Secure Modules in TinyOS

Saeed Aldawsari
Department of Computer Science & Engineering
University of Bridgeport
saldawsa@my.bridgeport.edu

Ahmed Alamri
Department of Computer Science & Engineering
University of Bridgeport
aalamri@my.bridgeport.edu

***Abstract*— In this paper, we discuss TinyOS as a flexible operating system that is suitable for wireless sensor networks. It is a powerful tool that is capable of forming a strong component of intelligent systems. Similarly, sensor networks are composed of accurate, low levels of power nodes that carry out simultaneous, reactive programs that operate within the limitations of power and memory. As a solution, we integrate components of the TinyOS with TinyHash or modules for better operations. We also present more data about four components based on our proposed protocol, which includes hash function, module hash table, base station, and algorithm chart.**

***Keywords*— TinyOS, wireless sensor network, WSN, hash function, module hash table, base station, and algorithm chart.**

I. INTRODUCTION

A sensor network is referred to as a network for the succeeding generations as evidenced by numerous studies. Today, many applications are now using the sensor network for various purposes, such as collecting and managing environment data, military service, emergency medical system, tracing and managing of goods, and among different fields across the globe. In relation, the sensor network has the ability to organize wireless networks in addition to sensor nodes. Likewise, these sensor nodes have certain limitations with regards to bandwidth, electricity consumption, memory, and ability to calculate. In general, sensor nodes can efficiently transmit through short-range broadcast communication. However, a wireless sensor network (WSN) usually is at risk for security threats exposure because there is the possibility for eavesdroppers, insertion of hateful messages, and modification of communication messages. In order to prevent such threats, communication data encryption and common authentication between the sensor nodes are necessary. On the other hand, majority of the risks concerning online security originate from weakly programming code. Therefore, it is a must to prevent the occurrence of such events by acquiring trustworthy hardware and software. Since software is prone to hacking activities, a fool-proof alert system is a must. Similarly, this alarm system should immediately notify the user of unauthorized alterations to prevent security threats.

II. BACKGROUND

1 – TinyOS:

TinyOS are systems that dominate majority of programming wireless sensor network devices. A TinyOS application is defined as a series of mechanisms that offer limited runtime support. A programmer usually documents a number of custom components and links that are derived from the TinyOS library. These components are written in nesC, which is a dialect of C with component extensions, standard programming, and concurrency. Afterwards, the nesC compiler interprets component assemblies into a monolithic C program that is gathered and enhanced by GCC.

In order to save energy, a TinyOS application usually sleeps or functions with low duty cycle. Likewise, applications are commonly interrupted and follow a limited two-level concurrency model. Many codes operate in tasks and without a preventive schedule. However, interruptions have the ability to obstruct activities other than atomic sections, which are executed by stopping interrupts.

Moreover, there are numerous reasons why TinyOS is famous for. One of the reasons is because nesC is very comparable to C, which is the embedded software's principal language. Such custom is significant for the user to adopt, however poses various problems related to C code. Another reason why TinyOS remains popular is because it offers a wide array of prepared components of the library. This allows the programmer to save their work for specific activities. Furthermore, the nesC compiler is equipped with race condition detector, which aids the developers in preventing concurrency viruses or bugs. Lastly, the popularity of TinyOS can be attributed to its design and overall static resource allocation model that supports programmers to easily discover dynamic allocation bugs. Additionally, static allocation aids the programmer maintain time and space operating costs to be at its lowest by surpassing the need for bookkeeping.

Therefore, we selected TinyOS as the foundation of our method or approach for reliable software development directed towards sensor networks. We have to work with a legacy language as well as a legacy code. Nonetheless, we discovered the advantage of using sensor network applications' properties along with TinyOS, in addition to its

concurrency and allocation models so that Safe TinyOS can be practically executed.

