An Locking and Unlocking Primitive Function of FSM-modeled Sequential Systems Based on Extracting Logical Property

Edward Jung*, Chih-Cheng Hung**, Ming Yang* and Seonho Choi***

*School of Computing and Software Engineering, Center for Biometrics Research, Southern Polytechnic State University, GA, USA

** Anyang Normal University, Anyang, China
and School of Computing and Software Engineering, Center for Biometris Research, Southern Polytechnic State University, GA, USA

*** Department of Computer Science, Bowie State University, MD, USA

Abstract

In this paper, we attempt to address the following question in the field of computer security: "can we build a primitive security function or a building block *without* an explicit modification of the original system for a class of systems modeled in a deterministic FSM?" As one of possible solutions, we propose a *logical (or functional) property extraction*-based solution (in general) and a *synchronicity*-based sequential locking and unlocking method (in particular). The proposed method is solely based on *extracting* an inherent *property* embedded in the original D-FSM. The property being considered is synchronicity due to the desirable characteristic that can be used for the design of an efficient sequential locking and unlocking method. The proposed method and its application to digital watermarks (as one of potential applications) are presented and analyzed. Both the feasibility and the limitation are discussed. However, the basic idea is general in a sense that (1) it can be applied to other security applications, and (2) other logical properties can be further explored.

Key Words: Property-based Security, Functional property, Synchronicity, Locking/Unlocking, Digital Watermarks, D-FSM

1. Introduction

One of core functions in designing secure systems is the mechanism of locking and unlocking (e.g., a password-based mechanism). In designing such a system (e.g., embedded systems or controllers), the system is frequently modeled using a deterministic finite-state machine (D-FSM) [8, 17]. Traditionally, the approach to the D-FSM based security solutions has focused on modifying the original D-FSM. Despite the higher-level of security that can be potentially achieved by this modification-based solution, it suffers from the increased complexity of modifying the original D-FSM and the associated overhead (e.g., performance, area). In many security applications, building an efficient locking and unlocking is important.

To address this problem, we proposed a solution ([7]) based on extracting the property (i.e., synchronicity) embedded in the original D-FSM and using that property to build a security building block (i.e., locking/unlocking). This does not require modifying the system (i.e., D-FSM) and the benefit of this approach is efficiency.

In this paper, we extend the earlier work [7] to the problem of digital watermarks as one of potential applications. Fundamentally, our work (both in this paper and [7]) is based on our belief (or principle) that any security solutions must minimize (or avoid if possible) the changes or modifications in the system solely to improve security. In doing so, we are promoting a *logical property extraction*-based security solution as much as possible (i.e., if the system possess a desirable property). In this approach, a certain type of property will be defined first, and then be extracted from the system, and finally be applied to build a certain type of security primitive functions (e.g., a locking/unlocking method) or the broader security solution itself. The contributions of this work can be summarized as follow:

- We propose fundamentally a different approach to dealing with the overheads of security solutions in a certain class of system (i.e., FSM-based sequential systems).
- We develop a locking and unlocking method as one of the security building blocks. The method is based on simply extracting the system's inherent logical property (i.e., synchronicity) embedded in D-FSM.
- Simple algorithms are devised for both the automatic generation of key input sequence (for locking) and for using the key input sequence to unlock the system (for unlocking).
- For a higher level of security (e.g., the military application), we outline a re-locking enforcement mechanism with a minimal modification of D-FSM.
- Using digital watermarks, we show that a set of watermark properties and attack resiliency can be satisfied with the proposed method. Collision probability is estimated.

Preliminaries are provided in Section 2. The locking and unlocking method is described in Section 3. In Section 4, we consider digital watermarks as an application. An enforcement method is outlined in Section 5 and the conclusion is made in Section 6.

