A Case for Validating Remote Application Integrity for Data Processing Systems

Jonathan M. Chu, Mirko Montanari, Roy H. Campbell
Department of Computer Science
University of Illinois at Urbana-Champaign
{jmchu2, mmontan2}@illinois.edu

***Abstract*—There has been a great increase in recent years as to the amount of data from the grid that has been going to online systems. As more smart meters get installed into the AMI(advanced metering infrastructure), there is a need to mitigate the potential security threats in the collection system. There are a multitude of attack vectors that an adversary may take to compromise the confidentiality of user data and it may take much time and effort for developers to securely cover all such attack vectors. In this paper, we analyze the architecture of AMI systems and how data moves from one end to the other. In particular, we discuss the need for more research in safe program validation that protects against information leaks. Security problems can arise when programs do not perform as intended and may reveal confidential information or take unexpected actions. We discuss a theoretical network architecture that could take advantage of such secure program validation. The model minimizes attack vectors by containing data in one secure location that we call a DBPC(database processing center) instead of transporting data to multiple locations through a traditional DBMS(database management system). When outside parties want access to the data, they can send verified secure applications to the DBPC to run their applications remotely without direct access to the data. We describe the design of the AMI simulator and DBPC prototype module that we implemented.**

Keywords-Advanced Metering Infrastructure; Database Processing Center; Database Management Systems; security; confidentiality; smart meters.

I. INTRODUCTION

Protecting data collected from consumers is one of the primary objectives for security in AMI systems[1]. Electrical data from smart meters can determine the energy usage patterns of a household and reveal private information about the consumer. This could include important data such as whether someone has gone on vacation and left their home unattended which could allow malicious people to take advantage of those situations. Moreover, power usage could leak important side channel information which could lead to attacks that could even make the encryption on computers vulnerable[2]. Data shows that there has been a great rise in the number of smart meters installed into the infrastructure and it has reached over 20 million units already, a number that is expected to grow even more in the coming years[3]. It has been shown that there are numerous attack vectors from which malicious hackers could attack the AMI. Covering all these vectors takes time and resources that utility companies have a hard time covering[4][5]. If even one aspect of security is missed, it could be enough for a hacker to take control of and shutdown the system. To implement an exhaustive and complete security coverage, developers must have an extremely high level of attention and awareness at all times, which is unreasonable to expect. Most security vulnerabilities do not occur because there is not a solution to them, but rather because there might not have been sufficient time and resources to implement a complete solution to every single possible problem. That is why we believe that is important to develop models of security which focus on easing the burden on developers and achieve the highest level of security with minimal resources. This can be achieved by reducing the number of potential attack vectors in the system so that there are fewer places that developers need to worry about protecting. Fig. 1 shows the different points of attack in the security stack for AMI. Also, AMI systems need to scale as more smart meters grow at an exponential rate[6]. In security there are multiple properties of distributed systems can be protected[1][7].

It is our position that there should be more of a focus in security on validating applications and programs. There are just not enough validation mechanisms available for companies to ensure that they are running safe programs or looking at safe documents. There should be a module that can check these programs and then send application digest messages to all interested third parties who may want to download the program so that they can compare the cryptographically secure digest with the program that they received and check its integrity.

In this paper, we introduce a new model that focuses on minimizing the distribution of data to different sites. We propose that data should be kept in one secure location that would be easy for developers to protect. By reducing the attack paths for an attacker, utility companies can focus their best efforts at protecting one key location instead of spreading themselves out too thin with resources they may not have. This is possible by developing a module that can receive database querying and processing applications from third parties and running verification checks to ensure the authenticity and safety of each new application that gets sent. By keeping data in one place, the DBPC(database processing center) can run other parties' applications locally while still protecting the confidentiality of the data from outsiders. Outsiders can still obtain the necessary statistics to complete their objectives and they do not need access to the intermediate data, they only need to know the final results.

II. STRONG SECURITY PRACTICES

A. *Why Program validation is important*

Although there are many cryptographic techniques that have been developed on ensuring that data is correct. There has been much less research done in ensuring that programs are correct. Programs should be checked for correctness because obviously, they handle the data and have access to vital parts of a system. Yet, despite this obvious vital part of computer security, there are not many ways to check whether a program is safe. It can be extremely difficult to tell what a program will do. There are even problems like the halting problem which state that one cannot know whether a program will even terminate or not.