2 – TinyHASH:

Recently, there has been great interest over security infrastructure made especially for sensor networks. Among these proposals was made by Chris Kalof, and was called TinySec. Another example is David Wagner's Link layer security structural design, which focused on sensor network-related issues like energy, computation capability, bandwidth, among others. Nevertheless, TinySec utilizes CBC_mode of encryption and CBCMAC for the purpose of authenticating via SkipJack Block Cipher. Today, TinySec is integrated in the TinyOS to strengthen sensor network security. In this paper, we present TinyHash according to the generalized hash algorithm. Thus, TinyHash functions as the module used to substitute authentication parts and the veracity of TinySec. Meanwhile, TinyHash components are developed the same way as TinySec in order to become compatible. Also, TinyHash instigates the HMAC component for authenticity and the Digest component for the accuracy of the messages. In addition, we describe common interfaces used for service related to hash algorithm.

III. THE PROPOSED PROTOCOL

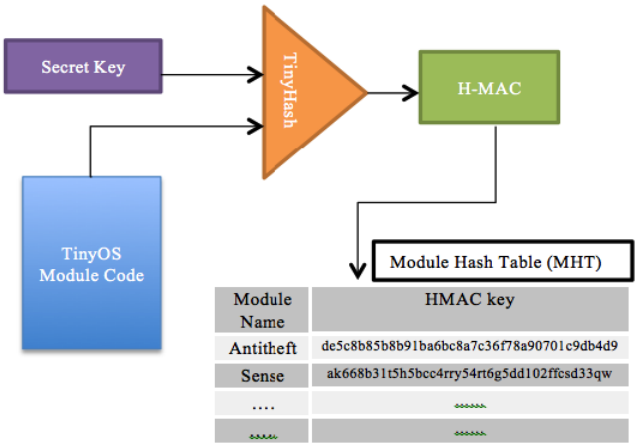


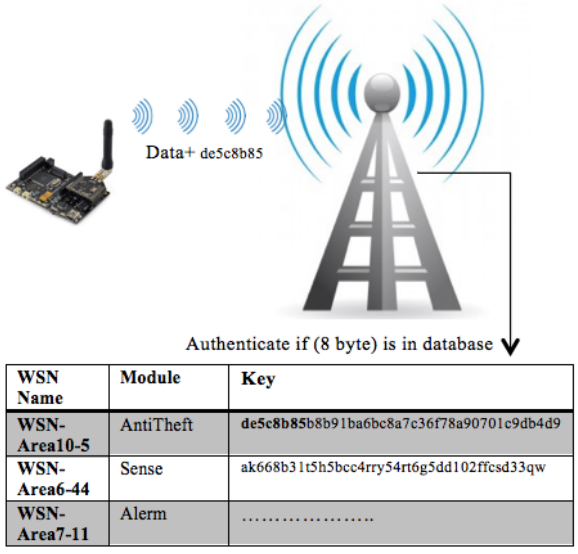
Figure-A

Our procedure is to use a module code that we will upload on an embedded memory, that doesn't involve unauthorized modifications. In order to accomplish this, we used both TinyHash module and Modules Hash Table (MHT).

In addition, our protocol is designed to hash or include a particular module code through TinyHash module, and achieve the outcome as the answer to H-MAC. Afterwards, we stored it in Modules Hash Table (MHT), which is a memory space in WSN where the HMAC Keys and function names are located. Out presentation is based on this protocol plan that is reliant on TinyOS.

As per procedure, we offered a technique to aid in reducing the consumption of battery life and hashing processes. Such method requires calculating hash at a specific period of time. For example, it can be set or scheduled four times or numerous times every day with lesser hashing to limit the process, depending on the security problem presented.

Moreover, we offer another method that will help reduce data packet whenever TinyOS will send information to the base station. The MHT will give the hash key module's virtual division in five parts, and out technique is to select a part of the hash key arbitrarily. Afterwards, it will be attached to the sensed information and sent to the base station in the end. Below is an example whereby TinyOS sends 8 bytes of hash key instead of 32 bytes. The main goal of 8 bytes sent by TinyOS to the base station is to authenticate unauthorized modification in module code. After the 8 bytes is received, the base station will receive the information sent and will determine its authenticity or lack thereof of the data received. By this time, the base station will be protected against unauthorized manipulation attempts with the module code. See figure-B.
Also, we offer more information about this through the four components of our proposed protocol: (A) Hash Function; (B) Module Hash Table; (C) base station; and (D) Algorithm Chart.



