A Scheme for the Generation of Strong ICMetrics based Session Key Pairs for Secure Embedded System Applications

Ruhma Tahir, Huosheng Hu, Dongbing Gu, Klaus McDonald-Maier
School of Computer Science and Electronic Engineering
University of Essex
Colchester, United Kingdom
rtahir@essex.ac.uk, hhu@essex.ac.uk, dgu@essex.ac.uk, kdm@essex.ac.uk

Gareth Howells
School of Engineering and Digital Arts
University of Kent
Canterbury, United Kingdom
W.G.J.Howells@kent.ac.uk

Abstract— **This paper presents a scheme for the generation of strong session based ICMetrics key pairs for security critical embedded system applications. ICMetrics generates the security attributes of the sensor node based on measurable hardware and software characteristics of the integrated circuit. In the proposed scheme a random session ID is assigned by a trusted party to each participating network entity, which remains valid for a communication session. Our work is based on the design of a key derivation function that uses an ICMetrics secret key and a session token assigned by the trusted party to derive strong cryptographic key pairs for each entity. These session tokens also serve the purpose of identification/authentication between the trusted parties and the respective nodes in each network. The main strength of our proposed scheme rests on the randomness feature incorporated via the random session ID's, which makes the generated strong private/public key pair highly resistant against exhaustive search and rainbow table attacks. Our proposed approach makes use of key stretching using random session tokens via SHA-2 and performs multiple itertions of the proposed key derivation function to generate strong high entropy session key pairs of sufficient length. The randomness of the assigned ID's and the session based communication hinders the attacker's ability to launch various sorts of cryptanalytic attacks, thereby making the generated keys very high in entropy. The randomness feature has also been very carefully tuned according to the construction principles of ICMetrics, so that it doesn't affect the original ICMetrics data. The second part of the proposed scheme generates a corresponding public session key by computing the Hermite Normal Form of the high entropy private session key.**

Keywords- ICMetrics (Integrated Circuit Metrics), rainbow table attacks, brute force attacks, Hermite Normal Form (HNF), key derivation function, session keys, key stretching.

I. INTRODUCTION

Increasing number of embedded systems with increasing complexity are present in all aspects of our lives; with their use in single user applications to large scale processes. Embedded system devices are often networked via wireless communication links to perform useful tasks [1-2]. However the wireless nature of the communication channel between the embedded devices makes them vulnerable to attacks by adversaries [3-4]. Therefore embedded systems security is a key aspect of embedded systems design and is currently a major area of research. ICMetrics (Integrated Circuit metrics) is a key technology that has been developed in connection with the requirement of security in embedded system applications [8]. ICMetrics makes use of system level characteristics to provide identification to the system [9]. It generates the secret key from measurable hardware/ software characteristics and specifications of the device. The stable ICMetrics key that is derived from hardware and software characteristics of a device provide a very strong link between the device and the cryptographic key; thereby enabling device verification with a very high degree of assurance [8].

A number in its raw form cannot serve as a useful key for cryptographic operations; since the raw keying material might be too short or have low entropy, which might lead it to be easily guessed by an attacker [16]. Therefore the generated cryptographic keys must possess certain properties, i.e. sufficient key entropy and length, before they can qualify to be kept as a key for secure cryptographic operations. The goal of our proposed scheme is to propose a key generation mechanism based on an ICMetrics basis number that generates high entropy public/private key pair having sufficient levels of security. This paper presents an SHA-2 based key derivation method for the generation of strong session based public/private key pairs from ICMetric data. As mentioned above, the design of the key derivation function forming a part of our scheme is based on an ICMetrics secret key and a random session token. The random session tokens for each entity are generated by trusted parties corresponding to their particular networks. These session tokens are based on random ID's that are assigned by trusted party higher up in the hierarchy from the requesting node. The effect of a random session token is to create a large set of possible keys associated with a particular raw ICMetrics key and the iterations lead to a significant increase in number of rounds performed by the attacker to derive the key. These session key pairs are generated in every session and are valid for a single

The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council under grant EP/K004638/1 and the EU Interreg IV A 2 Mers Seas Zeeën Cross-border Cooperation Programme – SYSIASS project: Autonomous and Intelligent Healthcare System (project's website http://www.sysiass.eu/).

communication session. The proposed session based strong ICMetrics public/private key pair generation mechanism has the following features that address the keying issues in secure embedded system applications:

