12-2018

Cyber Physical System Security — DoS Attacks on Synchrophasor Networks in the Smart Grid

Xingsi Zhong
Clemson University, xingsiz@g.clemson.edu

Cyber Physical System Security — DoS Attacks on Synchrophasor Networks in the Smart Grid

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Engineering

---

by
Xingsi Zhong
December 2018

Abstract

With the rapid increase of network-enabled sensors, switches, and relays, cyber-physical system security in the smart grid has become important. The smart grid operation demands reliable communication. Existing encryption technologies ensures the authenticity of delivered messages. However, commonly applied technologies are not able to prevent the delay or drop of smart grid communication messages.

In this dissertation, the author focuses on the network security vulnerabilities in synchrophasor network and their mitigation methods. Side-channel vulnerabilities of the synchrophasor network are identified. Synchrophasor network is one of the most important technologies in the smart grid transmission system. Experiments presented in this dissertation shows that a DoS attack that exploits the side-channel vulnerability against the synchrophasor network can lead to the power system in stability.

Side-channel analysis extracts information by observing implementation artifacts without knowing the actual meaning of the information. Synchrophasor network consist of Phasor Measurement Units (PMUs) use synchrophasor protocol to transmit measurement data. Two side-channels are discovered in the synchrophasor protocol. Side-channel analysis based Denial of Service (DoS) attacks differentiate the source of multiple PMU data streams within an encrypted tunnel and only drop selected PMU data streams. Simulations on a power system shows that, without any countermeasure, a power system can be subverted after an attack. Then, mitigation methods from both the network and power grid perspectives are carried out.

Form the perspective of network security study, side-channel analysis, and protocol transformation has the potential to assist the PMU communication to evade attacks lead with protocol identifications.

Form the perspective of power grid control study, to mitigate PMU DoS attacks, Cellular

Computational Network (CCN) prediction of PMU data is studied and used to implement a Virtual Synchrophasor Network (VSN), which learns and mimics the behaviors of an objective power grid. The data from VSN is used by the Automatic Generation Controllers (AGCs) when the PMU packets are disrupted by DoS attacks. Real-time experimental results show the CCN based VSN effectively inferred the missing data and mitigated the negative impacts of DoS attacks.

In this study, industry-standard hardware PMUs and Real-Time Digital Power System Simulator (RTDS) are used to build experimental environments that are as close to actual production as possible for this research.

The above-mentioned attack and mitigation methods are also tested on the Internet. Man-In-The-Middle (MITM) attack of PMU traffic is performed with Border Gateway Protocol (BGP) hijacking. A side-channel analysis based MITM attack detection method is also investigated. A game theory analysis is performed to give a broader view of this problem.

Acknowledgments

I would like to thank my advisor, Dr. G. Kumar Venayagamoorthy, and co-advisor Dr. Richard R. Brooks for their unconditional support to carry out the research and write the dissertation. This work might not have been accomplished without their abundant knowledge, and professional experiences. I would also like to thank them for their patience and dedication of time. Additionally, I would like to thank Dr. Rajendra Singh, and Dr. James J Martin for serving as my advisory committee. Lastly, thanks to my parents for their understanding and decades of support.

This work is supported in part by the US National Science Foundation (NSF) under grants 1312260, 1308192, 1232070, NSF Cyber-Physical Systems (CPS) grant 1544910, U.S. Dept. of State award number S-LMAQM-12-GR-1033, and the Duke Energy Distinguished Professor Endowment Fund. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the financial supports.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The “smart grid” is a modern power system that uses information and communication technologies to provide more efficient, reliable, and sustainable energy than traditional power grid [110]. Different from the conventional power grid, a smart grid allows for a bidirectional flow of information and energy, and the penetration of renewable energy sources, smart parking lots and prosumers makes the smart grid more flexible but also challenging to control [19]. Real-time monitoring and feed back from network connected sensors and devices are crucial to smart grid operations. As the network connection has become a part of the control loop of the smart grid, the smart grid is facing a whole new challenge in the area of network security.