Issues arise when programs do things that are unexpected or files are modified to contain malicious scripts. Programs which run on any system should be validated before being allowed any access. Potential avenues of research could be in looking at information flow control, which is research in controlling what information does a program look at or manipulate. This can be done through creating a new programming language or compiler that has labels which can check for program safety. An application developer might program in such a new safe language which would be able to validate itself with a compiler. A third party may be able to perform static analysis of the program if it knew that the program had to obey certain rules in flow control. There could also be more research done in the area of checking the safety of existing programming languages. Such analysis could include such areas as timing analysis to see if a malicious program would wait until a certain time before proceeding.

We believe that it is important that future research in AMI focus on validating program safety so that programs which run on databases are secure and that they only do what they are intended to. We think it is possible to develop specific security models for the particular dimensions of AMI that can help in fulfilling these tasks in a way that may not be feasible in the general case for all types of programs.

B. *Identifying Proper Security Mode: Confidentiality vs Avaliability*

There are two main security properties of systems, confidentiality and availability. We will next discuss which model of security fits a system better and why the AMI system is better suited for the confidentiality model while SCADA systems are better suited for the availability model.

1) *Confidentiality*

Confidentiality is when a system prevents the disclosure of information to unauthorized users who may be malicious users purposely trying to circumvent security or regular users who may unintentionally access or modify data that they are not authorized to.

One method to gain confidentiality is encryption. Symmetric encryption is when two nodes share a common secret key that nobody else knows about and share data between themselves by encrypting and decrypting messages with that secret key. An alternative is public/private key encryption. Access control protocols such as RBAC(Role-based access control) can also help in ensuring that only valid users have access to the systems that they are supposed to have access to [8].

2) *Availability*

Availability is when a system is always up and working to send messages, receive messages, respond to messages, process messages, etc. To create availability, a useful technique is to create backup systems that run in parallel to main systems. In this way one can double the reliability of a system if the backup is as stable as the main system. The system must also be able to detect malicious software that may try to take control of the system and do things that could damage it. If not, incidents like Stuxnet could occur in which critical machines are infected and operations are disrupted[9].

There are so many attack vectors that security developers struggle to cover all the possible entrances for an attacker. It is how the security developer puts together the pieces in the security stack which causes most vulnerabilities and random implementation issues[10][11].

III. ADVANCED METERING INFRASTRUCTURE

The AMI(advanced metering infrastructure) is a distributed collection system that gathers data from the smart meters located at the home and reports it to the utility company. As seen in Fig. 2, nodes in an AMI system communicate in both directions. Traditional AMR (advanced meter reading) systems only had one way communication from the meter to the utility company. AMI starts off at the HAN(Home Area Network) which may consist of the electrical metering devices within a home. These devices report their readings to the smart meter which collects all the data and prepares to send them out through what may be a repeater. The smart meter should encrypt the messages using a secure cryptographic protocol such as RSA or AES. The standard communication protocol is C12.18 for North American utilities and IEC 61107 for utility meters in the European Union[12]. It is very