1. To safeguard against issues related to key compromise, the proposed architecture is based on the use of ICMetrics values for key pair generation [17]. ICMetrics generates keys based on the hardware/software characteristics and specification of the device which provides an effective means to address the issues related to the secrecy of key. Our scheme intends to bind a cryptographic key with the device's ICMetrics information.
2. As stated above, each node in the network (entity/trusted party) is assigned a random ID that remains valid during a communication session and a completely new random ID is assigned for subsequent communication sessions. The trusted party makes use of this random session ID to generate a session token that helps safeguard against various sorts of cryptanalytic attacks. This random session token also serves to identify/ authenticate the participating entities.
3. In our scheme the high entropy session key pairs are formed by combining the ICMetrics generated basis number with the session token generated by the Trusted Party. Both of these secret values are combined through a SHA-2 based key derivation function.
4. The main challenge with the ICMetrics generated basis number is the entropy and length of the generated secret value. So our proposed design generates a high entropy key of sufficient length using SHA-2 based key derivation function with the session token and ICMetrics basis number as input. We iterate through multiple rounds of the SHA-2 based hash function to stretch the secret value to the required length, thereby also generating a key with high entropy that is resistant against brute force and rainbow table attacks.
5. Lastly, our scheme generates a public key corresponding to the generated high entropy private key by computing the Hermite Normal Form of the private key [6]. The Hermite Normal Form is particularly suitable for public key generation since its unique, non-reversible and doesn't require any random values for operations.

The remainder of this paper is organized as follows; in section 2 we discuss characteristics of a strong cryptographic key and how weak keys pose a threat to security of applications. Section 3 discusses the ICMetrics technology and its features elaborating on how ICMetrics could be a viable solution, but at the same time the weak secret key generated by ICMetrics feature values could pose a huge threat to security of the application. Section 4 introduces the security primitives on which the design of our proposed architecture rests. Section 5 explains the network topology for our proposed design. The design of our proposed framework for the generation of strong high entropy keys is presented in Section 6. Section 7 presents the analysis of its security. The conclusion and future work is presented in section 8.

II. Strong Cryptographic Keys and ICMetrics

A. Strong Cryptographic Keys

The security of sensitive applications mainly depends on the proper generation and protection of its cryptographic keys, since the underlying encryption/decryption algorithms are usually published and globally known. Therefore there are inherent risks if proper keying materials and key management are not used for secure operations; this can result in compromise of encryption keys that will seriously affect the integrity, confidentiality, and availability of communications. Without appropriate generation and handling of keys, they could be easily guessed, modified, or substituted by unauthorized personnel who could then intercept sensitive communications. Also, since embedded systems have typically limitations in terms of power, memory and processing power [1]; there is a need to make a sensible choice of cryptographic mechanisms for key generation, and to keep the length of key sizes according to individual applications [2], so that the employed solutions prove to be suitable for embedded system applications.

As stated above, the generated cryptographic keys must also have certain properties to qualify as a key to be used for secure operations, since any number in its raw form cannot serve as a key for cryptographic operations. A longer cryptographic key is more secure, since it makes it harder for the adversary to break the encrypted text [16]. Weak and low entropy keys are likely to being easily cracked by an adversary, therefore it is essential to have keys with high entropy, thereby making key secure against attacks by an adversary during their lifecycle [19]. Although strong cryptographic functions provide data privacy and protection, without secure keys the integrity and confidentiality of an application is still at risk.

A major threat in security applications having weak / low entropy keys is the launch of a brute force / exhaustive search attack [15]. A brute force attack is a very common methodology adopted by adversaries to break cryptosystems with strong cryptographic operations but weak keys; whereby instead of finding weaknesses in the encryption system, the attacker tries to crack the cryptographic key used for performing the cryptographic operations. The attacker tries each possible key combination to find the correct key that has been employed for carrying out the cryptographic operations. From an attacker's 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 [19].

B. ICMetrics

Traditionally cryptographic algorithms have always relied on the use of stored keys for functioning of the network. However the use of stored keys to enable secure communication is threatened by the fact that, if the key used to encrypt the data is compromised, it will result in loss of data that is encrypted using the compromised key.

ICMetrics (Integrated Circuit Metrics) makes use of system level characteristics to provide identification to the system [12]. It generates encryption keys from measurable properties of a given hardware device such as its hardware/software characteristics and the specification of the node; similar to the way biometrics extracts human features. ICMetrics computes the required metrics on those hardware and software characteristics that are difficult for the attacker to deduce [11]. These metrics / features are not static but vary in a depending on the system and its interaction with its environment. For example, the address and value from the data transactions of a processor; its program address; and metrics for the effectiveness of the program and data caches derived from performance counters, etc [12].