2. Preliminaries

2.1 Application model

Producer P is an entity that has the ownership of a creative work W (e.g., the design specification of a sequential system). Consumer C is an entity that needs to use W. In the horizontal business model (e.g., an outsourcing model), the relationship between P and C is *asymmetric*: W is not transparent to$C$. In the semiconductor industry, an example of P and C can be the *design specification* of IC/RTL/HDL-level blueprint (i.e., modeled in D-FSM) and a *foundry* which needs to use the blueprint, respectively.

2.2 Preliminary background

The following assumptions (A1, A2, and A3) are commonly used in the related literatures [2, 3, 4]. A1 is general while A2 and A3 are specific in the related security application domain.

- **A1**: A system is modeled with a deterministic finite state machine (D-FSM). D-FSM= $(I, O, S, \delta, \lambda)$, where$I$,$O$, and S are finite sets of inputs, outputs, and states, respectively. $\delta: I \times S \rightarrow S$ is the state transition function.$\lambda: I \times S \rightarrow O$ is the output function.
- **A2**: D-FSM has both a *power-up* state S_p and a *starting* state S_s.
- **A3**: The system (i.e., D-FSM) will function *only* when it is starting from S_s.

Synchronicity: The synchronicity of D-FSM has been studied [8, 12]. The study was conducted based on two notions: (1) uncertainty, and (2) synchronizing sequence. The primary results can be summarized as follows:

- **Uncertainty as non-increasing function**: Initially, a machine M[1] can be in any one of its "*n*" states. That is, the initial uncertainty of a machine is the entire set of states. By applying a sequence of input, the set of states that the machine can move (i.e., future uncertainty) will not increase.
- **Synchronizing sequence**: *Regardless of the initial state* or the output, a synchronizing sequence of M is a sequence which takes M to a specific final state, if M possesses the sequence.

The systematic way of deriving a synchronizing sequence was developed using a synchronizing tree T [8]. T is a binary tree, where a root node represents the initial uncertainty and the depth of T corresponds to the length of binary input sequence. For a given$M$, T can be constructed until some node is associated with an uncertainty containing just a single element known as a *singleton* state (i.e., no uncertainty). For the detailed description, refer to [8]. For the specific example relevant to the proposed work, refer to our previous paper [7].

2.3 Summary of previous work

Much of work done in designing D-FSM-based security solutions are based on modifying the structure of D-FSM [2, 3, 4, 6, 12]. Some has added dummy states while others have introduced additional edges between states. Despite the original functionality that is preserved, both an additional overhead and the increased level of design complexity have been reported for this approach. Another approach, possibly more promising one than changing the structure of FSMs, is to extract a *physical* property from the system [1]. However, these approaches are different from our approach which is based on extracting logical property that is inherently embedded in D-FSM. Thus, our approach is specific design-agonistic and can be used during the pre-optimization phase. With the logical level extraction, any design optimization techniques should be more effectively used.

[1] A deterministic finite-state machine D-FSM and simply, a machine M are used interchangeably.

3. Locking and Unlocking Method - General

We describe three necessary operations that are used in related applications: (1) creating a lock (locking), (2) embedding a lock, and (3) unlocking. The first two operations are performed by the producer P while the consumer C exercises the third operation.

3.1 Creating a lock (Locking)

A lock is fundamentally secret information created by P. Many possibilities of creating a lock exist. One way is to hide S_s. If this approach is taken, a specific sequence of input that will transit S_p to S_s is important and thus can serve as secret known as the pass key K_s. In this case, a lock can be defined as a pair of $< S_s, K_s >$. Another possible approach is to define a lock in terms of S_s (solely). In this case, however, the consumer C has to derive S_s by running some experiment (e.g., synchronizing experiments requiring the construction of a synchronizing tree). We take the former approach since it can potentially provide more flexibility to P and also less burden on C.