Smart grid technologies such as synchrophasor networks consisting of Phasor Measurement Units (PMUs) make it possible to monitor, analyzes, and control the electric power grid in real-time. PMUs are one of the most important network connected devices used in smart grid. The PMU measure frequencies, currents, voltages, and phase angles. Thses measurements are labeled with GPS time tags and transmitted to the Phasor Data Concentrator (PDC) at system control centers through synchrophasor networks for further analysis and control. Synchrophasor technology enables reliable and efficient power system operation but also make the system susceptible to network issues, such as connection instability and Denial of Service (DoS) attacks. Power system instability could result from modified, delayed or dropped measurement packets from PMUs in closed-loop application.

The subject of this study involves the security of smart grid synchrophasor networks (PMUs, PDCs, computers and communication systems), commonly known by the power systems society as the cyber-security of smart grids [77] [2] instead of the power systems security.

One common prejudice in smart grid study is that encryption is the solution to most network security issues, and data manipulation or data replay attacks are the most threating network attacks faced by smart grid operations. It is indeed necessary to deploy encrypted communication channels such as using the security gateways to establish Virtual Private Network (VPN) tunnels between locations, which also eliminates many cyber vulnerabilities, including data manipulation or data replay attacks. However, the deployment of encrypted communication is far from the complete solution.

1.2 Cyber Physical Systems Security of Smart Grid

In this section, the current security vulnerabilities found in literature, that are relevant to PMUs, are discussed and mapped to four general attack classes. Known network security vulnerabilities are also addressed in hopes of exposing gaps where further research needs to be conducted on their specific effects on PMU networks. These attacks and countermeasures are surveyed to provide future research possibilities.

Electricity is essential to maintaining the standard of living globally. Citizens need a reliable power grid. Increasing consumption requires either that more energy be produced or energy generation be increasingly efficient. The “smart grid” is a modern power system that uses information and communication technologies to provide efficient, reliable, and sustainable energy [110]. A smart grid allows for bidirectional flow of information and energy in transmission and distribution systems [19]. Real-time monitoring provides situational awareness of system conditions, and ensures that power is continuously available to consumers. Although real-time feedback is critical to power system operations, communications are susceptible to cyber-attacks.

Synchrophasor technology using Phasor Measurement Units (PMUs) provide feedback of the current state of the power system in real time. Synchrophasor measurements are sent to a substation/control center and stored in a database. PMUs communicate to the substation/control center using TCP/IP network connections. The use of TCP/IP makes the system vulnerable to known cyber-attacks. Stallings [101] divides attacks into four general classes: interruption, inter-

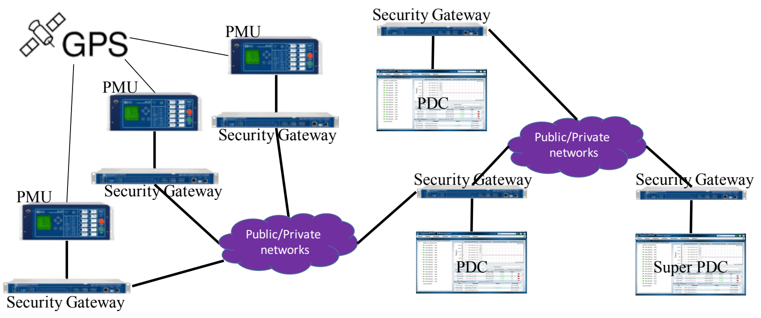


Figure 1.1: An Example Synchrophasor Network.

Table 1.1: `Documented Security Vulnerabilities`

Current PMU Attacks	***General Class of Attack***	***References to Attack***
Denial of Service	Interruption	[19–21, 61, 63, 78, 100, 103, 120, 124]
Physical Attack	Interruption	[11, 19, 21, 63]
Man in the Middle	Interception/Fabrication	[19, 21, 31, 63, 78, 80, 120]
Packet Analysis	Interception	[21, 78, 80, 103]
Malicious Code Injection	Modification	[19, 21, 31, 45, 72, 78, 100, 120, 123]
Data Spoofing	Fabrication	[11, 19, 21, 78, 96, 103, 120]

ception, modification, and fabrication. In this study, the current PMU security vulnerabilities are surveyed and mapped into four attack classes. Known network vulnerabilities are also presented under the four attack classes. Previous research, shown in Table 1.1, has investigated PMU vulnerability to Denial of Service (DoS), malicious code injection, packet analysis, physical damage, and data spoofing attacks.

