Improving Smart Grid Authentication using Merkle Trees

Melesio Calderón Muñoz[1], Dr. Melody Moh[2], Dr. Teng-Sheng Moh[2]
[1]*Cupertino Electric Inc., San Jose, CA*
[2]*Department of Computer Science, San José State University, CA*

Abstract—The electrical power grid forms the functional foundation of our modern societies, but in the near future our aging electrical infrastructure will not be able to keep pace with our demands. As a result, nations worldwide have started to convert their power grids into smart grids that will have improved communication and control systems. A smart grid will be better able to incorporate new forms of energy generation as well as be self-healing and more reliable. This paper investigates a threat to wireless communication networks from a fully realized quantum computer, and provides a means to avoid this problem in smart grid domains. We discuss and compare the security aspects, the complexities and the performance of authentication using public-key cryptography and using Merkel Trees. As a result, we argue for the use of Merkle Trees as opposed to public key encryption for authentication of devices in wireless mesh networks (WMN) used in smart grid applications.

***Index Terms*—Authentication, Merkle Trees, public-key cryptography, quantum-computer attacks, RSA, smart grid.**

I. INTRODUCTION

By 2050 worldwide consumption of electricity is expected to triple [1]. The electrical power grid has served humanity well to now, but as we seek new ways to generate energy and improve efficiency, we find that the existing grid will not be able to meet our needs. In addition power grids are still susceptible to large-scale outages that can affect millions of people [2]. These are the motivations for the creation of an "advanced decentralized, digital, infrastructures with two-way capabilities for communicating information, controlling equipment and distributing energy" [3]. This infrastructure will be better able to incorporate new forms of energy generation, as well as be self-healing and more robust. Each device in a smart grid will likely have its own IP address and will use protocols like TCP/IP for communication. Thus they will be vulnerable to similar security threats that face present day communication networks [4], however the stakes will be much higher.

This work looks at the threat to some public key encryption systems from the quantum computer in the context of smart grid security. This paper argues for the use of Merkle (Hash) Trees as opposed to public key on the smart grid, specifically when used to authenticate devices in wireless mesh networks (WMN). Some preliminary results of this work have been accepted to present as a poster [5]. This is a continuation of our research effort in smart grid [6] and in wireless mobile network security [7, 8].

A. PROBLEM STATEMENT

In recent years WMN have received a lot of attention and have becoming increasingly popular topologies due to their cost effectiveness and robustness. It seems likely WMN will be widely used in smart grid applications.

Authentication enables a node in a WMN to ensure the identity of the peer node it wants to communicate with. Public key cryptography is a means by which this can be accomplished. Public key is a very secure system today, however if a new type of computer called the quantum computer is fully realized, then a number of commonly use public key system will be broken; this is something we know today [9].

Merkle Trees can also be used for authentication. Their security rests on the use of cryptographically-secure hash functions, which we understand to be resistant to a quantum computer attack [10]. A Merkle Tree is a complete binary tree constructed from a set of secret leaf tokens, where each internal node of the tree is a hash of its left and right child. The leaves consist of a set of *m* randomly generated secret tokens. The root is public, and is the result of recursive applications of the one-way hash function on the tree, starting at the leaves [11].

For this work we implement a Merkle Tree authentication scheme and incorporate it into the ns-3 Network Simulator. We then compare its performance to that of a publicly available version of RSA, a public key encryption system. Our goal is to show that Merkle Trees are a reasonable alternative to public-key cryptography system for smart grid networks.

II. BACKGROUND

A. ELECTRICAL POWER GRID

Electrical equipment is installed with the intention that it will be in service for many years, even decades. To do otherwise would not be efficient or acceptable. Computer and communication technologies advance at a much more rapid pace. As a result, technology on the grid tends to lag. Many functions in the grid today continue to use communications technologies similar to those that were used in the 1980s and 90s such as dial-up connections used for personal computers [12]. Considering the expense, potential for disruption, and difficult to reach locations of some of this equipment, it seems clear why it is not updated with the latest trends in the

978-1-4799-7615-7/14/$31.00 ©2014 IEEE

computer world; the electrical power grid does not abide by Moore's Law.