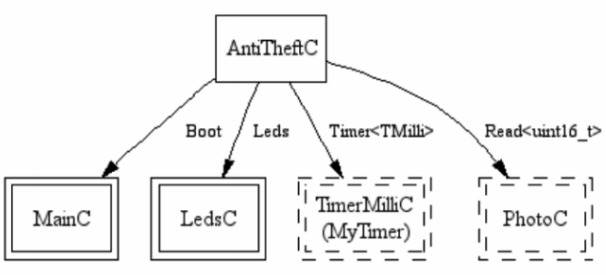
WSN Name	Module	Key
WSN-Area10-5	AntiTheft	**de5c8b85**b8b91ba6bc8a7c36f78a90701c9db4d9
WSN-Area6-44	Sense	ak668b31t5h5bcc4rry54rt6g5dd102ffcsd33qw
WSN-Area7-11	Alerm	……

Figure-B

A. Hash Function

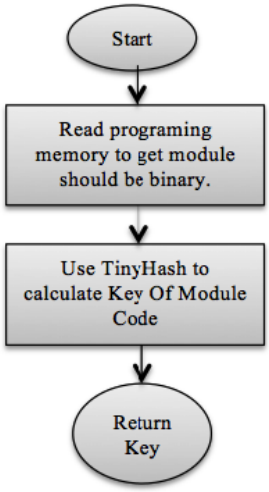
TinyHash applies the H-MAC scheme for authentication purposes, and applies SHA-1 hash algorithm for digesting messages. As Figure-A shows, we have two inputs, one of which is the secret key while the other is the TinyOS module code. After both inputs are entered to the TinyHash module,

the module will then calculate the H-MAC and give the result as 32 bytes.
Below is an example of TinyOS' AntiTheft module:




```
module AntiTheftRootC
{
 uses
 {
 interface Boot;
 interface SplitControl as SerialControl;
 interface SplitControl as RadioControl;
 interface LowPowerListening;
 interface DisseminationUpdate<settings_t> as SettingsUpdate;
 interface Receive as SettingsReceive;
 interface StdControl as CollectionControl;
 interface StdControl as DisseminationControl;
 interface RootControl;
 interface Receive as AlertsReceive;
 interface AMSend as AlertsForward;
 interface Leds;
 }
}
implementation
{
 event void Boot.booted()
 {
 call SerialControl.start();
 call RadioControl.start();
…..
……
……
```

As the above partial module code suggests, the module conformed to the Hex file and is uploaded as binary to embedded Memory (RAM) in WSN. After this, we begin to operate our algorithm in order to calculate hash keys.



Meanwhile, here is the execution times and TelOS size achieved after using the TinyHash module:

Algorithms	Time	Size of RAM
SHA-1	35 ms	140 Bytes
H-MAC	71 ms	165 Bytes

B. Module Hash Table (MHT)

Through the utilization of the TinyOS, we will stock the H-MAC key and the module stored in the memory after it has been hashed. See Figure-A. After the MHT receives the hash key, the option to divide the key into 5 parts will be available. This is so that the TinyOS can randomly select one of these 5 parts to send to the base station.

De5c8b85	B8b91ba	6bc8a7c3	6f78a907	01c9db4d9

An example is when TinyOS sends information that randomly attaches the key 6bc8a7c3 and sends it. Afterwards, it will select another key randomly. The table presented above presents the allocated free memory space in the microcontroller. In addition, it will restart when the scheduled time in the TinyHash arrives.