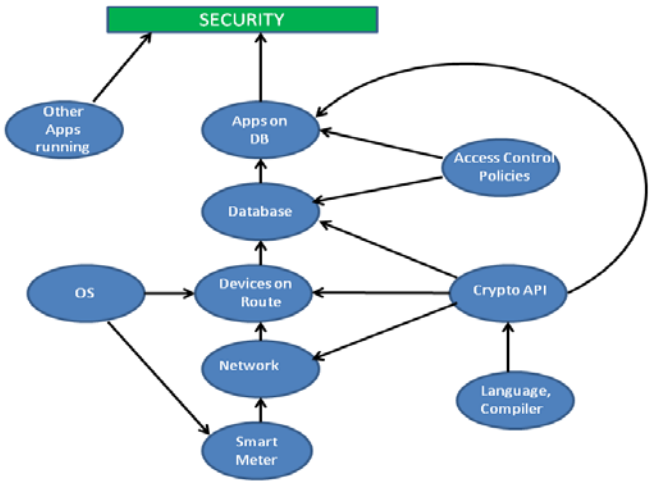

Fig 1: Security Stack for AMI

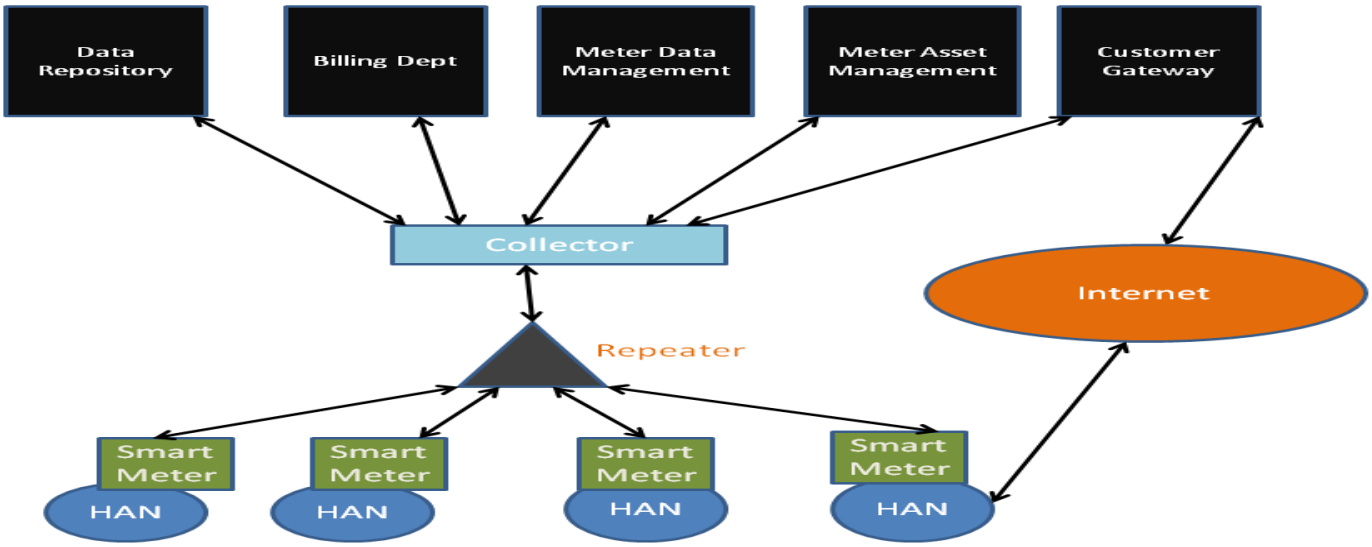

Figure 2: AMI Architecture

important for the smart meter to be secure so that people do not try to tamper with it and produce fraudulent numbers to begin with. There should be tamper proof seals and alarms to alert the utility company if someone tries to modify the smart meter in anyway. Also there should be an attestation protocol that compares the integrity of the programs on the embedded so that any modifications to the monitoring software on the smart meter are detected[13]. The utility company may have a meter asset management system in place to handle these checks and also to update the firmware on the smart meters themselves.

The smart meter may commonly wirelessly send their messages to a repeater node which relays the data farther out into the wide area network until it reaches an AMI Headend/Collector. This node serves to act as a gateway to the internet services by providing date to such systems as the data repository, the billing department, the meter data management system, the meter asset management system, and the customer gateway. It is important that the data is securely encrypted and employing the latest communication protocols to prevent eavesdropping and stealing of information as smart meter data travels from the local home to the servers of the utility company. One must be aware of replay attacks which can resend a duplicate packet such as a password for a login to be used by an attacker who may claim to be someone who he is not. This is significant because this type of attack may be more complex to defend against and develop protection for and would be one of the steps that could be missed in the security development process. It is usually the smallest obscure security features which gets exploited so it is important for a security developer to focus on small details. Also a man in the middle attack may occur when someone tries to tap into the line and pretend to be someone who he is not. It has been shown that it is possible to do man in the middle attacks even in encrypted VPN tunnels if one is not careful[14]. That is why it is important for security developers to not only implement secure protocols, but to implement them well and robustly using the latest cryptographic procedures.