B. Wireless Mesh Networks

In a WMN each node is a peer and messages can be forwarded through each peer to get to their final destination. Each node is usually connected to several other nodes, thus improving reliability since there can be multiple routes from source to sink. WMN do have drawbacks, particularly in that they are vulnerable to attacks due to their dynamically changing topology, absences of conventional security infrastructure, and wireless nature [13].

C. The ZigBee Standard and Security

The ZigBee standard defines a set of communications protocols for low-data-rate short-range wireless networking [14]. ZigBee beholds to the IEEE 802.15.4 standard, which defines the two bottom layers of the protocol, but then goes beyond that to implement two additional layers. ZigBee is well suited for controls applications because it is a low power, low data communications protocol, which can support a mesh topology. The goal of ZigBee is to be simple and inexpensive [15] and a better alternative to the complicated and expensive WiFi and Bluetooth.

Currently some smart meters in ZigBee WMNs are using elliptic key cryptography (ECC) for authentication [16]. ECC is a public key encryption system based on the discrete log problem. If the quantum computer were available tomorrow all smart meters using this system would become insecure.

III. Existing Solutions

Public key uses a public-private key pair; one used to encrypt, the other to decrypt. This has many advantages. The strength of public key encryption rests on difficult to solve math problems. For this work we are concerned with those based on the factorization problem and the discrete logarithms problem. Our experiment will deal specifically with the factorization problem.

A. Authentication using Public Key

Public key can be used to authenticate two devices in the following manner. For clarity call one device *Alice* and the other *Bob*. *Alice* sends a message to *Bob* claiming to be *Alice*. *Bob* needs more proof than this, so *Bob* encrypts a message *R* using *Alice*'s public key. Since the public key is public, anybody can encrypt a message, but only the holder of the private key can decrypt the message. Alice receives the message *R*, decrypts it then sends it back to Bob. Since she is the only holder of her private key, she has authenticated herself [17].

It is not known for certain at this point if factoring is "difficult" [17]. That is to say, the best factoring algorithm asymptotically is the *number field sieve*, which is an exponential-time algorithm [9]. Solving the factorization problem in a timely manner today is beyond the reach of the most powerful computers and most efficient algorithms.

The concern with information security is not just is it safe today, but will it be safe in the future? At one time the German Enigma machine was the state of the art in data encryption, today breaking it is a challenging graduate level homework problem. Electrical hardware installed in the grid is meant to stay in place for many years and need to remain secure while in the field.

B. The Threat of Quantum Information Processing

Today's computer architecture is based around the solid-state transistor and the binary number system. The quantum computer, currently in its infancy, is not bound to the limits of transistors or the binary system architecture. Instead, it uses atoms held in a magnetic field called qubits. The quantum computer is an analog machine that exploits the laws of quantum mechanics, and as a consequence a single qubit can take infinitely many quantum states. It is more than just that a quantum computer would be faster, it approaches problems differently and in the realm of quantum physics a computer can solve the factorization or discrete log problem in polynomial time rather than exponential time [9].

Large-scale quantum computer hardware requires each qubit to be extremely well isolated from the environment, yet it must be precisely controlled using external fields. These problems are far from trivial [18], however, "no fundamental physical principles are known that prohibit the building of large-scale and reliable quantum computers" [19].

Shor's algorithm, developed in 1994, is a quantum algorithm that attacks the problem of finding the period of a function, need for factoring. It uses quantum parallelism to produce a superposition of all the values of this function in a single step. It then uses a quantum Fourier transform and measuring the yields gives the period, which is then used to factor. It does this in polynomial time [19].