In a 'n"-state D-FSM, any state can be chosen as a locking state, if a certain condition is met. The condition is that the chosen locking state S_i must be *singleton* in the synchronizing tree T. If there are more than one singleton states, we choose the one in the lowest level. If there are more than one singleton states in the same level, choose one randomly. In this way, the singleton state S_s will provide the longest path from itself to the root node (or vice versa). The steps for creating a lock (by P) are as follow: Given an "n"-state D-FSM,

---

```
1)  Construct "synchronizing tree T";
2)  FOR LOOP (Level "l" = 1, 2, …., j)
        a.  If exist, record a set of all "singleton" states {S(l, 1), S(l, 2), ….}; /*
            at level "l"*/
3)  Determine the singleton state S* at the lowest level "l"; if more than one states,
    choose one randomly at the lowest level "l";
4)  Work backward from S* to the node at the root level (level 0) to construct the
    "longest" "synchronizing sequence"; /* this sequence becomes the pass key Ks */
5)  Send Ks to Consumer C in a secure way.
```

Note that the complexity of algorithm is $O(n^3)$ due to the construction of the synchronization tree T. Step 5 can be done using either public key system (e.g., PKI), symmetric key system (i.e., a shared key), or other method [14]. Also, note that in Step 5 only K_s will be sent to C. This is due to the property of "synchronicity" which will guarantee the D-FSM move to the correct initial state S_s, irrespective of the power-up state on C.

3.2 Embedding a lock

In our approach, a lock is (or should be) *embedded* in the D-FSM itself. There is no need to

explicitly embed the lock (which is done by the traditional method.) This is one of the key advantages of the proposed method.

3.3 Unlocking

Consumer C can unlock the system using K_s, the key input sequence, received from P in Step 5 (Section 3.2). The steps for unlocking are as follow:

```
1)  Power up the system; /* The system can be in "any" one of the states; initial
    uncertainty */
2)  Apply Ks to unlock the system; /* The system should be in the starting state Ss */
```

Note that the unlocking operation is simple. However, it is simple only for C who knows K_s. In the following section, we provide the analysis for the length (i.e., least upper bound) of pass key (K_s). It is known that the unlocking step is usually the most expensive step in many security applications [9, 16].

Limitation: Synchronizing sequence(s) do not exist in all sequential systems or D-FSMs. Thus, the proposed method can not be applied to a sequential system or D-FSM which does not possess a synchronizing sequence. A possible solution is provided in Section 6.

3.4 Analysis – Length of Synchronizing Sequence (K_s)

Intuitively, the longer the length of K_s, the higher the security level is. Since it is known that the least upper bound on the length of a synchronizing sequence is unknown, we should focus on analyzing a range of values, [Q_{min} min($least\ upper\ bound$) , Q_{max} max($least\ upper\ bound$)]. Based on [11], we present the results that are relevant to the study of this work below.

Theorem 1: If an n-state D-FSM has a synchronizing sequence, or sequences, then it has one such sequence whose length is at most $\frac{n(n+1)(n-1)}{6}$.

Proof: To reduce the initial uncertainty $S_1 S_2 \ldots S_n$ to a singleton uncertainty, it takes

$1 + (1+2) + \cdots (1+2+\cdots+n-1) = \sum_{k=2}^{n} \frac{k(k-1)}{2}$ Since $\frac{k(k-1)}{2} = 0$ for $k = 1$, $\sum_{k=2}^{n} \frac{k(k-1)}{2} = [\frac{n(n+1)(2n+1)}{6} - \frac{3n(n+1)}{6}] = \frac{n(n+1)(n-1)}{6}$ ■

Theorem 2: For every n, there exists an n-state D-FSM which has a synchronizing sequence of length$(n-1)^2$.

Proof: Refer to the proof in Theorem 13-5 [8] ■

Corollary 1: The least upper bound L on the length of synchronizing sequence is bounded by$(n-1)^2 \leq L \leq \frac{n(n+1)(n-1)}{6}$.

Proof: Directly from Theorem 1 and Theorem 2 ■

Theorem 3: For every *n*, there exists n^2 ways of constructing a pair of states $< S_p, S_s >$.