Fig. 1.1 shows an example Synchrophasor network. Global Positioning System (GPS) time stamps allow measurements to be synchronized so the network can be analyzed as a whole [20]. Once the synchrophasors synchronized with the GPS, each measurement the synchrophasor sends across the network includes a time stamp. All electrical distribution line measurements are sent via a network connection to a PDC. The PDC is a regional command station for the grid [63]. The PDC stores measurements for additional processing.

1.2.1 Interruption Attacks

The simplest interruption attack is a physical attack, for example, cutting a cable. A more sophisticated attack is denial of service. DoS attackers attempt to consume systems resources, such as bandwidth, to prevent users from accessing the system. DoS attacks can use flaws in network protocols to deny access to legitimate users. One common type of DoS attack is an Internet Control Message Protocol (ICMP) attack [124]. In a Distributed Denial of Service (DDoS) attack, the attacker takes control of multiple machines that are called zombies. Zombies are used to overload the target's bandwidth, resulting in legitimate traffic being slowed or dropped.

1.2.1.1 PMU Specific DoS

Researchers in [78] tested various types of DOS attacks such as network layer attacks, Internet Control Message Protocol (ICMP) attacks, transport layer attacks, Local Area Network Denial (LAND) attacks, and teardrop attacks against the PMU network. All of these attacks take advantage of weaknesses in network protocols. The PMU network is dependent on real-time measurement data, making it vulnerable to this attack. If a malicious person were to perform this attack on multiple PMUs, all of the measurements from those PMUs would either be dropped or delayed. This could cause inaccurate predictions about the status of the transmission system, delayed mitigation of power system problems, or total failure of measurement devices along the network.

1.2.1.2 DoS Countermeasures

Even though DOS attacks remain a serious threat to power grids that incorporate PMU networks, there are limited ways to prevent or mitigate the damage caused by an attacker. One possible solution to DOS attacks on PMU networks would be to use an "air gap" network. Air gap networks provide no physical connection to the larger Internet. This type of isolation comes with the disadvantage of cost, it is costly to build separate network infrastructure for the power grid. Some of the common DOS countermeasures are the use of big pipes or large bandwidth connections to insure the network can handle the traffic. DOS traffic can also be mitigated using distributed or redundant infrastructure. Researchers in [61] discuss DDOS attacks and there countermeasures. The paper discusses the different types of flooding attacks, the methods to detect flooding attacks, and existing countermeasures. [47] proposed a mechanism to resist Denial-of-Service attacks efficiently. [62] proposed an authentication scheme can prevent DOS attacks even if attackers break servers.

1.2.1.3 Physical Attacks

A PMU can be isolated from the network by using physical attacks that damage hardware or infrastructure. This includes cutting a network connection between the PMU and PDC or sabotaging the PMU. Limiting access to critical infrastructure can mitigate this type of attack. Not included are software attacks that effect hardware.

1.2.2 Interception

Two types of interception attacks that effect synchrophasor networks are (passive) packet analysis and (active) man-in-the-middle attacks. The basis of an interception attack is to use the information communicated via the PMU and PDC to the attacker's advantage unlike an interruption attacks that stops communications.

1.2.2.1 Packet Analysis

Contents of the PMU TCP/IP packets are susceptible to packet analysis, "sniffing". Program such as Wireshark [19], allow attackers to look at the traffic being sent across the network. If encryption is not used, all information communicated can be seen by the attacker. In [103], researchers used Wireshark to analyze the synchrophasor network. The network traffic was redirected and captured, and the data was in clear text. This makes it possible for an attacker to get passwords and other information sent across the network.

1.2.2.2 Packet Analysis Countermeasures

Adding a security gateway to the network, allows the packets to be sent across a virtual private network (VPN) tunnel. Researchers in [103] found that when packets were sent through a virtual private network (VPN) tunnel, the traffic was encrypted. VPN's allow users to create a virtual network between two sub-networks that are trusted. The packets sent within the VPN are in a secure tunnel between the clients. VPN tunnels commonly use the Secure Socket Layer (SSL)/Transport Layer Security (TLS) protocol to secure the communications between VPNs. To insure a secure connection the parties on the network use X.509 certificates to authenticate users and then exchange symmetric keys. This process is supposed to provide the system with security, but may have implementation and design errors. Some of the previously successful attacks on

SSL/TLS include [21]: DNS Cache Poisoning, ARP Poisoning, Man-in-the-Middle Attacks and TLS/SSL Certificate Attacks. VPNs are essential for securing traffic, but they need to be carefully implemented and their security verified. Each of these attacks have been shown to work on networks, research still needs to be conducted to see their impacts on PMU networks. They are not a panacea; rather they are part of a reasonable security program.