IV. Proposed Solution

A. Secure Hash Functions

It is well known that hash based algorithms are computationally less expensive than symmetric key algorithms, which in turn are computationally less expensive that public key algorithms. Popular cryptographic hash functions like SHA-1, SHA-2 or MD5 work much like block ciphers. They take plain texts and split it into fixed sized blocks, then iterated by way of a function for some number of rounds [17]. A hash function is considered secure if no collisions have been found.

The strength of the Merkle Tree authentication scheme rests on having a secure hash function, and practical cryptographic hash functions do exists. The purpose of a hash function is to produce a "fingerprint" of a message. For example a hash function *s()* is applied to a file *M* and produces *s(M)*, which identifies *M*, but is much smaller than *M* [17]. A

cryptographic hash function must provide compression, efficiency, one-way trapdoor, and weak and strong collision resistance. Strong collision resistance means that it is infeasible to find any x and y, such that $y \neq x$ and $s(x) = s(y)$. In other words, how resistant the hash function is with respect to a class of attack known as the birthday attack.

The goal of the birthday attack on a hash function is not to find a message x such that $s(x) = s(y)$, but rather to find two random messages x and y such that $s(x) = s(y)$ [20]. Thus, "The strength of a hash function against this brute-force attack depends solely on the length of the hash code produced by the algorithm" [10]. Therefore, to defend against a quantum attack the hash code only needs to be increased in length.

B. MERKLE TREES

A Merkle Tree is a complete binary tree constructed from a set of secret leaf tokens, where each internal node of the tree is a concatenation then a hash of its left and right child. The leaves consist of a set of m randomly generated secret tokens. Since it is a complete binary tree, $m = 2^h$ where h is the height of the tree and m is the number of leaves. The root is public, and is the result of recursive applications of the one-way hash function on the tree, starting at the leaves [11].

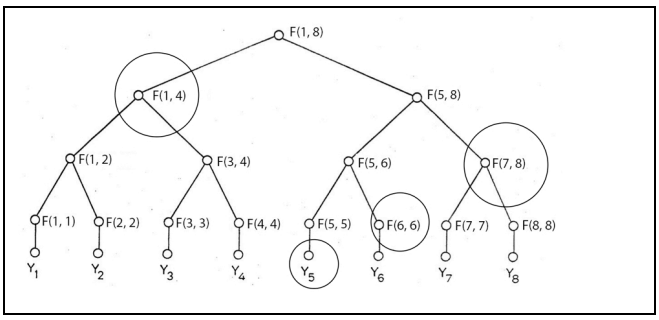

Figure 1. A Merkel Tree with authentication path

Figure 1 shows a Merkle Tree with 8 leaves ($m = 8$). This tree therefore has 8 one-time authentication tokens to offer. In a mesh application the client generates the tree, and the root of the tree is made public. Thereafter, the client can prove its identity to any mesh router, which compares the published root against the root that is generated when the hash function and authentication path is provided. Note that it is computationally infeasible to determine the secret token from the published root of the tree [11].

For example, if a client asks to be authenticated by a mesh router using leaf Y_5, below is the steps that would be carried out (referring to Figure 1):

Let F be a mapping function that we define by:

$$F(i,i) = s(Y_i)$$
$$F(i, j) = s(F(i,k), F(k+1), j))$$
where $k = (i+j)/2$

1. $F(1,8)$ is the root and is public, made known by the router
2. The client sends $F(1, 4)$ and $F(5, 8)$ and the router computes: $s(F(1, 4), F(5, 8)) = F(1, 8)$
3. The client sends $F(5, 6)$ and $F(7, 8)$ and the router computes: $s(F(5, 6), F(7, 8)) = F(5, 8)$
4. The client sends $F(5, 5)$ and $F(6, 6)$ and the router computes: $s(F(5, 5), F(6, 6)) = F(5, 6)$
5. The client sends Y_5 and the router computes: $s(Y_5) = F(5, 5)$
6. The router has now authenticated the client through authenticating Y_5

Note that using this method, only h transmissions are required to authenticate even though the tree has 2^h leaves (secret tokens).