Proof: In an *n*-state D-FSM, there exists "*n*" number of possible power-up states S_p and "*n*" number of possible starting state S_s . Thus, the total number of pairs is n^2 ■

Corollary 2: The least upper bound Q on the number of ways of unlocking the system by trying all possibilities is bounded by $n^2(n-1)^2 \leq Q \leq (n^2)\frac{n(n+1)(n-1)}{6}$.

Proof: Directly from Corollary 1 and Theorem 3 ■

Implication: Corollary 2 says that if both K_s and S_s are unknown, the adversary would try this many possible ways to unlock the system (in the worst case). The feasibility of applying the synchronicity to the benchmark systems is shown in Table 1. (For the detailed discussion, refer to [7])

Table 1. The minimum and maximum least upper bound for the sample circuits using benchmark sequential circuits [ISCAS'89][5], where *: a number at least bigger than "1.9e+25"; **: a number at least bigger than "4.5e+30"

Circuits	No. of FFs	No. of States	Least Upper Bound Q (min)	Least Upper Bound Q (max)
s27	3	8	3136	3584
s820, s832	5	32	984064	3724629
s1488, s1492, s386, s510	6	64	16257024	119275520
s208	8	256	4261478400	122166094506
s27-n3	9	512	68451303424	3909359763456
S1196, s1238	18	262144	4.7e+21	1.3e+26
s991	19	524288	7.5e+22	4.4e+27
s382, s400, s444, s526, s526n	21	2097152	1.9e+25	4.5e+30
s635, s838	32	4294967296	*	**
s15850	597	5.1e+179	*	**
s35932	1728	2^{1728}	*	**

4. Digital Watermarking - Application

We consider digital watermarking as an application. First, we do the simple mapping between the general locking/unlocking method (i.e., three operations in Section 3) and the watermarking process. Then, we address the properties of watermarks and attack resiliency. Finally, we analyze the collision probability.

4.1 Mapping

A digital watermarking process is very similar to the three operations described in Section 3.1. The following three-step processes are used in digital watermarks [9, 13]: (a) signature (i.e., watermark) creation, (b) signature embedding, and (c) signature verification. The process can be directly mapped to the three operations.

4.2 Watermark Properties and Attack Resiliency

A digital watermark must satisfy a set of properties to provide security and resiliency against attacks [9]. We consider the following main properties (Table 2) and demonstrate that they can be satisfied by the methods proposed in Section 3.

Table 2. Properties of digital watermarks in a form of both general and specific properties.

Property	Specific properties	
Unobtrusiveness	The presence of signature (i.e., watermark) should not change/modify the original functionality of D-FSM.	P1.1
	The signature should be invisible to the design (i.e., D-FSM)	P1.2
Universality	The same watermarking method should be applicable to all common sequential systems (i.e., D-FSMs)	P2
Unambiguity	Retrieving the signature should be conclusive in proving the ownership.	P3
Robustness	The embedded signature should be difficult to remove.	P4.1
	Multiple producers or individuals who create their signatures should have a unique set of characteristics (i.e., avoid the *collision of watermarks*)	P4.2

In Table 3, we provide the informal reasoning for satisfying each of the watermark properties (P1.1 – P4.2) listed in Table 2.

Table 3. Satisfying the watermark properties with the reasoning

Specific Properties	Reasoning
P1.1 Preservation of functionality	Any explicit changes to the design (i.e., D-FSM) are not made. The original functionality should be preserved.
P1.2 Invisibility	The signature is naturally embedded in the original design (D-FSM).
P2. Universality	The proposed method can be used for all common sequential systems, as long as the system possesses the synchronicity property. [NOTE: 94.6% of the benchmarks MCNC89 FMSs possess the synchronicity property [12]]
P3. Unambiguity	The probability of finding a valid sequence is very low without knowing the secret key pass. The retrieval of a signature is a strong proof of ownership. [For further improvement, see Section 5]
P4.1 Removal of signatures	The signature is an internal property of system. Removing an internal property of system will be not only difficult but will change the functionality of the system.
P4.2 Collision of signatures	The probability of collisions is very low. Guessing the secret key correctly (K_s) and the staring state (S_s) will be very difficult. [See Section 4.3]