After each ICMetric key generation stage the produced ICMetric key [13] is temporary and exists only locally, and the reproduction of the ICMetric key once again takes place from measurable characteristics of the integrated circuit [10]. The ICMetric key is generated each time it is requested for identification or encryption functions. It consists of measuring the system features, applying the normalisation maps and combining the various feature values to generate a basis number for identifying the system. The individual feature values are be combined using two possible feature combination techniques; each generating a different sized key of varying stability [8]. The feature addition-combination technique generates a stable but small basis number by adding individual feature values, while the feature concatenation-combination technique generates a long but less stable basis number by concatenating individual feature values. The resultant basis number is used for the subsequent derivation of the key required for the actual encryption process. Modifications to either the software executing on a given device, or to its hardware, will cause variations of the feature measurable characteristics and therefore the derived basis value. This in turn will mean that the system has been tampered and will not be able to take part in future operations [5].

In terms of issues related to key compromise of a device, ICMetrics proves to be breakthrough technology, thereby generating keys for secure identification of the device. However the ICMetrics generated keys can be weak, since they maybe short and of low entropy; and could infact pose a severe threat to the security of the system. Since a stable ICMetrics generated basis number is small in size with low entropy, it makes the underlying device open to attacks such as brute force and exhaustive search attacks. These attacks can in turn completely compromise the security of the application and even lose the essence/advantage of ICMetrics. The proposed scheme intends to resolve the issue of weak keys by generating strong keys with high entropy length, thereby making ICMetrics a very viable option for secure systems.

III. Security Primitives

All security applications require proper generation and maintenance of keys during their lifecycle. The security of an application also depends on the strength of the cryptographic key. In the following section we briefly describe the various security primitives that we have employed in our proposed scheme for the generation of strong key pairs based on ICMetrics:

A. Key Generation and Public Key Cryptography

Key generation is one of the most sensitive cryptographic functions, since the security of the entire application depends on the proper generation of keys. Keys are normally generated using random number generators (RNG) or pseudorandom number generators (PRNG). Cryptographic keys that form a part of symmetric key schemes have a single secret key that is shared between communicating parties and is used for all encryption/ decryption operations. However a major drawback of symmetric keys is the severity of cryptanalytic attacks. Cryptographic keys that form part of asymmetric key schemes have two parts; a private key and a corresponding public key. A public key can be publically distributed whereas a private key has to be kept secret. The public key is used by other parties to send messages securely to the person that generated the key pair while a private key is used to decrypt messages. Furthermore, with the help of trusted third party, all the interaction can be carried out in a secure and confidential manner [26].

B. Key Derivation Function

A Key Derivation Function (KDF) is a special transformation function that can bring a number in raw form to a form that can be securely used as a key for secure operations [16-17]. This transformation on the raw number will safeguard the key against brute force attacks and exhaustive search attacks[14]. This function is essential in all security applications and generates keys with high entropy that can be safely used in security critical applications. The key derivation function, takes the raw password and a random salt as input, and applies multiple iterations of a function H (such as hash or block cipher); to generate keys with high entropy. The random salt value in the KDF hinders the attacker's ability to construct rainbow tables, thereby protecting the system against pre-computed hashing attack/rainbow table attacks. The salting is required as the lack of it could allow an attacker to work out how many times a password hash occurred to reveal the original password.

C. Key Stretching

Key stretching is a method to hinder an attacker's ability to reproduce a key derivation function [19]. Key stretching makes use of iterated hashing to generate keys of a particular length with high entropy. Key stretching strengthens the key against brute force attacks by increasing key length and entropy thereby making it infeasible to launch brute force attacks. Although key derivation functions and key stretching give a sense of similarity, since they are both based on hash generation functions to generate high entropy keys [18]. However their design principles are significantly different, since key stretching tends to derive long keys by concatenating the output of each hash function iteration while Key Derivation Functions (KDF) iteratively hash the same input key to produce high entropy key.

D. Hash Function

A hash function [20] is a deterministic function that takes an arbitrary length bit string as input and outputs a fixed length bit string called a hash value. A hash transformation H, applied on an arbitrary sized input and mapped to a fixed length output 'n' is denoted by:

$$H: \{ 0, 1 \}^{*} \rightarrow \{ 0, 1 \}^{n};$$

For a function to qualify as a hash function it should typically possess the following properties:

- H should be pre-image resistant; such that given a hash value 'h' it is computationally infeasible to find m.
- H should be collision resistant; it must be computationally infeasible to find two different inputs that hash to the same value.
- The hash function H should be publically known and the hash value H(m) of an input m can be found efficiently.

The length of generated hash value depends on the hash function [21]. The well-known hash algorithms include MD5, SHA1, SHA2 etc.; each having variants of differing hash value lengths for an input block [20-21].