1.2.2.3 Man-in-the-Middle Attacks

Man-in-the-Middle (MIM) attacks occur when an attacker poses as the other side of a legitimate protocol session to both the legitimate client and server. For example, if B and A communicate, the intruder I replaces the BA link with two links BI and IA. In a PMU network, a Man-in-the-Middle attack would occur between the PMU and the PDC. The attackers disguise themselves as the PDC to the PMU and as the PMU to the PDC. Man-in-the-Middle attacks can use methods such as route table poisoning, modified packet source and destinations, and using compromised certificates. Researchers in [24] discuss a method of using false certificates to conduct a Man-in-the-Middle attack. The paper discusses the false certificate vulnerability for a HTTPS connection. However, this method could be further utilized in any system that uses certificates to secure connections. In the PMU network, if the PDC uses X.509 certificates for authentication then a Man-in-the-Middle attack would be possible between the PDC and the PMU.

1.2.2.4 Man-in-the-Middle Attacks Countermeasures

To prevent this type of attack from occurring, the users need to authenticate the server they try to connect to [80]. One of the most common authenticate applications applied today is to employ Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol to authenticate the server. SSL/TLS provides support for user authentication based on public key certificates. Use public key certificates system to defeating man-in-the- middle attacks is theoretically a solved problem. However, in many cases [32], secrets are sent directly from clients to servers with almost every request across all layers of the network stack make this technique vulnerable. In [32], researchers developed a layered solution to solve the problem, each layer consisting of minor adjustments to existing mechanisms across the network stack. [62] proposed an authentication scheme can prevent MIM attacks even if attackers break servers.

1.2.2.5 Side Channel Attack

Protocol tunneling is widely used to add security and/or privacy to Internet applications. Recent research has exposed side channel vulnerabilities that leak information about tunneled protocols [46]. Side-channels extract information by indirectly observing implementation artifacts. For example, a significant timing side-channel vulnerability for SSH can extract the system password from interactive sessions [98].

1.2.2.6 Side Channel Attack Countermeasures

These attacks can best be countered by saturating the communications channels leaving no available bandwidth for patterns to emerge. Due to the extreme resource requirements, this approach can only be used in extreme cases [46].

1.2.3 Modification

Modification attacks look for security vulnerabilities to corrupt, highjack, or alter a legitimate process. An example of this type of attack is malicious code injection. In a malicious code injection attack, the attacker inserts new instructions into a piece of code to alter its form of execution. One common type of modification attack is Structured Query Language (SQL) injection. SQL is a standardized language for managing databases. This form of attack allows the attacker to insert a script into a piece of code to alter the database.

1.2.3.1 PMU Specific Malicious Code Injection

PMUs continuously send measurements to the PDC. Before measurements are sent across the network, the PMU sends a configuration message to the PDC to specifying the data table to set up in its database [103]. The PDC does not authenticate the configuration message. Instead, it creates new tables specified in the configuration message. This leaves the system vulnerable to code injection. Researchers in [123] describe two insertion attacks: code-injection and return-oriented programming. In code-injection, the attacker directly inserts shell code (a set of malicious instructions) into the program. Return-oriented programming reuses binary code already present in the system as shell code. Shell code is the software exploit payload. Either attack can send malicious instructions to the database management system, to add, delete, or modify the database, or take control of the system. Since all the data sent to the PDC is stored in a database, PMUs are particularly susceptible to SQL injection attacks [31]. The SQL injection vulnerability most frequently comes when queries are formulated from user inputs. This attack occurs if user input is not properly validated before inclusion in the SQL query [45].This attack can add, delete, or modify the database contents and therefore directly modify, corrupt, append, and/or disable PMU measurement information. The attacker can make the condition of a transmission system appear to be the opposite of its actual state. For example, if there is an issue with a transmission bus or line, an attacker can modify measurements to indicate it is normal, which could put the power system at risk for an outage.