4.3 Analysis – Collision Probability

In this section, the collision probability is *estimated*. First, we describe the general problem. Then, the problem of estimating collisions for the proposed method is addressed using relaxed assumptions. Most of analysis in this section is performed based on [1].

General Problem: *what is the probability that among K possible objects drawn from the population* $\Omega = \{1, 2, \dots, n\}$ *at least two have the same value?* (i.e., the birthday problem).

Specific Problem: *Given a collection of K binary sequences, each of length L, what is the probability of collision among the K binary sequences?*

Analysis: Consistent with the analysis in [1], the collision probability can be formulated as follow: Let $P_i$denote the probability that the synchronizing sequence under consideration is the binary sequence *i*. There are a total of *K* sequences with length *L*, where the minimum and maximum values of *K* are 1 and2^L, respectively. Let $P = (P_1\, P_2, \dots, P_K)$ be the collection of probabilities of all the sequences. At least two cases should be considered:

Case 1: All the probabilities P_i are equally likely (i.e., $P_1 = P_2 = \cdots = P_K = \frac{1}{K}$). Then, the probability of no match between *K* binary sequences is given by $P\left(L, K, \frac{1}{k}\right) = \frac{(2^L)!}{\left(K^{2^L}\right)(K^L - K)!}$.

The probability of collision under this case (i.e., equally likely) is:

$$P_c^e = 1 - P\left(L, K, \frac{1}{k}\right) = 1 - \frac{(2^L)!}{\left(K^{2^L}\right)(K^L - K)!} \quad (1)$$

Case 2: When the sequences are *not* equally likely, the probability of collision is generally given by the following formula:

$$P(collision) = 1 - P\,(L, K, P) = 1 - K! \sum_{1 \le w_1 < \cdots < w_k \le n} P(w_1)P(w_2) \dots P(w_K) \quad (2)$$

There are three sub-cases in specifying $P\,(L, K, P)$ in (2):

- **Sub-case 1**: The binary bits in each sequence are independent and identically distributed with probability $P(bit = 1) = \alpha, P(bit = 0) = 1 - \alpha$. Then, $P_i = (1-\alpha)^{N(0,s)}\alpha^{N(1,s)}$, where the number of "zeros" and "ones" are denoted by $N(0, s)$and $N(1, s)$, respectively.
- **Sub-case 2**: The bits in each binary sequence are independent but not identically distributed with probability$P(bit\ i = 1) = \alpha_i, P(bit\ i = 0) = 1 - \alpha_i$. If we define the function $G(b_i) = \alpha_i$ if $b_i = 1$, and $G(b_i) = 1 - \alpha_i$ if $b_i = 0$, then $P_i = \prod_{i=1}^{L} G(b_i)$.
- **Sub-case 3**: The bits in each binary sequence are correlated and their cumulative distribution function is *P*.

In [1], the authors suggested that the Nunnikhoven's approximation [11] be used, since the exact solution of (2) is intractable.

Estimation of collision probability ($L = d_{max}$ in T): Since finding the exact collision probability is complex, we consider the worst case at level *L* with the maximum number of nodes in the synchronizing tree$T$.