The billing department will collect the metering data from the collector and put it in its own database to run pricing algorithms to measure the total cost that a customer may owe to the utility for his energy consumption for that month. Keep in mind that smart meters are more advanced than traditional utility meters because they are able to include the exact time when each unit of energy was spent whereas an old meter can only determine the total energy used at the end of the month. This means that smart meter data can reveal information about customers concerning their energy usage patterns. It also means that the utility company can create a more advanced pricing scheme which can charge customers more for using energy at peak service hours.

The MDMS(Meter Data Management System) is used to keep track of the data and validate it and assure that the data has not been modified on the path from the smart meter to the MDMS. It can do this by calculating hash values of the data or checking the MAC value of the encryption protocol. The Meter Asset Management system on the other hand, checks the status of the smart meters and validates that they are functioning correctly. They may issue remote commands to turn off/on the smart meter depending on the circumstances.

The customer gateway can be accessed over the internet by the consumer to check his electricity usage and see how much money he is spending per month and how to conserve energy. When people are more conscious of their energy usage, they are more likely to improve and reduce their use. However, the internet may not be a safe place for data to transmit so it is important for the consumer to be sure that his connection is secure and safe.

IV. Architecture of Program Verifying Infrastructure

Our architecture is centered around a DBPC(database processing center) that utilizes an infrastructure that can check and validate applications before they are sent to the DBPC to be run. This infrastructure consists of a PVA(program verifying authority), a traditional PKI(public key infrastructure), an external party who wants to run their program on the DBPC and the DPBC itself as can be visualized in Fig 3.

A. Program Verifying Authority

The PVA is crucial in accepting programs from third parties who wish to run scripts on the database processing center. The PVA is responsible for validating the programs by running it through analysis which could be carried out manually through a human being or automated through a more complex mechanism such as information flow control[15]. It can be hard to determine the safety of executing programs automatically as there does not exist a general purpose language that can do such validation yet. However, a human could manually look at the code and run sample tests to determine whether a program is malicious or not. There can be security monitoring tests on the computer to see if anything malicious occurs during runtime or whether the data outputted is the same as what the original third party author claimed would be outputted. It would primarily check to see that the program does not output any confidential information that it should not be revealing to the outside party. For example, if an outside party like the government wanted to do analysis of the power consumption of a city, then only the aggregate results should be outputted. There should not be results from individual consumers. One could have policies in place for programs that state rules like programs must request data from at least 100 consumers and only be able to obtain aggregate statistics. Another type of rule could be that a program can only access the overall monthly electricity total, but not the individual time vectors of when the electricity was being consumed. A validation process for a program would include checking which data it would access. Also, the validation process would then need to ensure that the data accessed was then properly mixed together so that one could not obtain anybody's individual data from the output. Moreover, the PVA would need to keep a history of programs requested from the third party to make sure that it is not abusing the system and issuing too many program requests which could delay the time of service to validate authentic good programs. There could also be cases where a program that is only run once is safe, but when run twice is unsafe. For example if a program calculates aggregate statistics for 100 consumers, and then in its next iteration calculates aggregate statistics for those same 100 users plus one more additional user, then it could figure out what the individual user's statistics are from how it affected the aggregate numbers. There are many ways that a program could be corrupted and the area of studying the safety of programs is an open topic in research.

B. Message Digest for Application

However, if a program has been validated as being safe, the PVA would calculate a digest of the program by calculating a hash value using the latest SHA-2 hash function. This hash value is usually unique to the original program and relies on the property that it is very difficult for a hacker to generate an equivalent malicious program with the same hash value as the original. Even the slightest change to the original program would completely change the hash value.