1.2.3.2 Countermeasures

In [123] the authors suggest detecting code insertion by monitoring program performance. If an attacker has inserted shell code into a program, the program should behave unusually. Another method to overcome malicious code injection attacks on PMU networks is to use randomized encryption keys when transferring messages [50]. Only the PMU and the PDC would have the key. If the attacker does- not have the random key, then they will be unable to decipher the instructions. In modern operating systems, code injection is made difficult by either randomizing the system address space or monitoring the stack to detect buffer overflows [21]. Both of these approaches could be relevant to PMU network implementation. [118] proposed a geographical routing protocol allows peers in a Smart Grid system to adjust their interaction behaviors based on the trustworthiness of others. The protocol can avoid packet drop attack and data spoofing attack by using a safe forwarding list. [62] proposed an authentication scheme can prevent attackers to resent previously intercepted login messages even if attackers break servers. To counteract SQL injection attacks: either check inputs for characters that can be abused, or use parameterized statements that force user inputs to follow a static template. These templates only allow certain inputs to be translated into queries. For further prevention, databases should also have strict access controls for allowing users to modify or manipulate data.

1.2.4 Fabrication

Fabrication attacks occur when attackers create a fictitious asset on the network. Data spoofing and the previously mentioned Man-in-the-Middle attack both are fabrication attacks. Both send fabricated data across the network. The PMUs continuously send data to the PDC along with instructions on how to set up database tables. Man-in-the-Middle and data spoofing attacks can send false measurements. This differs somewhat from modification attacks, since the data can be not related to current measurement data. If the PDC collecting the data does not authenticate PMUs on the network, the PDC may accept fabricated data.

1.2.4.1 PMU Data Spoofing

Since the control centers will depend on the feedback from the PMUs to make decisions about the current condition of the electrical grid, the accuracy of the data is important. Data spoofing gives the control center forged data instead of actual data. This can be detrimental to the stability and reliability of the grid. In [100], researchers inject false data into the system. This data can be either PMU measurements or measurement time stamps. In [96], authors spoof the GPS time stamps for measurements. Altering measurement times can render measurements useless. For example, readings can be reordered to make capacity reduction seem to be capacity increase, etc.

1.2.4.2 Data Spoofing Countermeasures

In [120], authors show that data spoofing has very little effect when it is limited to a single data feed. To mitigate data spoofing the grid can use multiple PMUs to monitor the same electrical transmission bus or line, which would make spoofing more complicated. The use of redundant measuring devices is supported by [11]. Those authors use redundant smart meters, but the same idea is relevant to PMUs. In [41], authors proposed a multi-antenna based algorithm to detect time stamp spoof. The GPS signals from two receiving antennas are fed to two independent processing units. The differences of the carrier-signal-to-noise ratio (ClNo) from these two receiving paths can be exploited as probabilistic measures. Using this, the change in the distribution of sequential samples is detectable. [118] proposed a geographical routing protocol, which can avoid packet drop attack and data spoofing attack by using a safe forwarding list.

1.2.5 Access Control

With insufficiencies of the legacy power grid communications protocols, increased data generation and communications bring about new challenges in access and collecting the data. Some recent researches presented access control methods for Electric Power Synchrophasor Networks, which could provide both security and efficiency [121], [119], [12], [27]. In [108], researchers presented a data collector protocol, which can verify the integrity while they are not given access to the content. [62] proposed an authentication scheme can prevent Off-line dictionary attacks.

1.2.6 Countermeasures After Attacks Occurred

Some other researches are focused on attack detections and countermeasures after attacks occurred. [48] proposed game payoff formulae. A decision analysis can then be obtained by applying backward induction technique on the game tree. [64] provided a dynamical systems context for formulating distributed multi-switch strategies. [10] presented intrusion detection methods used to analyze the measurement data to detect any possible cyber-attacks on the operation of smart grid systems. [39] analyzed the cyber-attack related time delay issues, and deals with the minimization of negative effects of such delays in smart gird system.