E. SHA-2

SHA-2 [19] is a set collision resistant cryptographic hash functions proposed by the National Security Agency (NSA). It is the second in Secure Hash Algorithm (SHA) series designed by the NSA. SHA-2 is a stronger hash function due to its security properties from its predecessor SHA-1 [20]; and is the standard adopted in many security applications and protocols such as TLS, IPSec, etc. SHA-2 has four hash variants, namely SHA-224, SHA-256, SHA-384 and SHA-512, each having varying sized digests. SHA-224 outputs a 224 bits digest, computing on block sizes of 512 bits in each of the 64 rounds of the hash computation operation. SHA-256 outputs a 256 bits digest, working with 512 bit blocks in each of the 64 rounds of hash computation operation. SHA-384 and SHA-512 both work with block sizes of 1024 bits in each of the 80 rounds while giving digest sizes of 384 bits and 512 bits respectively [24].

F. Hermite Normal Form

In our research, the public key for the generated high entropy private key is generated based on Hermite Normal Form (HNF) of the high entropy private key due to the uniqueness of the HNF. Every matrix has a unique corresponding Hermite Normal Form 'H' and can be bought into its corresponding Hermite Normal Form by a sequence of elementary column operations.

A non-singular square matrix 'H' is said to be in Hermite Normal Form (HNF) if it has the following properties[6][7]:

- $h_{ij} = 0$ for $i < j$ (i.e., H is lower triangular);
- $0 \leq h_{ij} < h_{ii}$ for $i>j$ (i.e., H is non-negative and each row has a unique maximum entry, which is on the main diagonal).

IV. Network Topology

Our proposed scheme is an idea for networked environments and all entities/ devices forming part of the network have trust in a Trusted Third Party (TTP), and similarly all TTP's trust the Master TTP (MTTP) as shown in Fig. 1. The MTTP is controlled by the user and is responsible for controlling all TTP's in the network, whereas TTP's are responsible for controlling their own localized networks i.e. the individual entities. Trusted Third Party (TTP) enables entities that never had contact before to interact securely and confidentially. For the proposed scheme we make the following assumptions:

- The communication links between the MTTP, TTP and entities are all unprotected. Therefore the data being transmitted over the communication channel should be protected.
- The ICMetrics data of each device is only stored in the device for the duration of the session.

Authentication between all the participating entities takes place based on traditional authentication protocols [22-27] and is not the focus of this paper.

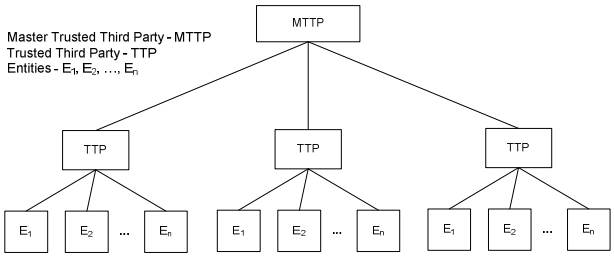

Fig. 1. Network Topology for Proposed Architecture

V. The Proposed Security framework

Our proposed security framework details a scheme that generates strong session key pairs based on ICMetrics basis number. The proposed scheme is an idea to improve the security of ICMetrics keys by generating high entropy ICMetrics keys that are resilient against brute force and rainbow table attacks. The following section details the steps involved for the generation of strong public/ private session key pairs based on ICMetrics:

A. Key Setup

Each device that is part of the network generates an ICMetrics basis number based on the extracted feature values. The basis number is formed by the addition of individual feature values to generate a short (35 bits) but stable basis number and hence serve as a secret key. The key setup algorithm generates the master private key for the Master Trusted Party (MTTP), Trusted Third Party (TTP) and all other entities that form a part of the network. The ICMetrics generated basis number of the MTTP and the TTP serves as its master private key.

Master Private Key of Master TTP 'MMPrt' = Basis Number using ICMetrics of Device

Master Private Key of TTP 'MPrt' = Basis Number using ICMetrics of Device

Similarly all entities with identities ID1, ID2, …, IDn also generate their its master private key's MPr1,MPr2,…, MPrn respectively based on their ICMetrics basis number and hence use their master private key for further operations as shown in fig. 2.