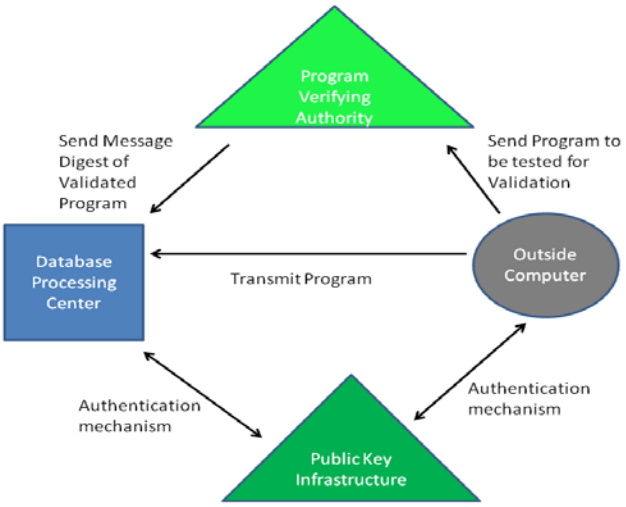

Figure 3: Program Verifying Infrastructure

If a program has been successfully validated, the PVA would send the message digest to the DBPC. In the future when the DBPC received a request to run a program from a third party, it could calculate the message digest of the program to be run and compare it to the DBPC's library of validated programs. If it correctly matched, then it would give the program the green light to execute on the database.

C. *Digital Certificates*

If there are many different programs, then having every DBPC keep track of all the hashes of the different programs may be too much strain on the disk space of the DBPC. A more scalable version that could be created in future versions would be a PVA that generates digital certificates which guarantee the authenticity and validity of the program as verified by the PVA which issued the certificate.

In the future we imagine that there may be multiple PVAs which exist online and a database could selectively decide which digital certificates from which PVAs were reputable and only accept programs from senders that have the necessary digital certificate.

D. Types of Programs

The PVA should ideally verify any program that runs on the database. While it is important that a program does not modify the database in a malicious manner, there are also programs that hackers may try to send that have other goals such as exploiting buffer overflows and taking root control of a computer system. Therefore even relatively innocuous programs that may claim to only read the database, may have malicious intent.

E. Public Key Infrastructure

The DBPC would also need to check the access control permissions of the entity who sent the program. This could be done through the use of a PKI(public key infrastructure) which could run in parallel to the PVA. The PKI could verify the authenticity of the entity to ensure that there is no man in the middle attack. The PKI can do this by utilizing public/private key algorithms such as the standard RSA algorithm which is very secure. It generates digital signatures for third parties and issues out digital certificates which bind signatures to identities so that the DBPC can verify that the computer it is communicating with is indeed the real one.

V. Implementation of AMI Simulator and PVA

To implement a simulator for the Advanced Metering Infrastructure we utilized the JAVA 1.7 programming language and created a software simulator by constructing JAVA classes and objects which represented the various components in the AMI system as shown in Fig. 4. The JAVA language was chosen because of its portability and strong support for multithreading environments and also for its support of cryptographic libraries. A program can be written once in JAVA and still be run in any operating system that has the Java Virtual Machine. First, a smart meter object was created. This object was set up to receive simulated from a Home Area Network. The data from the Home Area Network was created by looking at a study of previous measurements that were likely to come from the home[16]. The homes in this particular study had an average utilization of 6769 kWh/yr, which was similar to the statistics stated by PG&E which showed that the Northern California average home consumed 6287 kWh/yr. We looked at some of the figures in this study to create a random simulation of fluctuating electrical flow that would happen 20 times a minute. To do this we used the pseudorandom generator in JAVA to come up with the random distribution of electrical voltage and current. A repeater class was also created which received data from the smart meter object and relayed it to the AMI Head End/Collector object. The repeater was programmed to be multithreaded and utilized the socket class to accept incoming messages through TCP. It was programmed to be able to handle multiple incoming connection requests while still only maintaining one outgoing connection to the AMI Head End/Collector. Threads in JAVA share the same memory and open files. This was able to be done through the use of synchronous methods in java which allows multiple threads to share the use of one resource while still guaranteeing consistency. The Collector object would take in the data and send it to the DBPC for storage and processing.