1.2.7 Summary of This Section

The "smart grid" is a modern power grid that uses information and communication technology to provide reliable, efficient, and sustainable power. To upgrade the current power system to a smart grid, PMUs are being installed. Due to limited amount of research on how these attacks effect PMU networks, further research needs to be conducted to determine their effects.

1.3 Objectives

The objectives of this dissertation are list as follows:

- Investigate side-channel vulnerabilities in encrypted synchrophasor network communication and its consequences.
- Investigate and evaluate mitigation methods against the above vulnerabilities.
- Perform game theory analysis to the synchrophasor network security.

1.4 Contributions to Date

The contributions of this dissertation are list as follows:

- Comprehensive reviews of synchrophasor network security.
- Packet size side-channel and inter-packet timing side-channel are discovered in encrypted synchrophasor network communication.
- Protocol transformation from botnet traffic to synchrophasor protocol is implemented.
- DoS attack that exploits side-channel vulnerabilities on tie-line bias control using Automatic Generation Controller are investigated and consequences outlined.
- Countermeasures against above DoS attacks using Cellular Computational Network [112] based Virtual Synchrophasor Network are developed and shown to be most effective.
- Real-time DoS attacks and Virtual Synchrophasor Network countermeasures are performed.
- The resiliency test of Virtual Synchrophasor Network by mitigating multiple DoS attacks is performed.
- Border Gateway Protocol (BGP) hijacking attack detection on PMU data stream are proposed and evaluated.
- Game Theory Analysis is performed to find the best strategy under different circumstances.

1.5 Organization of this Dissertation

This dissertation begins with introducing necessary background knowledge in Chapter 2.Then, Chapter 3 focuses on the side-channel vulnerabilities of PMU; Chapter 4 explores the feasibility of protocol transformation. DoS attacks based on this vulnerability is presented in Chapter 5. A mitigation method of above DoS attack is proposed in Chapter 6. The resiliency of the mitigation method is studied in Chapter 7. The best strategy under different circumstances are identified in Chapter 9 Finally, a conclusion is given in Chapter 10.

1.6 Publications

1. A. Keenan, R. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, pages 118–132, Cham, 2013. Springer International Publishing

2. J. E. Padilla, M. J. Patitz, R. Pena, R. T. Schweller, N. C. Seeman, R. Sheline, S. M. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: Fuel efficient computation and efficient assembly of shapes. In G. Mauri, A. Dennunzio, L. Manzoni, and A. E. Porreca, editors, *Unconventional Computation and Natural Computation*, pages 174–185, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg

3. C. Beasley, X. Zhong, J. Deng, R. Brooks, and G. Kumar Venayagamoorthy. A Survey of Electric Power Synchrophasor Network Cyber Security. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pages 1–5, Oct 2014

4. X. Zhong, A. Ahmadi, R. Brooks, G. K. Venayagamoorthy, L. Yu, and Y. Fu. Side channel analysis of multiple pmu data in electric power systems. In *2015 Clemson University Power Systems Conference (PSC)*, pages 1–6, March 2015

5. X. Zhong, P. Arunagirinathan, A. Ahmadi, R. R. Brooks, and G. K. Venayagamoorthy. Side-channels in electric power synchrophasor network data traffic. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, CISR '15, pages 3:1–3:8, Oak Ridge, Tennessee, USA, 2015. ACM

6. X. Zhong, L. Yu, R. Brooks, and G. K. Venayagamoorthy. Cyber security in smart dc microgrid operations. In *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, pages 86–91, June 2015

7. X. Zhong, Y. Fu, L. Yu, R. Brooks, and G. K. Venayagamoorthy. Stealthy malware traffic - not as innocent as it looks. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 110–116, Oct 2015

8. A. Keenan, R. Schweller, M. Sherman, and X. Zhong. Fast arithmetic in algorithmic self-assembly. *Natural Computing*, 15(1):115–128, Mar 2016

9. Y. Fu, Z. Jia, L. Yu, X. Zhong, and R. Brooks. A covert data transport protocol. In *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 1–8, Oct 2016

10. X. Zhong, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. Denial of service attack on tie-line bias control in a power system with pv plant. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(5):375–390, Oct 2017

