Security Fusion: A New Security Architecture for Resource-Constrained Environments

Suku Nair, Subil Abraham, Omar Al Ibrahim
HACNET Labs
Computer Science and Engineering
Southern Methodist University

Abstract

There is a huge demand for wireless sensors and RFID tags for remote surveillance and tracking. However, in order for such technologies to gain wide acceptance in industry, there needs to be strong security integrated into them. Traditional cryptographic schemes are infeasible due to hardware, computation, and power constraints. To that end, we introduce a new security paradigm, namely *security fusion*. In this approach, strong security properties are synthesized from weaker point-to-point properties, thereby minimizing the resource requirements at each node without compromising the system-level security. In this paper, we describe the concept of security fusion and give a motivation example using finite state machines.

1 Introduction

Implementing security in sensors and RFID have posed great challenges given the severe limitations in processing power, memory and computational cycles. For instance, a passive RFID tag that complies with the EPC C1G2 standard [1] has a baseband of 7500 Gate Equivalence (GE) out of which 200-2000 gates are reserved for security functionalities [14] [16]. The chip area for such tags is approximately 0.8mm^2 and the power consumption is roughly 30 μW with EEPROM programming [8]. Wireless sensors are also limited in processing speed, storage capacity and power. Popular sensor devices have RAM sizes ranging from 4 KB on the Mica2 platform [4] to 10KB on the TelosB platform [5]. The storage capacity is usually between 4KB and 512KB and may come incorporated with Bluetooth, Chipcorn CC100, or IEEE 802.15.4 compliant transceivers. Today, devices such as micro-sensors and nano-sensors are increasing in popularity over conventional sensors. These sensors are known for their extremely small size, low cost, and high sensitivity. Therefore, supporting cryptography has become a perennial problem given these characteristics. Table 1 (ref. [15]) provides a comparison between various cryptographic algorithms with respect to the number of cycles, gates, and average power consumption. Generally speaking, the complexity involved with traditional cryptography is formidable for low-cost sensors and RFID. Despite Moore's Law, sensors and RFID will remain resource-constrained because of their design. On top of that, new advancements in process technologies will be insufficient for tighter constraints in the new generations of these devices.

Table 1: Cost analysis of various algorithms

Algorithm	Key Bits	Plaintext	Cycles	GE	Power (μW)
AES	128	128	1016	3595	8.15
TEA	128	64	64	2355	12.34
SHA-1	*L*	192	405	4276	26.73
LFSR	32	64	92	685	0.1582
DES	56	64	144	2309	2.14
ECC	113	*L*	195159	10k	*L*

L denotes number of bits

On the other extreme, lightweight cryptographic algorithms [7] [10] [11] [15] are also undesirable because of their weak resistance to cryptanalysis and other attacks. Consequently, there has always been this tradeoff between the cryptographic strength and the resource requirements of the device. This tradeoff is quite natural when using a point-to-point security model. In other words, in a point-to-point security model, system security is always constrained by the device capabilities, and depending on the application, it can be overkill to include heavy duty encryption.

In this paper, we describe a new paradigm in security for resource-constrained environments. Rather than lending to traditional point-to-point approaches, the goal here is to focus on aggregating lightweight security properties from multiple nodes to build a strong in-network security platform. We coin the term *security fusion* to refer to this concept of aggregating less stringent point-to-point security properties between nodes to construct strong system-level security properties. The principle of the security fusion approach is that nodes do not provide all the protection by themselves, since they only implement a simple lightweight primitive, but unlike traditional schemes, these primitives are fusible to provide greater security, much like the thin strands of a thick bulk rope.

Security fusion is suitable for aggregation-based environments with large numbers of communicating nodes. Here, we mention a few applications in real-time systems that satisfy these characteristics.

- *Low-cost RFID*: Security has become a rising concern for RFID mainly because scanning and replicating tags require little money or expertise. While cryptographic RFID purports to offer seemingly strong protection, they are too expensive to be used in large quantities. To overcome factors like power budget, computational latency, and cost, security fusion enables simple mechanisms to be incorporated in RFID tags, which can be consolidated quickly by the middleware. With multi-tag RFID systems [2], security fusion becomes more effective.

- *Sensor networks*: Sensor networks consist of autonomous nodes that collectively obtain measurements from the environment to improve system-level decisions. Due to the aggregate nature of data in sensor networks, it is natural to define security through a fusion model that protects the monitoring, tracking, and controlling functions of an application. Security fusion has a lot of potential in sensor networks. Using security fusion, it is possible to collate multiple sensor read-outs to reach a globally authentic decision.