The DBPC was set up to store the data in a MySQL v5.5 database. Then we created the PVA which would take in JAVA JAR files and create a digest of the program and verify the validity of the program for use in the DBPC. Once a program was validated, it would send a message to the DBPCS through a TCP connection. A third entity was programmed that we called the outside party which would generate the jar file and send it to the PVA for validation. Once the third entity received confirmation that the program had been validated, it could send the program to the DBPC to be executed on the database where it would then be run. The DBPC would check that the program was the same one that had been validated by computing the

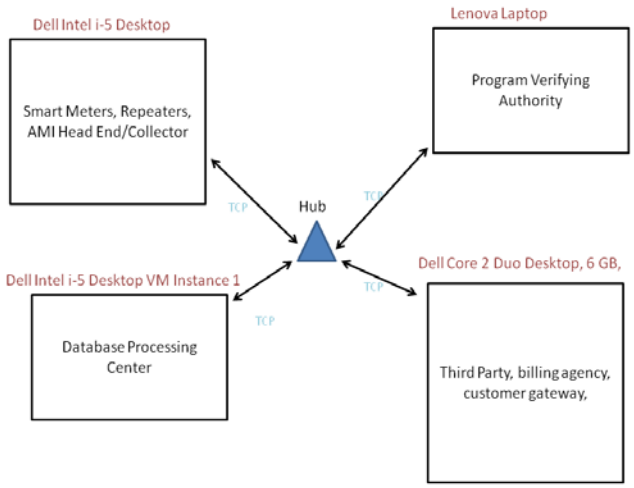

Figure 4: Structure of AMI Implementation

SHA-2 value of the program and comparing that value to the list of approved programs that it received from the PVA. Each of the objects in this simulation were spawned as separate processes on one machine running on Windows 7 on an intel i-5 2400 processor and 16GB of RAM.

JAVA has a tool for signing and verifying JAR files called jarsigner. This tool utilizes a JAVA keystore which keeps track of private keys and their associated public-key certificates. It was this tool which was used to ensure the integrity of the validated programs.

VI. RESULTS

Using our system, we were able to demonstrate a proof of concept to implement our DBCP model. We were able to send a validated program through our AMI simulator that could operate on the confidential dataset and only output the relevant results to the third party. By implementing this, we were able to provide greater confidentiality for the users by providing an extra layer of security at the application level and not just the data level. It could operate efficiently in terms of time depending on what process was used to validate the programs.

VII. Limitations

While we believe in our approach, we also understand the potential limitations in that verifying programs can be extremely complex to do. Having humans read through source code to ensure correctness can be difficult and time consuming. Moreover, it may not be practical for large systems to have all their data in one place if different parts of the system are not connected to one another. Also, having a PVA that can service multiple systems from different organizations would mean putting the PVA online and the putting the corresponding client online as well which could hurt security if one does not want any of their computer systems on the internet.

VIII. Related and future work

Current industrial companies like Oracle, eMeter, and Itron do not handle application integrity like we describe and do not protect sufficiently against rogue scripts[17]. Recent research has shown that a static approach to program validation through information flow control is the preferred, secure way to check programs. Dynamic techniques are not as effective because it is too difficult to check all the different paths that may come from a program and the side effect issues[18]. This is in line with our static Program Verifying Authority which checks the safety of programs before they are run. There has also been recent work done in information flow control for multiple parties and the calculus behind how to prevent processes from leaking information[19]. There has also been work in doing hybrid information-flow control in which both static and dynamic flow control is used to protect information[20]. Moreover, recent work in homomorphic encryption has made computing on encrypted data more plausible, which would give databases the ability to run queries and search through encrypted data without ever decrypting it[21]. Differential privacy is another interesting approach that can involve preserving individual privacy by adding noise to the database. By adding a value to one person's data while subtracting that value from another person's data, one can change the individual values while still keeping the aggregate values the same.

IX. Conclusion

In this study, we look at how to securely transmit safe programs over the internet to be used by databases and third parties. Our model enhances security by minimizing the attack vectors from which an adversary can attack a system. It reduces the amount of data that needs to be duplicated and instead puts data securely into one safe Database Processing Center. The validated programs which run on the DBPC are not aware of the individual intermediate values and only get the output that they need. This falls in line with the security idea, "the principle of least privilege," which states that users should not have more rights or access to resources that they do not need. In the future, we envision that more work will be done in analyzing programs for information leaks and the creation of new programming languages and compilers that will be better able to perform static analysis of the paths that information may take. However, this is definitely a very difficult field and much work must still be done to better ensure the security of programs that work with sensitive, confidential data.