Worst-case scenario: From Theorem 1, $d_{\max} = \frac{(n-1)^2 n^2}{2}$. The number of nodes at level $L = d_{max}$in *T* should be between 1 and$2^{d_{max}}$. Since the maximum collision probability

should occur at the level where there is the highest number of nodes in$T$, all the nodes at level $L = d_{max}$should be considered. In the worst case, there exist $2^{d_{max}}$ nodes. Suppose that the set of nodes at the last level is denoted by $V = \{v_1, v_2, \cdots, v_t\}$, where $t = 2^{d_{max}} = 2^{(n-1)^2(n)^2/2}$. Each node in level L is any subset of V. Assuming that each element in the subset is equally likely, the probability of a node v_i such that $v_i$is *singleton* is:

$$P(v_i \in singleton\ at\ L = d_{max}) = p_s = \frac{t}{\sum_{i=1}^{t}\binom{t}{i}} \quad (3)$$

Then, the *average* number of singleton nodes at level $L = d_{max}$ will be $N(a,s) = p_s * 2^{d_{max}} = p_s * 2^{(n-1)^2(n)^2/2}$. Since the necessary condition for the existence of synchronizing sequence is that the node should be singleton, we can now consider only singleton nodes. Let $V^* = \{v_1^*, v_2^*, \dots, v_{N(a,s)}^*\}$ = a set of nodes with a singleton state at level$L = d_{max}$. Assuming that each node in V^* occurs equally likely, we can apply the birthday problem mentioned above. Specifically, the maximum collision probability under the relaxed assumptions can be estimated as follow: Given $V^*$with$N(a,s)$, the probability of collision is:

$$P(collision; L = d_{max}) = P_c(L = d_{max}) = 1 - P(2^{(n-1)^2(n)^2/2}, N(a,s), 1/N(a,s)) \quad (4)$$

Note that (4) denotes the worst-case, meaning that the collision probability will not exceed this number. Due to the extreme calculation, we were unable to run the simulation even for a reasonable value of n. However, we expect that the collision probability is very low. This is due to the fact that the number of nodes in level $L = d_{max}$ is exponentially growing (i.e., 2^L) and the number of possible subset of states (i.e., uncertainty) should grow accordingly. On the other hand, the *ratio* of the expected number of singleton nodes (i.e., the necessary condition) should decrease at level $L = d_{max}$ (i.e., the lowest level in the synchronization tree$T$).

5. Enforcement Method

For a stronger security solution, an additional security countermeasure that can prevent an adversary from attempting to break the key input sequence by a repeated trial search. The proposed method is to add *a self-loop singleton state* to D-FSM. That is, a minor modification to D-FSM is needed and can be done as follow:

```
1)  Create a redundant state S(r) and add it to the original D-FSM;
2)  Add a self loop to S(r); /* δ(S(r), 0) = δ(S(r), 1) = S(r) */
3)  Identify the path Pc(critical path) = S(1)*, S(2)*, …, S(t)*; /* Given a derived
    key sequence Ks = I(1)*I(2)*…I(p)*,Pc can be constructed; S(t)* = Ss, t = p + 1 */
4)  Add a set of edges from S(1)*, S(2)*, …, S(t)* to S(r) such that δ(S(1)*,
    NOT(I(1)*)) = S(r), δ((S2)*, NOT(I(2)*) = S(r), etc.
```

Fig.1. (a) shows the segment of an original D-FSM. The modified system is shown in Fig. 1(b). The modified system becomes a non-deterministic FSM (ND-FSM). However, any ND-FSM can be converted to a deterministic FSM [8]. Despite the number of nodes in the converted D-FSM will increase, the locking/unlocking method presented in the paper can be applied.

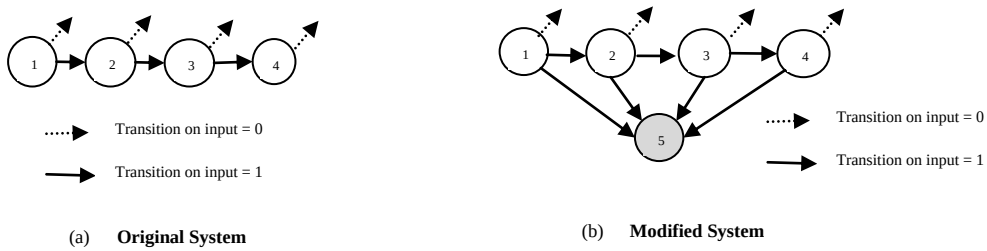

Fig. 1 Segments of original system and modified system where the original system is a deterministic FSM and the modified system is non deterministic FSM.