- *Embedded systems*: Another potential application for security fusion is to preserve the component-level integrity of embedded systems. For instance, recent generations of smart phones consist of various interconnected components such as Bluetooth radio, WiFi radio, flash memory, and Codec chip. These components are manufactured by different suppliers and then integrated into a single platform. In the event of counterfeiting, non-genuine components can be detected through security fusion using an aggregate verification check of the platform.

2 Previous Work

There are several topics which span the discussion on previous work. Here we briefly review some of the work in securing resource-constrained environments.

A couple of security architectures have been proposed to protect sensor networks from various attacks. Perrig et al. [11] introduced an architecture called SPINS which consists of two building blocks: SNEP - a security protocol that employs symmetric cryptography using RC5 for encryption and Message Authentication Code (MAC) for integrity, as well as μTESLA - which provides authentication for data broadcast. The SNEP protocol is designed for resource-constrained sensor networks. To encrypt messages in the SNEP protocol, each sensor node shares a master key with a base station, from which it derives one-time encryption keys to be fed to RC5. The other building block of SPIN, μTESLA, provides authenticated broadcast for sensor networks. μTESLA was designed based on the standard TESLA protocol, but is developed to address limited computing environments.

Another security architecture that has recieved a swirl of attention in the past few years is TinySec. TinySec, proposed by Wagner et al. [9], is a link layer security architecture for micro-sensor networks which is tightly coupled with Berkeley TinyOS. The motivation for TinySec was to increase the network resistance against denial-of-service attacks by enabling early detection of unauthorized packets in the network, and preventing them when they are first injected as opposed to waiting for packets to route to the destination, thus saving energy and bandwidth.

3 Security Fusion

The general approach of the security fusion paradigm is to introduce lightweight elements at individual nodes that provide a strong aggregation of security. In this section, we describe some of the theoretical foundations for these concepts. We implicitly discuss the concept of security fusion in-line with the problems of achieving authentication in a decision system. However, the broad motivation of the framework extends to other security attributes including integrity and confidentiality. It is important to note that security fusion is not just simply a framework for secure consensus among the nodes, but rather it extends to a fusion of security as a property. In contrast to approaches like trusting crowded-sourced nodes, the intended objective is to develop a centeralized challenge-response system that aggregates security characteristics from multiple nodes. In this section, we present an instantiation of the security fusion concept based on the model of a finite state machine. But first, we begin by describing the physical architecture and the attack model.

3.1 Physical Architecture

The physical architecture of the security fusion framework consists of a typical centralized cluster-based network which is comprised of a large number of nodes (Figure 1). Each node is equipped with three components: computation, storage, and a communication. The physical architecture also consists of a group of readers acting as cluster heads. The communication model is limited to a challenge-response interaction between the reader and the nodes. Thus, minimal communication is performed in these nodes and no delegation of messages is necessary. In the setup stage of the architecture, every node shares a secret key with the system. Further, the keys are assumed to be distributed to the nodes before deployment.

3.2 Attack Model

In the security fusion architecture, we consider an attacker that is adaptive across space and time. Given these characteristics, we identified a number of attacks that can be launched:

- *Malicious read*: With a rogue reader in possession, an attacker may attempt to read out from a node. By interrogating repeatedly, he may collect packets to derive the secret.
- *Brute-force attacks*: An attacker may expand his attack by exhausting all possible responses for a given node. He may listen passively to a communication or actively read out to collect packets.
- *Replay attacks*: An attacker may collect packets and replay them back to the system at a later time. The success of the attack will depend on whether an attacker can replay the correct responses to fool the system.
- *Physical capture*: A determined attacker may go as far as capturing a node to extract the shared secret. We assume that compromised nodes will not reveal the secrets of other nodes as no two nodes will share the same secret.
- *Replication*: In an attempt to spoof the responses, an attacker may replicate a node and insert it into the network hoping to disrupt the system.

3.3 Security Fusion Example: Finite State Machines

In this paper, we present an example of the fusion concept based on the model of a finite state machine. Finite state machines serve as a basic computational model that

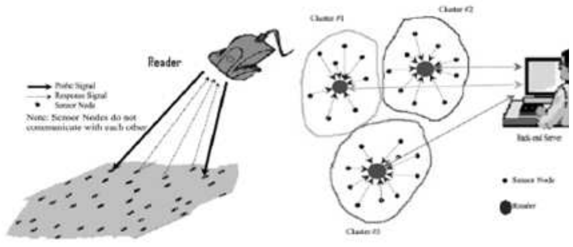

Figure 1: Physical architecture

can help build systems with secure global properties. In this example, each node will have a unique state machine that is shared as a secret key with the backend system. The underlying state machine governs the output pattern of the node by following certain transition rules. Furthermore, each state of the machine is assigned a disjoint set of uniquely randomly-generated values, or *pseudonyms*, which represent the outputs. The input to the state machine is internally derived and can be used to express something like sensor detection results (as boolean logic detection/ non-detection).