To recap, the client transmits to the mesh router the secret token Y_i and the path to the root. The root is public so there is no need to transmit that. The client is authenticated by the fact that the mesh router is able to regenerate the value of the root based on the hash function $s()$ and the path provided by the client [21].

V. COMPLEXITY ANALYSIS

A. COMPLEXITY ANALYSIS OF MERKLE TREES

1. Complexities of Build-Time and Authentication-Time

Since a Merkle Tree is a complete binary tree the number of nodes at height h is 2^h. The height of the tree with n leaves is $log_2 n$. The number of internal nodes in such a tree of height h is:

$$1 + 2 + 2^2 + ... + 2^{h-1} = \sum_{i=0}^{h-1} 2i$$

$$= (2^h - 1)/(2-1)$$

Therefore, there are $(2^h - 1)$ internal nodes [22].

To build a Merkle Tree in each node we have an asymptotic upper bound of $O(2^h)$ with additional cost for the hash function.

For our experiment the *hash()* function available with the *tr1/functional* library of *C++* was used. *C++* uses *MurmurHashNeutral2* as its hash function, which uses a "Merkle-Damgard-esque" construction for its hash [23]. This has a padding scheme on the front end to make sure all input into the compression function is of the same length. The input is broken into blocks that are then compressed. The compression involves taken the result so far and combines with the next block. Many cryptographic hash functions work this way [24]. So we can say that asymptotically the time complexity of the hash function is $O(\beta)$, where β is the key size.

The total build time is therefore $O(2^h) + O(\beta)$. The time to authenticate is bounded by the height of the tree, as illustrated in the previous section and Figure 1, and the hash function, i.e., $O(h\,\beta)$.

2. Memory Complexity

Clearly the amount of memory a Merkle Tree requires is proportional to the size of the tree, and the key size. Its memory complexity is therefore:

$$(2^h + 1) * \beta = O(\beta\, 2^h).$$

3. Message Complexity

As shown in previous sections, the Merkle Tree sends an authenticating path back to the request. Each entry is $O(\beta)$ and there are h entrjes in this path, so we have $O(\beta\ h)$ message complexity.

B. Complexity Analysis of RSA

1. Computational Time Complexity

Public-private key generation relies on modular exponentiation. This is when an operation is raising one number to a power modulo another number. This is resource heavy—time, energy and processor resources. Assume the *public key: (N, e)* and *private key: d*, satisfy:

$$lg\ e = O(1),\ lg\ d \leq \beta \text{ and } lg\ N \leq \beta$$

Then applying a public key requires $O(1)$ modular multiplications and uses $O(\beta^2)$ bit operations. Therefore the build time complexity is $O(1) + O(\beta^2) = O(\beta^2)$.

For authentication time, to apply a secret key requires $O(\beta)$ modular multiplications, for a total of $O(\beta^3)$ bit operations [22].

2. Memory Complexity

In terms of memory consumption RSA does hold an advantage since it does not require a tree, with one set of keys it can authenticate with an unlimited number of devices. For each node they only need to hold their private key. Since public keys are public, the nodes do not need to retain that information. Therefore the amount of memory used is $O(\beta)$.

3. Message Complexity

RSA works in three exchanges. The message complexity is therefore also $O(\beta)$.

C. Summary

Table 1 summarizes the time, memory and message complexities of Merkle Tree and of RSA.

Table 1. Complexity Analysis

COMPLEXITY COMPARISON			
		MERKLE TREE	**RSA**
COMPLEXITIES	BUILD TIME	$O(2^h) + O(\beta)$	$O(\beta^2)$
	AUTH TIME	$O(\beta\ h)$	$O(\beta^3)$
	MEMORY	$O(\beta\ 2^h)$	$O(\beta)$
	MESSAGE	$O\ (\beta\ h)$	$O(\beta)$

VI. Experiment Setup

The experiments were setup using ns-3, a discrete event network simulator widely used in industry and academia for the purposes of testing and evaluating networks. Both Merkle Tree and RSA authentication schemes were added into ns-3. This experiment was run on a MacBook Air running OS X 10.8.5, with a 1.8 GHz Intel Core i5 and 4 GB 1600MHz DDR3 of memory.