6. Conclusion

We proposed a sequential locking and unlocking method using a system's logical property of synchronicity. The proposed method can be performed without changing or modifying the original system modeled in D-FSM, as long as the synchronicity property exists in the D-FSM. One possible solution of resolving this limitation (i.e., no existence of synchronicity) is to incorporate a partial scan so that a small subset of final states (i.e., non singleton states) is made to be directly controllable by external input.

7. Acknowledgments

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF1210060.

References

[1] Y. Alkabani, F. Koushanfa and N Kiyavash. Trusted integrated circuits: A nondestructive hidden characteristics extraction approach. *LNCS 5284*, 2008, 102-117.

[2] Y. Alkabani and F. Koushanfa. Active hardware metering for intellectual property protection and security. *USENIX Security Symp*., 2007, 291-306.

[3] Y. Alkabani and F. Koushanfa. Active control and digital rights management of integrated circuit IP cores. *Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, 2007, 227-233.

[4] Y. Alkabani, F. Koushanfa, and M. Potkonjak. Remote activation of ICs for piracy prevention and digital rights management. *Int. Conf. on Computer-Aided Design*, 2007, 674-677.

[5] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. *Int. Symp. Circuits and Systems*, 1989, 1929-1934.

[6] R. Chakraborty and S Bhunia. Hardware protection and authentication through netlist level obfuscation. *Int. Conf. on Computer-Aided Design*, 2008, 674-677.

[7] E. Jung, C-C Hung, S. Choi and M. Yang, An efficient locking and unlocking method of sequential systems, *ACM Research in Applied Computation Symposium*, 2012, 428-433.

[8] Z. Kohavi. Switching and Finite Automata Theory. McGraw-Hill, 1978.

[9] F. Koushanfa. Provably secure active IC metering techniques for piracy avoidance and digital rights managements. *IEEE Trans. on Information Forensics and Security*, vol. 7, no. 1, 2012, 51-63.

[10] F. Koushanfa and G. Qu. Hardware metering. *IEEE Design Automation Conference (DAC)*, 2001, 490-493.

[11] T. Nunnukhoven, A birthday problem solution for nonuniform birthday frequencies, *The American Statistician* 46(4), 1992, 270-274.

[12] C. Pixley, S. Jeong and G. Hatchtel. Exact Calculation of Synchronizing Sequences Based on Binary Decision Diagrams. *IEEE Trans. on Computer-Aided Design of Circuits and Systems*, Vol. 13, No. 8, 1994, 1024- 1034.

[13] A. Oliveira. Techniques for the creation of digital watermarks in sequential circuit Design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*. Vol. 20, No. 9, 2001, 1101-1117.

[14] M. Rodoper, W. Trappe and E. Jung, An IBC and Certificate Based Hybrid Approach to WiMAX Security, Journal of Communications and Networks (JCN), Special Issue on Secure Wireless Networking, Vol. 11, No. 6, 2009, 615-625.

[15] M. Tehranipoor and C. Wang (editors). Introduction to Hardware Security and Trust. Springer, 2011.

[16] I. Torungoglu and E. Charbon. Watermarking-based copyright protection of sequential functions. *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 3, Feb. 2000, 434-440.

[17] L. Yuan and G. Qu. Information hiding in finite state machine. *Information Hiding Conference (IHC)*, Springer, 2004, 340-354.

*Corresponding author: Edward Jung, Ph.D.
Department of Computer Science and Software Engineering
Southern Polytechnic State University
1100 South Marietta Parkway
Marietta, GA 30060, USA
E-mail: ejung@spsu.edu