11. R. R. Brooks, L. Yu, Y. Fu, G. Cordone, J. Oakley, and X. Zhong. Using markov models and statistics to learn, extract, fuse, and detect patterns in raw data. In N. S. Rao, R. R. Brooks, and C. Q. Wu, editors, *Proceedings of International Symposium on Sensor Networks, Systems and Security*, pages 265–283, Cham, 2018. Springer International Publishing

12. X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. Phasortoolbox – a python package for synchrophasor application prototyping. In *2018 Clemson University Power Systems Conference (PSC)*, 2018

13. X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. A virtual synchrophasor network for power system resiliency under concurrent dos attacks. *IEEE Transactions on Power Systems*. Submitted for peer review

14. X. Zhong, G. K. Venayagamoorthy, and R. Brooks. Bgp hijacking detection in synchrophasor network. *IEEE Transactions on Power Systems Letter*. Submitted for peer review

1.7 Summary

The original intention in deploying synchrophasor devices is for reducing the system operation cost while increasing both revenue and service quality. However, the cost of a guaranteed delivery of real-time PMU measurement data is expensive.

Traditional power grid controls do not address the new challenges in the smart grid, and existing network security solutions do not fulfill smart grid operation requirements. Little research has been carried out at this intersection to identify the potential problems, evaluate their impact on synchrophasor networks and addressing those vulnerabilities.

This dissertation gives a brief overview of the area and then profoundly dig into one specific problem. Network security vulnerabilities that are unique to smart grid operations are identified. Such vulnerabilities are critical to the smart grid operation but easily overlooked without enough understanding of both areas. The consequence of the vulnerabilities is presented. The solution specifically for this problem is explored and evaluated. Finally, the perspective of this discussion is expanded with game theory analysis to give a broader view of the network security research in the smart grid.

The study and results presented in this dissertation reduced the gap in the field, effectively reduces both the cost and risk of adopting PMU in closed-loop applications and further enhancing the reliability of the smart grid.

Chapter 2

Background

2.1 Overview

In this chapter, the terminology and necessary background for this dissertation are presented.

2.2 The Synchrophasor Protocol

Since PMUs are widely distributed and used by many electric power utilities, a standard protocol ensures consistent data storage and network communications between PMU networks. The current PMU communications standard is the IEEE C37.118 protocol [52], which defines synchrophasor data conventions, measurement accuracies, and communications formats . To adhere to the IEEE C37.118 protocol, the synchrophasor has to recognize five frame types: Data frame (binary), Two configuration frames (binary), Header frame (ASCII) and Command frame (binary). The configuration and header frames describe the synchrophasor configuration, the data frame contains measurements, and the command frame tells the PMU when to start and stop taking measurements. Once measurements have been collected, they are processed using a phasor data concentrator (PDC) such as the open-source OpenPDC [7]. OpenPDC takes measurements and sorts them by their time stamps. The measurements are archived in a database using their time stamps.

The structure of the protocol is illustrated in Figure 2.1. The mapping of the protocol in a IP packet is shown in Figure 2.2 A typical sequence diagram of communications between OpenPDC and one PMU in the application layer is shown in Figure 2.3. OpenPDC starts a new session by sending a command asking the PMU to stop current data transmission, followed by a request for a configuration frame, which describes the format of PMU measurement packets used for this transmission. After receiving the configuration frame from PMU, OpenPDC sends the PMU a start command. Once the measurement data transmission starts, it will not stop until a stop command is received or the connection is lost. OpenPDC acknowledges each measurement data packet it receives.

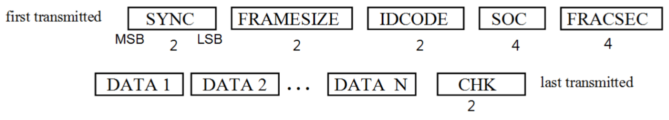


Figure 2.1: Example of frame transmission order. This figure is from IEEE Standard C37.118.2 [52]