A. Wireless Mesh Network

In industry today we are starting to see utilization of WMN for among other things, lighting control systems. Currently these networks are limited to discrete sections of buildings, not entire buildings, and are often limited to no more than 64 devices. For this reason we defined this experiment to have a network of 64 nodes.

B. Merkle Trees

The Merkle Tree algorithm was coded as described in the previous sections. The Merkle Trees were added into the existing ns-3 node structure, which represent devices in the WMN; this would be the build time. Later when the nodes are being linked into a network, we add a functionality of authentication of nodes; this would be the authentication time.

The initial assumption was that a Merkle Tree with 16 leaves (depth of 4) would be sufficient. If we assume a network of 64 devices, then if every node can authenticate with 16 other nodes around it, that should be plenty to create a robust system. Also, since a bigger tree consumes more resources, there is an advantage to having a small tree; deep trees are no more secure than shallow ones.

C. RSA

The RSA software used was obtained from *rsa Project* [25]. In the RSA scheme a private key is stored at the node. The public key is made public so there is no need for it to be stored in the node. This functionality was added in the same locations as the Merkle Tree in ns-3.

RSA is able to use a public-private key pair to authenticate itself with any number of other nodes; this is an advantage over Merkle Trees. That is, the Merkle Tree scheme needs to know ahead of time how many nodes (devices) it will need to be able to authenticate with.

RSA key generation calculations depend on the length of the keys. For the sake of this test we choose 32 bits. We also have the length of our Merkle root at 32 bits. Albeit this would not be secure in a real system, it gives us good modeling data in a reasonable amount of time. We do test larger keys to see what impact the length has on calculation times.

VII. Experiment Results

For our comparison we wanted to see how big of a Merkle Tree we could build and authenticate with. A large tree provides a large number of authentication tokens. Since RSA

can authenticate with as many devices as it wants. We measure the time taken during the construction of the nodes (Build Time), then again during the linking of the nodes where the authentication process was performed.

A. Build Time

We can see from Figure 2 that RSA is very slow to build compared to Merkle, taking on the order of 35000 milliseconds to build. When compared to a Merkle Tree of shallow depth, we see the Merkle scheme has a clear advantage. We do see the Merkle scheme slow as the tree grows larger. Around a depth of 16 we start to see noticeable slowing in the Merkle scheme. At a depth of 16 each node has 65,536 leaves to authenticate with.

It was not possible to see at what depth the Merkle Tree equaled RSA's time, because at depth 25, the computer that was running the tests started to report memory problems, then seized up. At that depth we were building a Merkle Tree with 33,554,432 leaves.

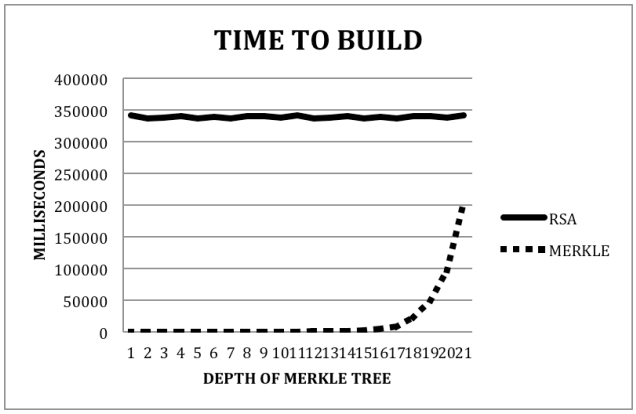

Figure 2. Build Time Comparison

B. Authenticate Time

For the Merkle Trees we can see from Figure 2 that the larger the tree the more traversing of the tree we need to do to provide our authentication path. Still Merkle continues to do better than RSA for authentication. In these plots RSA is using a 32-bit key. With the 256-bit key, RSA did well worse than the Merkle Tree taking about 3 minutes to authenticate one single node. This number would then be proportional to the size of the network and number of links. What we can see from all of this is that Merkle Trees are a viable alternative to the use of public key for authentication.