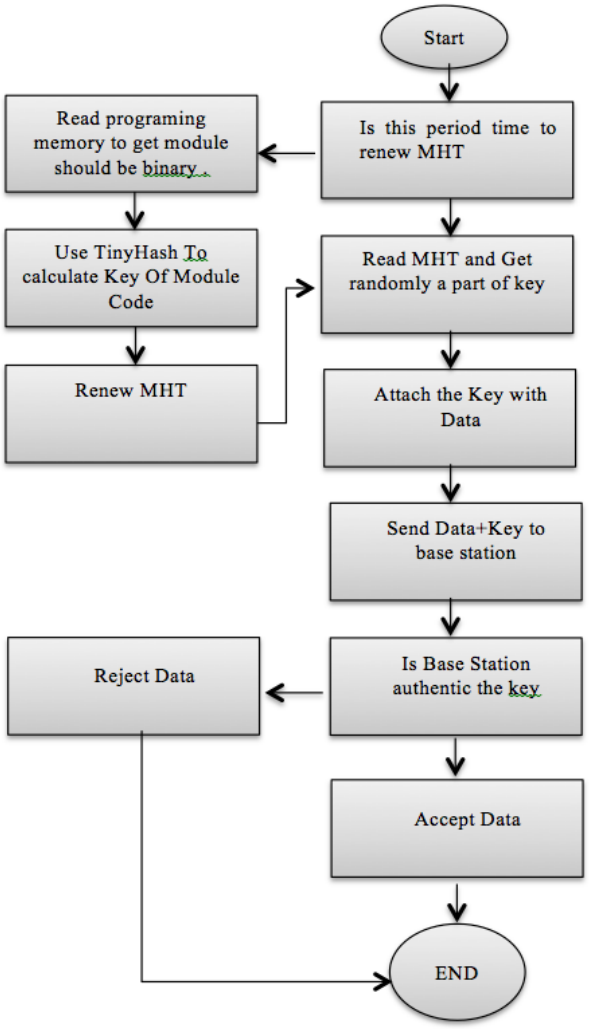
C. Base Station:

The base station refers to the location that receives all the data originating from the WSN. It is powerful and offers more resources, and is saved in a safe location. Among the resources available, it is where overall WSN information is stored and located. It has database involves 3 columns: the WSN name; the module; and the key. First, the WSN name contains the WSN number and the area number. Meanwhile, the module is the same as the WSN modules. Lastly, the key is the module's hashed key. In the end, the database or base station has all the WSN highly sensitive data prior to the time the WSBN is used. An example of this is as follows: WSNArea10-5, Antitheft, de5c8b85b8b91ba6bc8a7c36f78a90701c9db4d9. As this code suggests, the WSN is reserved in Area 10, which has a number 5, antitheft module, and a hashed key. After the base station received the data containing the 8 bytes, it will send an inquiry to the database to authenticate the data that was received. If it is proven that the data received is accurate and correct, the base station will then accept the data from the WSN. Otherwise, it will be rejected.

An example of a query is found below:

"SELECT wsnName,module,key FROM wsndb WHERE Key LIKE %'de5c8b85'% ".

D. Algorithm Chart:



III. CONCLUSION

In conclusion, TinyOS is a crucial operating system that is timely with present security risks like hacking. By using a module code, unauthorized activities and alterations will become preventable via the use of TinyHash module and modules hash table. Similarly, the protocol we designed, which includes hash function, module hash table, base station, and algorithm chart, is a feasible method to help reduce battery consumption (drainage) and assist in the hashing process that could avoid security issues.

TinyOS is a promising operating system that is driven by events and concurrency structured following a TinyHash module. It not only helps manage concurrency better but also proves to be a significant protocol targeting software and its components. In the end, it must be remembered that TinyOS is a continuous development of language tools, optimization, software protocols, and algorithm.

REFERENCES

1. Cooprider, N., et al. *Efficient memory safety for TinyOS*. in *Proceedings of the 5th international conference on Embedded networked sensor systems*. 2007. ACM.

2. Kasture, A., A. Raut, and S. Thool. *Visualization of Wireless Sensor Network by a Java Framework for Security in Defense Surveillance*. in *Electronic Systems, Signal Processing and Computing Technologies (ICESC), 2014 International Conference on*. 2014. IEEE.

3. Kazienko, J.F., et al., *Practical evaluation of a secure key-distribution and storage scheme for wireless sensor networks using TinyOS.* CLEI Electronic Journal, 2011. **14**(1): p. 8-8.

4. Lee, H., Y. Choi, and H. Kim. *Implementation of tinyhash based on hash algorithm for sensor network.* in *Proceedings of world academy of science, engineering and technology*. 2005. Citeseer.

5. Lee, J., K. Kapitanova, and S.H. Son, *The price of security in wireless sensor networks.* Computer Networks, 2010. **54**(17): p. 2967-2978.

6. Levis, P., et al., *TinyOS: An operating system for sensor networks*, in *Ambient intelligence*. 2005, Springer. p. 115-148.

7. Xiaoliang, Z. and C. Yunfang. *Research of wireless injection attacks based on TinyOS*. in *Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on*. 2013.