In the execution of the state machine, the nodes communicate the boolean logic by randomly selecting a pseudonym from a set of possible pseudonyms assigned to the current state before the state machine transitions to the next state. Then, the outputs are sent to the backend system and decoded to reach a consensus. The security properties derived from this consensus are hard, meaning that they are non-trivially amplified with increased number of nodes.

In the analysis of the fusion methodology, the state machine stored in each node is deterministic. However by using the pseudonyms, the obfuscation converts the deterministic state machine (DFA) into one which is larger in size and non-deterministic (NFA) from an attacker's perspective. For a certain selection, we can show that the NFAs are cumbersome to reduce by minimization algorithms, especially when scaled to a large number of nodes. These non-deterministic properties are collated to build a strong fusion property for security.

3.4 Description of the State Machine

A state machine can be described by ”transition” and ”output” rules. Transition rules define the mappings from the current state to the next state given the input value. Output rules define possible values emitted by a node for a given state and transition. Let us assume we have a state machine with n states $(s_1, s_2, ...s_n)$. Since we are considering a state machine with a single-bit input, we have two transitions per state: ”0” and ”1”.

In the following, we describe the transition and output rules using a Mealy model [6].
Transition rules: (Current state, Input) $\rightarrow$ Next state

- $(s_i, 0) \rightarrow s_j$
- $(s_i, 1) \rightarrow s_v$, where $(0 \leq i, j, v \leq n)$

Output rules: (Current state, Input) $\rightarrow$ Output

- $(s_i, 0) \rightarrow k_j$
- $(s_i, 1) \rightarrow k_v$, where $k_j \neq k_v$

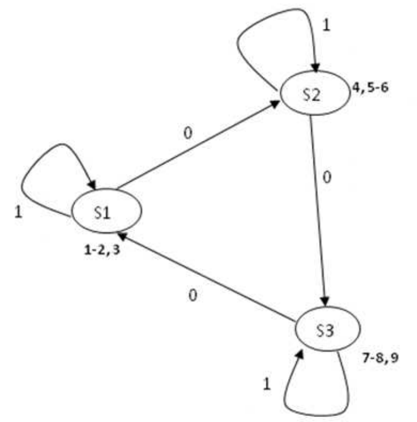

Figure 2: State diagram

Consider assigning k unique pseudonyms to each state in the state machine, of which $p(1 \leq p \leq k)$ pseudonyms are used to represent a transition on (0) and $q(q = k - p)$ pseudonyms are used to represent a transition on (1). In the execution of the state machine, a node randomly selects a pseudonym from either set p or q assigned to the current state. Then it transmits the pseudonym and transitions to the next state using the transition rules. As an example, consider a 3-state finite state machine

1. n=3 $\{s_1, s_2, s_3\}$
2. k=3 [Each state is assigned a set of 3 numbers of which $p(1 <= p < k)$ numbers may be used to represent (0) and $q = k - p$ numbers may be used to represent a (1).]
3. The total set of numbers available for the 3-finite state machine = nk = 9
4. Each state $\{s_1, s_2, s_3\}$ will have k=3 numbers assigned to it.

Figure 2 illustrates the state diagram and Table 2 shows an example of a pseudonym assignment for the state diagram. In the next section, we describe the protocol and authentication procedure using the state machine model.

Table 2: Pseudonym assignment example

States	p Transition on 0	q Transition on 1
s_1	1 or 2	3
s_2	4	5 or 6
s_3	7 or 8	9

3.5 Security Protocol

In the security protocol, each node is associated with a unique state machine which is shared as a secret key with the backend system. After each interrogation from the reader, nodes may change their states according to the transition rules defined prior to deployment. The reader verifies the legitimacy of a node by checking the expected current state and confirming the correct application of the rules.
A basic protocol for this interaction is depicted below.
Denote N: node, R: reader

1. `R` $\rightarrow N$: `Send read query`
2. `N: Obtain <bit value> (0/1)`
3. `N` $\rightarrow$ `R: N moves to the next state based on <bit value> and outputs a pseudonym from set p or q`
4. `R resolves N's output and re-syncs`

It should be noted that even though the values read from each node are similar to a point-to-point security channel, no security properties are derived from just one value. Rather, a large number of the read values are collated to derive the security fusion properties.