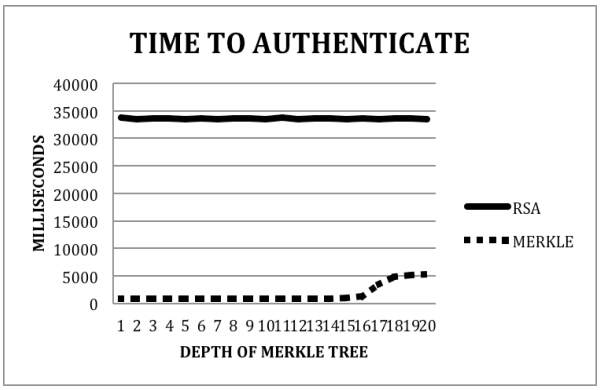

Figure 3. Authentication Time Comparison

VIII. Related Studies

An important work on smart grid, or smart distribution grid (SDG), security was by Wang and Yi [26]. They investigated two issues. First, they proposed a security framework for SDG based on WMN, and analyzed the potential security attacks and possible counter-attack measures. Next, they developed a new intrusion detection and response scheme (smart tracking firewall). They evaluated its performance, and found that the smart tracking firewall and detect and respond to security attacks in real-time, and thus suits its application for SDG.

Another recent work [27] evaluates authentication schemes for multi-gate mesh network in smart grid applications. This work provides additional support for the use of Merkle Trees in smart grid. The most recently adopted IEEE 802.11s standards supports simultaneous authentication of equals (SAE) for its security protocol. This protocol uses one password shared by all devices. The standard also offers efficient mesh security association (EMSA) as an alternative approach. Both protocols use 4-way handshaking at which time a network is vulnerable to denial of service (DoS) attacks. The authors provide a method of 4-way handshaking that uses a Merkle Trees based scheme for authentication. They use ProVerif to analysis the vulnerabilities of the network and the resilience added by use of Merkle Trees to defend against DoS attacks. This work does not look at quantum computer attacks on WMN.

IX. Conclusion

An important factor in the quality of life in the future will depend on energy; how we get it, how we use it, how we distribute it. Smart grid is an important step toward a future with a quality of life better than the one we have today. Smart grid will take a generation to complete yet everything that is to come must be built on a solid foundation of information security.

The main objective of this work is to discourage the use of discrete log and factorization based public key encryption in smart grid communication domains. We looked at public key authentication in WMN and offered as an alternative a Merkle Tree based authentication mechanism. The build-time and authentication-time complexities, and memory and message

complexities of RSA, a public-key authentication method and of our proposed Merkel-tree-based authentication methods are analyzed and compared. Their performance in terms of build-time and authentication-tie are evaluated and compared. We found that the proposed Merkel-tree-based method is lightweight, and takes less time to build and to authenticate. These studies showed that Merkle Trees-based authentications are lightweight, secure, resistant to quantum computer attacks.

The next step in this research is to build a network with an RSA key of up to 2048 bits, which today would be considered a secure length. We would also attempt to study public key encryption systems that are not vulnerable to a quantum computer attack, such as the NTRU Cryptosystem [28]. We would still expect them to be slower than Merkle Trees. Merkle Trees therefore offer lightweight, secure and quantum attack resistant security that should be considered for use in smart grid applications.

REFERENCES

[1] Kathy Kowalenko, "The Smart Grid: A Primer", *The Institute, IEEE*, p. 5, December 2010.

[2] U.S.-Canada Power System Outage Task Force, "Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations", p. 1, April 2004.

[3] NIST 7628, "Guidelines for Smart Grid Cyber Security", p. 3, September 2010.

[4] Ye Yan, Yi Qian, Hamid Sharif, David Tipper, "A Survey on Cyber Security for Smart Grid Communications", *Communication Surveys and Tutorials, IEEE,* Volume 14, issue 4, Section III, p. 4, January 2012.