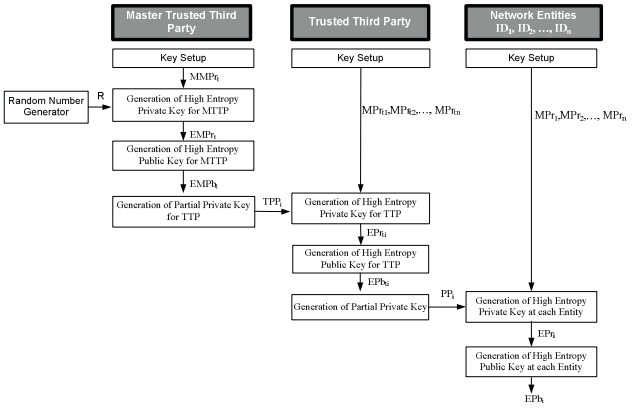

Fig. 2. Functional Diagram of Proposed Design

B. Generation of High Entropy Private Key for Master TTP

This algorithm is responsible for generating a high entropy private key for the MTTP which it can use for further operations. The high entropy private key for the MTTP is generated based on its own master private key 'MMPrt' and a 128 bit random number. Both of these are combined to generate high entropy private key 'EMPrt' for MTTP, that it uses to communicate to the TTP's and for secure onward operation, as shown in Fig. 3.

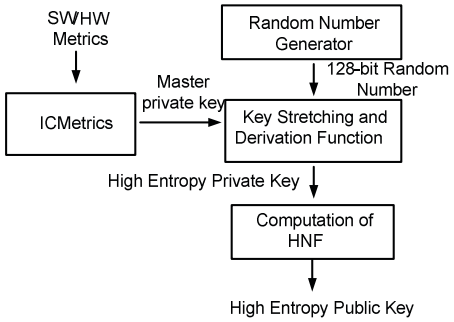

Fig. 3. High Entropy Key Generation at MTTP

Both the 128 bit random number and MMPrt are combined via SHA-2, as shown in Table 1. 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 derivation function and safeguards against rainbow table attacks. By adding a salt value to the MMPRt, the possibility of using pre-computed hashes for attacks is reduced.

Table 1. Proposed Key Derivation and Stretching Algorithm for MTTP's Key

X_0 = SHA-2(MMPr$_t$, 128 bit random value)
For s = 1 to n
X_i = SHA-2(MMPr$_t$, X_{s-1})
$X_0 = X_0 \| X_i$
High Entropy Private Key 'EMPr$_t$' = X_0

As stated above, SHA-2 is used for the purpose of key stretching and derivation using a random salt, since it will both bring an increase the entropy and length of the key and at the same time safeguard against rainbow table attacks. SHA-2 based key stretching and derivation algorithm takes the ICMetrics generated basis number and a random 128 bit salt as input and after performing multiple iterations of SHA-2 rounds, it produces a longer stretched high entropy key that is safe from pre-computed hashing attacks. Now the generated key 'X0' is broken into blocks of equal size according to the required key length; starting from the right and appending zeros to the left most block to make its size compatible with rest of the blocks.

Then finally all blocks are XORed to generate the required sized key, which gives the final high entropy private key for the entity IDi. Exclusive Ors add an extra layer of protection but the actual security derives from the iterations of the hashing function and the salt value.

C. Generation of High Entropy Public Key for Master TTP

Subsequently, public key is computed from the high entropy private key 'EMPrt' using the following:

Public Key of MTTP 'EMPbt'= Hermite-Normal-Form (High Entropy Private Key of MTTP 'EMPrt')

The MTTP has a high entropy public/private key pair generated that it uses for secure onward operations with other TTP's in the network.

D. Generation of Partial Private Key for the TTP$_i$

This algorithm is run by the MTTP in which it generates a partial key for each TTP. The identity 'ID_{Ti}' of a TTP is randomly generated by the MTTP at a network join request. The size of this random identity information associated with a device can range from anything between 48-128 bits in length, by setting a trade-off between the required security and the resources available.

1. ID_{ti} = 128 bits Random Number
2. Compute Q_{ti} = SHA-2 (ID_{ti})
3. Output partial private key 'PP_{ti}'=($EMPr_t$)x(Q_{ti})

The generated partial private key 'PP_{ti}' generated by the MTTP is sent to entity TTP_i. The process of supplying partial private keys takes place confidentially and authentically, the MTTP ensures that the partial private keys are delivered securely to the TTP. The partial key sent by the MTTP is used

by the TTP for the generation of its high entropy private key. The MTTP removes the partial key from its memory once it has been communicated to the TTP. In future, whenever the entity requests the MTTP for a partial key, the MTTP generates a new random value for the TTP which can serve as its 'ID_{ti}' and uses it for the generation of a new partial key for the TTP_i. The generation of partial key based on a new random ID every time a request is made, helps safeguard from pre-computed dictionary attacks on the keys.