In order to properly authenticate a node, the backend system needs to maintain a copy of the state machines associated with all the nodes in the network. These state machines are indexed by the corresponding node id. When interrogated by a reader, a node responds with a pseudonym and possibly with its id. This value of the pseudonym depends on the state machine rules defined prior to deployment. Since the system can obtain the node's state and the pseudonyms assigned for that state, it could simply authenticate a node by the pseudonym. The backend system can also deduce the original value of the node using the mapping between the pseudonyms and the actual information. After getting the response, the system will also update its copy of the state machine corresponding to that node with the new state value. In the back-end server, the system assigns a unique node id and stores flag bits to track the current state of the node. For every state and machine input, the server also stores the next state and pseudonym set. Whenever the server receives a response, it checks if the value matches any of the elements in the pseudonym set of the node's current execution. Otherwise, the response is rejected.

3.6 State Machine Selection

Because these state machines vary in their security strength, a stringent criteria is needed to select machines with strong properties. In this section, we describe the criteria for selecting state machines to provide strong protection for the overall architecture. The selection is based on the following factors:

1. *State reachability*: Here we mean complete reachability in which every state should have a path to every other state through one or more transitions. From a security perspective, a machine with unreachable states is analogous to a secret key with unused bits. Therefore, any state machine with unreachable states or with a potential of being unreachable should be discarded after a number of transitions.

2. *State complexity*: The generated state machine is stored as a deterministic automaton. Using statistical properties, an attacker can only simulate a non-deterministic version of the machine. Converting a non-deterministic machine to deterministic may lead to exponential blow up in the number of states. The explosion in state space does not occur for all non-deterministic machines. Therefore, choosing structures that lead to explotion in the state space is the key for increasing the complexity. In [13], Sutner proposed a class of non-deterministic machines that satisfy the desired property.

3. *Pseudonym randomness*: Pseudonyms need to be cryptographically secure so that an adversary cannot predict the values. Since the numbers are generated prior to assignment, they can be pre-computed using crypto classes of PRNG algorithms. These algorithms either use stream or block ciphers. Examples of current implementations include Blum Blum Shub, ANSI X9.17, and CryptGenRandom.

4. *Pattern randomness*: Pattern randomness highly depends on the selection randomness of the pseudonyms in every state. We propose to use different randomness tests to choose machines with higher output distribution. Using a black box approach, a random input stream of 1's and 0's (which can be crypto generated) is produced to evaluate the distribution of the pseudonyms generated by the state machine. A basic statistical test is to analyze the frequency of the outputs to measure the distributed equiprobability. Other tests based on chi-square [3], such as serial and poker tests, may be conducted. These tests evaluate the frequency for a sequence of numbers of some length, e.g. sequence of every five numbers at a time.

3.7 Security Analysis

In this section, we evaluate the state machine approach based on the attack model described previously in Section 3.2. Since the state machines do not reveal the actual state mappings, an attacker can only collect the pseudonyms assigned to the states. In other words, the pseudonyms are used to hide the internal state of the machine, so if an attacker is going to reconstruct the state machine, he will need to collect successive pseudonyms by observing the output pattern.

3.7.1 State machine complexity

One simple way an attacker has of reconstructing the state machine is to treat every pseudonym as a state value. Based on that, the state machine constructed by the attacker is an NFA with nk states and nk^2 transitions for every node, where n is the number of states in the encoded machine and k is the number of pseudonyms assigned to each state. It can be shown from formal theory, that the NFA derived by the attacker is equivalent in behavior to the DFA encoded inside the node. Though an attacker is able to reconstruct a state machine to model the readout patterns, the size of the state machine constructed by the attacker is larger than the original machine (with nk states and nk^2 transitions for every node). This is an artifact of the non-determinism introduced by hiding the internal state of the machine. The node state machine is modeled by assigning the pseudonyms to a given pair of state and input, where as the state machine derived by the attacker is modeled by using the pseudonyms as the state values.

3.7.2 State machine minimization

With a large number of nodes, the complexity for storing the NFAs is formidable for the attacker because it requires vast amount of memory and processing capabilities at his disposal. Therefore, in order to scale the attack, the attacker needs to resort to a minimization algorithm in order to reduce the size of the state machines produced by the NFAs. One of the classical algorithms to minimize these state machines is an algorithm proposed by Hopcroft's [12]. Hopcroft's Algorithm is a minimization procedure which transforms a given state machine into an equivalent state machine with minimum number of states.