[5] M. Muñoz, M. Moh, T.-S. Moh, "Improving Smart Grid Security using Merkle Trees," accepted to present at *IEEE Conference on Communications and Network Security* (CNS'14), to be held in San Francisco, Oct 2014.

[6] A. Kapoor "Implementation and evaluation of the DFF protocol for Advanced Metering Infrastructure (AMI) networks," Technical Report (Master Writing Project), Dept. of Computer Science, San Jose State University, June 2014.

[7] R. Wong, T.-S. Moh, and M. Moh, "Efficient Semi-Supervised Learning BitTorrent Traffic Detection: An Extended Summary," in *Proc. of 13th Int. Conf on Distributed Computing and Networking – ICDCN 2012*, held in Hong Kong, China, January 3-6, 2012; and *Lecture Notes in Computer Science 7129*, Springer 2012.

[8] L. Yang and M. Moh, "Dual Trust Secure Protocol for Cluster-Based Wireless Sensor Networks," in *Proc. IEEE 45th Asilomar Conference on Signals, Systems and Computers*, held in Pacific Grove, CA, Nov. 2011.

[9] Peter Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", *SIAM Journal of Computing*, p.26, 1997

[10] William Stallings, *Cryptography and Network Security*, p. 259, Prentice Hall, 1999.

[11] Lakshmi Santhanam, Bin Xie, Dharma Agrawal, "Secure and Efficient Authentication in Wireless Mesh Networks using Merkle Trees", *33rd IEEE Conference on Local Computer Networks*, 2008.

[12] IEEE-USA Board of Directors, "Building a Stronger and Smarter Electrical Energy Infrastructure", pp. 6-9, February 2010.

[13] Muhammad Shoaib Siddiqui, Choong Seon Huong, "Security Issues in Wireless Mesh Networks", *MUE '07 International Conference on Multimedia and Ubiquitous Engineering*, p. 718, April 2007.

[14] ZigBee Specifications 053474r17, ZigBee Alliance, Jan 2008

[15] Josh Wright, "ZigBee Hacking and the Kinetic World", http://www.youtube.com/watch?v=BkVcElfOVyw, 2010.

[16] Certicom Device Certification Authority for ZigBee Smart Energy,https://www.certicom.com/index.php/device-authentication-service/smart-energy-device-certificate-service, 2014.

[17] Mark Stamp, *Information Security Principles and Practices*, Wiley, 2011.

[18] C. Monroe, J. Kim, "Scaling the Ion Trap Quantum Processor", *Science*, p. 1164, March 2013.

[19] Eleanor Rieffel, Wolfgang Polak, *Quantum Computing, A Gentle Introduction*, The MIT Press, 2011

[20] Bruce Schneier, *Applied Cryptography*, p. 429, Wiley, 1996

[21] Ralph Merkle, "Secrecy Authentication and Public Key Systems", *Information Systems Laboratory, Stanford Electronics Laboratories*, Chapter V, p. 40, June 1979.

[22] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, The MIT Press, 2009.

[23] Austin Appleby, "Murmur Hash," https://sites.google.com/site/murmurhash/, 2014.

[24] Wikipedia, "Merkel-Damgard Construction," http://en.wikipedia.org/wiki/Merkle-Damgard_construction, 2014.

[25] The RSA Project, https://code.google.com/p/rsa/, 2014

[26] X. Wang and Ping Yi, "Security Framework for Wireless Communications in Smart Distribution Grid", *IEEE Transactions on Smart Grid*, Vol. 2, No. 4, December 2011.

[27] B. Hu and H. Gharavi, "Smart Grid Mesh Network Security Using Dynamic Key Distribution With Merkle Tree 4-Way Handshaking", *IEEE Transactions on Smart Grid,* Vol. 5, No., March 2014.

[28] NTRU Crypto System, http://tbuktu.github.io/ntru/, 2014.