E. Generation of High Entropy Private Key at each TTP

The TTP with identity IDti, combines both the MTTP generated partial private key 'PPti' and its own master private key 'MPrti' to generate high entropy private key 'EPrti' for IDti, that can be used for secure onward operation, as shown in Fig. 4.

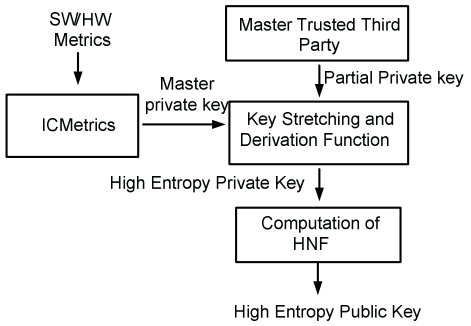

Fig. 4. High Entropy Key Generation at each TTP

Both PPti and MPrti are combined via SHA-2, as shown in Table 2. The purpose of the SHA-2 based key stretching and derivation algorithm is to combine and thus stretch the key, so that it qualifies for use in secure operations.

Table 2. Proposed Key Derivation and Stretching Algorithm for TTP's Private Key

$X_0 = SHA\text{-}2(MPr_{ti}, PP_{ti})$
For s = 1 to n
$X_i = SHA\text{-}2(MPr_{ti}, X_{s-1})$
$X_0 = X_0 \parallel X_i$
High Entropy Private Key 'EPr$_{ti}$' = X_0

A new partial private key for a session based on a random IDti, prevents rainbow table attacks. As above, we propose the use of SHA-2 for the purpose of key stretching and derivation since they will both bring an increase in the key length as well as increase the entropy of the key, while safeguarding from rainbow table attacks. The SHA-2 based key stretching and derivation algorithm takes the ICMetrics generated basis number along with a new partial key as input and after performing multiple iterations of SHA-2 rounds, it produces a longer stretched high entropy key, which is highly resistant against rainbow table attacks.

Now the generated key 'X0' is broken into blocks of equal size according to the required key length; starting from the right and appending zeros to the left most block in order to make its size compatible with rest of the blocks. Then finally all blocks are XORed to generate the required sized key, which gives the final high entropy private key for the entity IDi thereby adding an extra layer of protection.

F. Generation of High Entropy Public Key for TTP

This step involves the TTP computing its corresponding public key from the high entropy private key 'EPrt' using:

Public Key of TTP 'EPbt'= Hermite-Normal-Form (High Entropy Private Key of TTP 'MPrt')

Finally the TTP has a high entropy public/private key pair generated that it can use for onward operations/ communication with other entities in the network. Once these key pairs have been used for the required purpose, they are removed from the memory. If the high entropy key pair is required in future it is regenerated based on ICMetrics values of the device.

G. Generation of Partial Private Key for Entity E_i

This algorithm is run by the TTP in which it generates a partial key for an entity that requests the TTP to form part of the network. The identity 'IDi' of an entity is randomly generated by the TTP at a network join request. The size of this random identity information associated with a device can range from anything between 48 bits to 128 bits is length, by setting a trade-off between the required level of security and the resources available.

1. ID_i = 128 bits Random Number
2. Compute Qi = SHA-2 (ID_i)
3. Output partial private key 'PP_i'=(EPr_t)x(Q_i)

The generated partial private key 'PPi' generated by the TTP is sent to entity IDi. This process of supplying partial private keys takes place confidentially and authentically. The partial key sent by the TTP is used by the entity for the generation of its high entropy private key. The TTP removes the partial key from its memory once it has been communicated to the entity. In future, whenever the entity requests the TTP for a partial key, the TTP generates a new random value for the entity which can serve as its 'IDi' for operations. The generation of partial key based on a new random ID every time a request is made, helps safeguard from pre-computed dictionary attacks on the keys.

H. Generation of High Entropy Private Key at each Entity

The user with identity IDi, combines both the TTP partial private key 'PPi' and its own master private key 'MPri' to generate high entropy private key 'EPri' for IDi, that can be used for secure operation, as shown in Fig. 5.

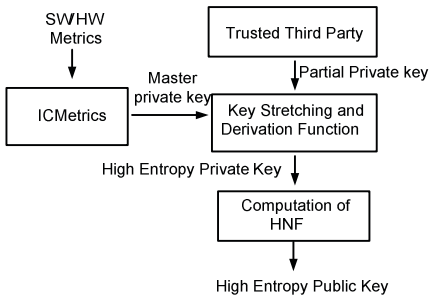

Fig. 5. High Entropy Key Generation at each Participating Entity

Both PPi and MPri are combined via SHA-2, as shown in Table 3. The SHA-2 based key stretching and derivation algorithm takes the ICMetrics generated basis number and a newly generated partial key as input and after performing multiple iterations of SHA-2 rounds, it produces a longer stretched high entropy key, that is secure against rainbow table attacks.