Based on Hopcroft's Algorithm, it takes $mlog(m)$ steps to minimize a deterministic finite state machine (DFA) with *m*-states. Hence, if the state machine derived by the attacker was deterministic, it would take $(nk)log(nk)$ steps to minimize it. However since the state machine derived by the attacker is non-deterministic, we cannot directly apply the algorithm. Instead, the attacker needs to convert the NFA to a DFA first before applying the algorithm. Unfortunately for the attacker, there exists some NFA that leads to a state blowup when converted to a DFA.

- NFA-DFA Blowup: Given a natural number *m*, there exists an *m*-state NFA whose minimal equivalent DFA has $\geq 2^{m-1}$ states.

This is a long-known result in automata theory and in our case, we have $m = nk$ states. Because of the state blowup in the conversion, the overall problem becomes *NP-hard* [12] [13]. That is, converting an NFA to a DFA can cause an exponential increase in the number of states which makes the problem of minimization *NP-hard* whenever you do not start with a DFA. In this scheme, we realize that not all NFAs will produce an exponential blowup. However, in the earlier subsection, we described a selection criteria required to guarantee this property. Based on this property, we can see that the security complexity is not simply a linear correlation of the number of states. Further, it is possible that an attacker handles state explosions for one or few NFAs, but in order to make the problem intractable for the attacker, the system needs to have a large number of nodes. Since the number of states at each node is small by design, an attacker may be able to exploit them, but a large number of them would be exponentially complex to reduce because of the security fusion property. The question as to how many nodes are needed to have a practical sense of security will depend on the underlying assumptions on the memory and computational resources available to the attacker and the size of the individual state machines.

4 Conclusion

In this paper, we have introduced the concept of security fusion, a new security architecture for resource-constrained environments. Since standard cryptographic solutions are likely to be impractical for such an infrastructure, our architecture employs lightweight techniques to circumvent large-scale cloning and replay attacks for system-level protection. In this paper, we described a model based on finite state machines with some details on the architecture. Analysis of the security properties demonstrated that the security complexity is hard with respect to the number of states. These properties can be leveraged to build the theory and practice of security services in resource-constrained environments. For future work, we plan to extend the architecture and apply other fusion techniques for improved security properties and overall complexity.

References

[1] EPC Global. http://www.epcglobalinc.org/home/.

[2] Bolotnyy, L., Krize, S., and Robins, G. The Practicality of Multi-Tag RFID Systems. In Proc. International Workshop on RFID Technology - Concepts, Applications, Challenges (IWRT).

[3] Chernoff, L. The Use of Maximum Likelihood Estimates in X2 Tests for Goodness-of-Fit. *The Annals of Mathematical Statistics 25* (1954), 579–586.

[4] Crossbow. Mica2 Datasheet, Crossbow. http://www.xbow.com/.

[5] Crossbow. TelosB Datasheet, Crossbow. http://www.xbow.com/.

[6] Hennie, F. *Finite-State Models For Logical Machines*, 4th edition ed. John Wiley & Sons, New York, USA, 1968.

[7] Huang, C., and Chien, H. Security of Ultra-Lightweight RFID Authentication Protocols and Its Improvements. ACM SIGOPS Operating Systems Review, ACM, pp. 83–86.

[8] Kang, H., Hong, S., Song, Y., Sung, M., Choi, B., Chung, J., and Lee, J. High Security FeRAM-Based EPC C1G2 UHF (860 MHz-960 MHz) Passive RFID Tag Chip. *ETRI Journal 30*, 6 (2008), 826–832.

[9] Karlof, C., Sastry, N., and Wagner, D. TinySec: A Link Layer Security Architecture for Sensor Networks. Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys).

[10] Needham, R., and Wheeler, D. TEA Extensions. Tech. rep., Cambridge University, 1997.

[11] Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, J. SPINS: Security Protocols For Sensor Networks. Proceedings of Seventh Annual International Conference on Mobile Computing and Networkings.

[12] Skiena, S. *The Algorithm Design Manual*. Springer-Verlag, New York,, 1998.

[13] Sutner, K. The Size of Power Automata. *Theoretical Computer Science 295*, 1-3 (2003), 371–386.

[14] Wang, C.-H., and Huang, C.-W. A Collaborative Network Security Platform in P2P Networks. NISS '09. International Conference on, pp. 1251–1256.

[15] Wang, J. R&D of Gen2 Tag with Enhanced Security Mechanism. Tech. rep., Auto-ID Lab, 2009.

[16] Wheeler, D., and Needham, R. TEA, a Tiny Encryption Algorithm. Fast Software Encryption: Second International Workshop, Lecture Notes in Computer Science, p. 363366.