key Sizes," *J. Cryptol.*, vol. 14, pp. 255–293, 2001.

[70] N. P. Smart, V. Rijmen, B. Warinschi, and G. Watson, "Algorithms, Key Sizes and Parameters Report," 2013.

[71] G. Sowmya, D. Jamuna, and M. V. K. Reddy, "Blocking of Brute Force Attack," *Int. J. Eng. Res. Technol.*, vol. 1, no. 6, 2012.

[72] M. Curtin, *Brute Force: Cracking the Data Encryption Standard*, 1st ed. Springer, 2007.

[73] J. Katz and Y. Lindell, *Introduction to modern cryptography*, 2nd ed. Chapman & Hall/ CRC, 2014.

[74] M. S. Turan, E. Barker, W. Burr, and L. Chen, *Recommendation for Password-Based Key Derivation - Part 1: Storage Applications*, no. December. 2010, p. 14.

[75] D. Eastlake and P. Jones, *Secure Hash Algorithm 1 (SHA1)*. 2001, p. IETF.

[76] *Secure Hash Standard (SHS)*. National Institute of Standards and Technology, 2015.

[77] "MD5 Collisions - The Effect on Computer Forensics," 2006.

[78] X. Wang and H. Yu, “How to Break MD5 and Other Hash Functions,” in *EUROCRYPT 2005: Advances on Cryptology*, 2005, pp. 19–35.

[79] X. Wang, Y. L. Yin, and H. Yu, “Finding Collisions in the Full SHA-1,” Springer, Berlin, Heidelberg, 2005, pp. 17–36.

[80] R. K. Meyers and A. H. Desoky, “An Implementation of the Blowfish Cryptosystem,” in *2008 IEEE International Symposium on Signal Processing and Information Technology*, 2008, pp. 346–351.

[81] M. Bishop, *Computer Security: Art and Science*, 1st ed. Addison-Wesley, 2015.

[82] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification*. 2000.

[83] NIST, “Digital Identity Guidelines: Authentication and Lifecycle Management,” 2017.

[84] J. Viega, M. Messier, and P. Chandra, *Network security with OpenSSL*. O’Reilly, 2002.

[85] “Valgrind Documentation,” 2016.

[86] M. Wolff, “Massif-Visualizer Memory Profiling UI,” 2011.

[87] D. Hook, *Beginning Cryptography with Java*. Birmingham: Wrox Press Ltd, 2005.

[88] Z. Gutterman, B. Pinkas, and T. Reinman, “Analysis of the Linux random number generator,” in *2006 IEEE Symposium on Security and Privacy*, 2006, p. 15 pp.-pp.385.

[89] N. Datta, “Zero Knowledge Password Authentication Protocol,” in *New Paradigms in Internet Computing*, Springer, Berlin, Heidelberg, 2013, pp. 71–79.

[90] W. Huqing and S. Zhixin, “Research on Zero-Knowledge Proof Protocol,” *IJCSI Int. J. Comput. Sci. Issues*, vol. 1, no. 1, pp. 194–200, 2013.

[91] Jiang Huiping, “Strong password authentication protocols,” in *2010 4th International Conference on Distance Learning and Education*, 2010.

[92] H. Lee, D. Won, K. Sohn, and H. Yang, “The Efficient 3-Pass Password-Based Key Exchange Protocol with Low Computational Cost for Client,” in *Proceedings of the Second International Conference on Information Security and Cryptology*, 2000, pp. 147–155.

[93] F. Kiefer and M. Manulis, “Zero-Knowledge Password Policy Checks and Verifier-Based PAKE,” in *19th European Symposium on Research in Computer Security - Volume 8713*, 2014, pp. 295–312.

[94] S. M. Bellovin and M. Merritt, “Encrypted key exchange: password-based protocols secure against dictionary attacks,” in *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992, pp. 72–84.

[95] D. P. Jablon, “Strong password-only authenticated key exchange,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 5, pp. 5–26, Oct. 1996.

[96] S. M. Bellovin and M. Merritt, “Augmented encrypted key exchange : a password-based protocol secure against dictionary attacks and password file compromise,” in *Proceedings of the 1st ACM conference on Computer and*

communications security - CCS '93, 1993, pp. 244–250.