Acknowledgment

This material is based upon work supported by the Department of Energy under Award number DE-OE0000097.

References

[1] Cleveland, F.M.; , "Cyber security issues for Advanced Metering Infrastructure (AMI)," Power and Energy Society General Meeting – Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE, vol., no., pp.1-5, 20-24 July 2008.

[2] Hwang, D.D.; Tiri, K.; Hodjat, A.; Lai, B.-C.; Yang, S.; Schaumont, P.; Verbauwhede, I.;,"AES-Based Security Coprocessor IC in 0.18-um CMOS with Resistance to Differential Power Analysis Side-Channel Attacks, "Solid-State Circuits, IEEE Journal of, vol.41, no.4, pp. 781-792, April 2006.

[3] "Form EIA-861," U.S. Department of Energy, OMB No. 1905-0129, 2010.

[4] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel, "Multi-vendor penetration testing in the advanced metering infrastructure," in Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010, pp. 107-116.

[5] G. Lenzini, M. Oostdijk, and W. Teeuw, "Trust, Security, and Privacy for the Advanced Metering Infrastructure," Novay/RS/2009/010, 2009.

[6] R. Berthier, W. Sanders, and H. Khurana. "Intrustion Detection for Advanced Metering Infrastructure: Requirements and Architectural Directions" in 1st IEEE International Conference on Smart Grid Communication, pp 350-355, 2010.

[7] Oracle Utilities. Advanced Distribution Management [White paper].

[8] D. Ferraiolo and R. Kuhn, "Role-Based Access Controls," 15th NIST-NCSC National Computer Security Conf., pps. 554-563, Baltimore, MD, Oct. 13-16 1992.

[9] Kim Zetter. (2011, July 11). *How Digital Detectives Deciphered Stuxnet, the Most Menacing Malware in History*.

[10] RJ Anderson, 'Security Engineering – A Guide to Building Dependable Distributed Systems', Wiley (2001) ISBN 0-471-38922-6.

[11] Viega J, McGraw G (2002) Building secure software: how to avoid security problems the right way. Addison-Wesley, Boston.

[12] Snyder, A.F.; Stuber, M.T.G.; , "The ANSI C12 protocol suite - updated and now with network capabilities," *Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, 2007. PSC 2007* , vol., no., pp.117-122, 13-16 March 2007.

[13] M. LeMay, G. Gross, C. A. Gunter, and S. Garg, "Unified architecture for large-scale attested metering," in Hawaii International Conference on System Sciences. Big Island, Hawaii: ACM, January 2007.

[14] N. Asokan, V. Niemi, and K Nyberg. Man-in-the-Middle in tunneled authentication protocols. Technical Report 2002/163, IACR ePrint archive, October, 2002.

[15] Myers, A. 1999. JFLOW:Practical, mostly-static information flow control. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*(San Antonio, Tex.), 228-241.

[16] J.P. Ross, and A. Meier, "Whole-House Measurements of Standby Power Consumption," in Proc.2nd Int'l Conf. on Energy Efficiency in Household App. And Lighting, vol. 108(13), 2000.

[17] Bob Lockhart, "Pike Pulse Report: Meter Data Management," Pike Research, 2011.

[18] A. Russo, K. Claessen, and J. Hughes. A library for light-weight information-flow security in haskell. In Haskell Symposium, pages 13-24. ACM, 2008.

[19] S. Capecchi, I. Castellani, M. Dezani Ciancaglini, and T. Rezk. Session Types for Access and Information Flow Control. In P. Gastin and F. Laroussinie, editors, Proc. CONCUR'10, volume 6269 of LNCS, pages 237-252. Springer 2010.

[20] Moore, Scott and Stephen Chong. Static analysis for efficient hybrid information –flow control. Proceedings of the 24th IEEE Computer Security Foundations Symposium (CSF): June 27-29, 2011, Cernay-la-Ville, France.

[21] Gentry, C. Fully Homomorphic Encryption Using Ideal Lattices. In STOC, 2009.