Table 3. Proposed Key Derivation and Stretching Algorithm for Entity's Private Key

$\mathbf{X_0 = SHA\text{-}2(MPr_i, PP_i)}$
For s = 1 to n
$\mathbf{X_i = SHA\text{-}2(MPr_i, X_{s-1})}$
$\mathbf{X_0 = X_0 \,||\, X_i}$
High Entropy Private Key 'EPr$_i$' = X$_0$

Now the generated key 'X0' is broken into blocks of equal size according to the required key length; starting from the right and appending zeros to the left most block to make its size compatible with rest of the blocks. Then finally all blocks are XORed to generate the required sized key, which gives the final high entropy private key for the entity IDi.

I. Generation of High Entropy Public Key at each Entity

The public key is computed from the high entropy private key 'EPri' using:

Public Key of IDi 'EPbi'= Hermite Normal Form(High Entropy Private Key of IDi 'EPrt')

Finally the entity IDi now has a high entropy public/private key pair generated that it uses for secure onward operations/ communication with other entities in the network.

VI. Security Analysis

In the proposed scheme we have combined the security advantages of the ICMetrics secret key, random session tokens and security properties of SHA-2 based key derivation function for the generation of strong session based private key. The security of our scheme is based on a session token issued by the trusted party and the ICMetrics basis number of the device fed to the SHA-2 based key derivation function. These input values are rehashed the required number of times to produce a strong session based public/private key pair. To safeguard the generated strong key from various cryptanalytic attacks, such as brute force or rainbow table attacks; the ID of each node is randomly generated and remains valid only for the duration of the session. This random session ID is used in the generation of session token and is used for further key derivation operations. The session token also serves to authenticate/identify each participating node, thereby assuring the origin of information; since only entities that have been assigned session tokens from their respective trusted parties can communicate with other entities in the network. To regenerate the public/private key pair during a particular session, the generating entity must have the knowledge of both the session token and secret basis number. Knowing only one of them does not allow the generation of key pairs. This safeguards the network from attackers, since only authenticated entities that have been issued a partial secret by the TTP can form part of network. The SHA-2 based key stretching and derivation operation on the ICMetrics secret key, as previously indicated results in a longer and high entropy private key that helps safeguards against brute force/rainbow table attacks.

The second phase of our scheme generates a corresponding strong public key by computing the HNF. The corresponding public key of a generated strong private key is generated by computing the HNF of the private key. The Hermite Normal Form of a number is unique and non-reversible, therefore it proves to be a very good choice for the computation of the corresponding public key. Each participating entity's public key is made available to other entities by transmitting it along with messages or by placing it in a public directory. But no further security is applied to the protection of A's public key. This idea helps preserve the security properties of ICMetrics generated keys and at the same time generates high entropy key pairs for use in future operations.

VII. Conclusion and Future Work

A framework for generation of strong high entropy session key pairs of sufficient length for ICMetrics secret key is introduced in this paper. The proposed scheme effectively combines the functionalities of ICMetric keys coupled with session tokens from the trusted third party, thereby providing authentication and identification of the device. In this paper, we have designed a password-based key derivation function, in which we employ SHA-2 for stretching the ICMetrics key using session token from the TTP, to generate a high entropy ICMetrics private key. Random session token's make the generated key is strongly resistant to brute force and rainbow table attacks. The high entropy public key corresponding to the high entropy key is computed by calculating the Hermite Normal Form of the high entropy private key. The proposed scheme has been very carefully tuned with the underlying requirements of ICMetrics, while making use of random values to safeguard against cryptanalytic attacks.

In future we plan to evaluate our scheme through experiments and analysis, thus benchmarking the results

against existing key generation schemes. We are confident that our scheme will be a viable solution for secure generation of strong high entropy key pairs. We also plan to design a protocol for secure communication based on the generated high entropy key pairs.

REFERENCES

[1] I. Ryu, “Issues and Challenges in Developing Embedded Software for Information Appliances and Telecommunication Terminals”, Proceedings of the ACM SIGPLAN 1999, LCTES’99, Vol. 34, Issue7, July 1999, pp. 104-120.

[2] P. Koopman, “Embedded System Design Issues- The Rest of the Story”, Proc. of Intl. Conference on Computer Design, Austin, 7-9 October, 1996, pp. 115-121.