[97] X. Yi, F.-Y. Rao, Z. Tari, F. Hao, E. Bertino, I. Khalil, and A. Y. Zomaya, "ID2S Password-Authenticated Key Exchange Protocols," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3687–3701, 2016.

[98] B. A. Forouzan and D. Mukhopadhyay, *Cryptography and Network Security E/2 Book Online at Low Prices in India | Cryptography and Network Security*. McGraw Hill Education, 2010.

[99] T. Jamil, "The Rijndael algorithm," *IEEE Potentials*, vol. 23, no. 2, pp. 36–38, Apr. 2004.

[100] G. M. Stevens and C. Doyle, *Privacy : wiretapping and electronic eavesdropping*. Nova Biomedical, 2002.

[101] WolfSSL, *CyaSSL User Manual*. 2014.

[102] S. Gueron, "Intel Advanced Encryption Standard (AES) New Instructions Set," 2012.

[103] F. R. Beyette, G. J. Kost, C. A. Gaydos, and B. H. Weigl, "Point-of-Care Technologies for Health Care," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 3, pp. 732–735, Mar. 2011.

[104] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[105] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

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

[107] M. Fukumitsu, S. Hasegawa, S. Isobe, and H. Shizuya, "On the Impossibility of Proving Security of Strong-RSA Signatures via the RSA Assumption," in *Australasian Conference on Information Security and Privacy*, 2014, pp. 290–305.

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

[109] I. Ristic, *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, 2014.

[110] J. A. Davies, *Implementing SSL/TLS using Cryptography and PKI*. Wiley, 2011.

[111] M. O. Rabin, "Probabilistic algorithm for testing primality," *J. Number Theory*, vol. 12, no. 1, pp. 128–138, Feb. 1980.

[112] C. Negus, *Docker Containers*, 1st ed. Prentice Hall, 2015.

[113] P. Raj, J. S. Chelladhurai, V. Singh, S. Holla, and O. Hane, *Docker: Creating Structured Containers*. Packt Publishing, 2016.

[114] M. Tso, "Multivariate Normal Distribution," *Lecture Notes*, 2008. [Online]. Available: http://www.maths.manchester.ac.uk/~mkt/MT3732 (MVA)/Notes/MVA_Section3.pdf. [Accessed: 10-Jul-2017].

[115] C. B. Do, "The Multivariate Gaussian Distribution," *Lecture Notes*, 2008. [Online]. Available: http://cs229.stanford.edu/section/gaussians.pdf.

[116] A. K. Sood and R. J. Enbody, “Crimeware-as-a-service—A survey of commoditized crimeware in the underground market,” *Int. J. Crit. Infrastruct. Prot.*, vol. 6, no. 1, pp. 28–38, 2013.

[117] Y. Xiao and Y. Pan, *Security in Distributed and Networking Systems*. WORLD SCIENTIFIC, 2007.

[118] R. Maes, “Physically Unclonable Functions: Constructions, Properties and Applications,” Katholieke Universiteit Leuven, 2012.

[119] R. Pigan and M. Metter, *Automating with PROFINET : Industrial Communication Based on Industrial Ethernet*. Wiley, 2015.

[120] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies : A Comprehensive Introduction*. Princeton University Press, 2016.

[121] Q. Wang, M. He, M. Du, S. S. M. Chow, R. W. F. Lai, and Q. Zou, “Searchable Encryption over Feature-Rich Data,” *IEEE Trans. Dependable Secur. Comput.*, vol. 99, pp. 1–1, 2016.

[122] S. Tahir, M. Rajarajan, and A. Sajjad, “A ranked searchable encryption scheme for encrypted data hosted on the Public Cloud,” in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 242–247.

[123] A. McEwen and H. Cassimally, *Designing the Internet of Things*. Wiley, 2013.

[124] L. Piwek, D. A. Ellis, S. Andrews, and A. Joinson, “The Rise of Consumer Health Wearables: Promises and Barriers,” *PLoS Med.*, vol. 13, no. 2, pp. 1–9, 2016.