[3] P. Koopman, "Design Constraints on Embedded Real Time Control Systems", Conf. on Sys. Design and Network Architecture, May 8-10, 1990, pp. 71-77.

[4] L. Khelladi, Y. Challal, A. Bouabdallah, N. Badache, “On Security Issues in Embedded Systems: Challenges and Solutions”, International Journal of Information and Computer Security 2008, Vol. 2, No.2, pp. 140-174.

[5] R. Tahir, K. D. McDonald Maier, “Improving Resilience against Node Capture Attacks in Wireless Sensor Networks using ICMetrics”, IEEE Conference on Emerging Security Technologies, Portugal, September 5-7, 2012.

[6] C. Henri, “A Course in Computational Algebraic Number Theory”, Graduate Texts in Mathematics, Berlin, New York.

[7] G. Shmonin, “Hermite normal form: Computation and applications”, http://disopt.epfl.ch/webdav/site/disopt/shared/IntPoints2009, Feb. 24, 2009.

[8] E. Papoutsis, W. G. J. Howells, A. B. T. Hopkins, and K. D. McDonald-Maier, “Integrating Multi-Modal Circuit Features within an Efficient Encryption System”, Third International Symposium on Information Assurance and Security, IEEE Computer Society Washington, DC, USA, 2007, pp. 83-88.

[9] E. Papoutsis, W. G. J. Howells, A. B. T. Hopkins, and K. D. McDonald-Maier, "Ensuring Secure Healthcare Communications via ICmetric based Encryption on unseen Devices", Symposium on Bio-inspired, Learning and Intelligent Systems for Security, Edinburgh, 20-21 Aug. 2009, pp. 113-117.

[10] E. Papoutsis, W. G. J. Howells, A. B. T. Hopkins, and K. D. McDonald-Maier, "Integrating Feature Values for Key Generation in an ICmetric System," in IEEE NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2009) San Francisco, California, 2009, pp. 82-88

[11] G. O. Karame, S. Capkun, U. Maurer , “Privacy-Preserving Outsourcing of Brute-Force Key Searches”, Proceedings of the 3rd ACM workshop on Cloud Computing Security, New York, USA, 2011, pp.101-112.

[12] F. F. Yao, Y. L. Yin, “Design and Analysis of Password-Based Key Derivation Functions”, IEEE Transactions on Information Theory, Vol 51(9), pp. 3292-3297.

[13] R. Tahir, K. D. McDonald Maier, “An ICMetrics based Lightweight Security Architecture using Lattice Signcryption”, IEEE Conference on Emerging Security Technologies, Portugal, September 5-7, 2012.

[14] B. Kaliski, “PKCS #5: Password-Based Cryptography Specification Version 2.0”, RFC 2898, Sept. 2000.

[15] C. Adams, G. Kramer, S. Mister and R. Zuccherato, “On the Security of Key Derivation Functions”, Conf. on Industrial Simulation, Spain, pp. 134-145.

[16] J. Kelsey, B. Schneier, C. Hall, and D. Wagner, “Secure Applications of Low-Entropy Keys”, Information Security Workshop (ISW 1997), Japan, pp. 121-134.

[17] NIST's Policy on Hash Functions, National Institute on Standards and Technology Computer Security Resource Center, March 29, 2009.

[18] C. Paar, J. Pelzl, “Hash Functions-Understanding Cryptography, A Textbook for Students and Practitioners", Springer, 2009.

[19] X. Wang, Y. L. Yin, and H. Yu, “Finding Collisions in the Full SHA-1”, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, pp. 17-36.

[20] B. Schneier, "Schneier on Security: Cryptanalysis of SHA-1", Schneier.com.

[21] “Secure Hash Standard (SHS)”, FIPS PUB 180-3, October 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

[22] A. A. Pirzada, C. McDonald, “Kerberos Assisted Authentication in Mobile Ad hoc Networks” Proceedings of the 27th conference on Australasian Computer Science, Australia, Vol 26, pp. 41-46.

[23] J. Clark, J. Jacob, “A Survey of Authentication Protocol Literature: Version 1.0 17”, November 1997.

[24] J. T. Kohl, B.C. Neuman, “The Kerberos Network Authentication Service”, RFC1510, 1993.

[25] D. Harkins, D. Carrel, “The Internet Key Exchange (IKE)”, RFC2409, 1998.

[26] B. Schneier, “Applied Cryptography: Protocols, Algorithm, and Source Code in C”, John Wiley & Sons, Inc., 1996.

[27] X. Li, J. Han, Z. Sun, “Design Principles and Security of Authentication Protocols with Trusted Third Party”, AUUG 2004, Australia, pp. 103-107.