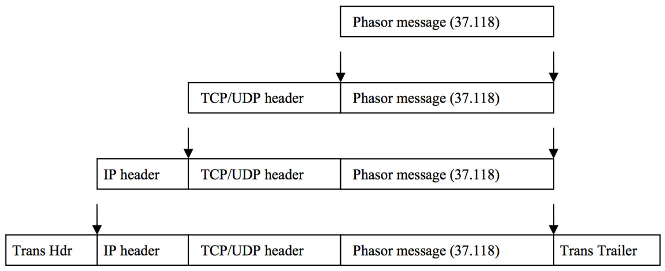


Figure 2.2: Mapping of IEEE C37.118 data into a TCP or UDP packet A transport layer header and trailer are shown, as it would be when using Ethernet. This figure is from IEEE Standard C37.118.2 [52]

2.3 Security Gateways

Security gateways establish VPN tunnels between private networks through public network. The security gateways encrypt and decrypt packets. The deployment of security gateways reduces the risk of sending critical data through an unsecured network. In the power system, security gateways are used in critical connections, such as, communications between PMUs and PDCs. Each packet sent from a PMU or PDC contains a TCP/IP header and payload, which will be encrypted by security gateways. In order to send encrypted data through the conventional network, encrypted data will be attached to a new IP and Encapsulating Security Payload (ESP) header as the encrypted

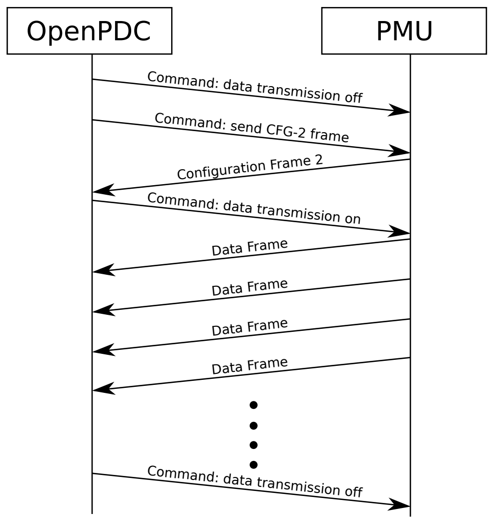

OpenPDC
PMU
Command: data transmission off
Command: send CFG-2 frame
Configuration Frame 2
Command: data transmission on
Data Frame
Data Frame
Data Frame
Data Frame
Command: data transmission off

Figure 2.3: Observed sequence diagram of Synchrophasor protocol in application layer

payload. After encryption, any data in the original packet like packet destination, source, and original payload are encrypted. The source and destination IP address of the encrypted packets are usually the IPs of two security gateways.

Security gateways use the Internet Protocol Security (IPsec) protocol to ensure communications and interoperation. As a framework protocol to secure the connection, IPsec mainly provides authentication, verification and confidentiality. Authentication means checking if the data is from where it claims to be. Verification means checking whether it has been altered. Confidentiality means checking if the data is visible to third parties in transit [58]. IPsec protocols mainly provide authentication, verification and confidentiality. Authentication ensures the data is from where it claims to be, verification checks whether the received data has been altered, and confidentiality ensures the data been transferred is not visible to third parties [58]. Advanced Encryption Standard (AES) and Triple Data Encryption Algorithm (TDEA or 3DES) are commonly used for encryption. The authentication and encryption methods make it practically impossible to decrypt and modify an encrypted packet without proper keys. Fig. 2.4 demonstrates the process that encrypts a PMU measurement package by a security gateway.

The encryption algorithms used in IPsec have five modes, which are: ECB (Electronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher FeedBack), OFB (Output FeedBack), and CTR (Counter) [49]. At the beginning of an encrypted connection session, two security gateways

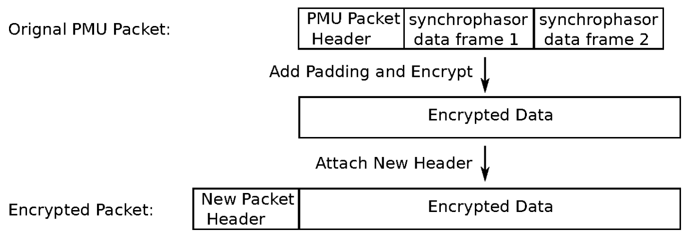


Figure 2.4: The PMU measurement packet encryption.

will negotiate the algorithm used for this session. In the experiments, it is found that CBC is used in most cases. IPsec-secured links are defined in terms of Security Associations (SAs). Each SA is defined for a single unidirectional flow of data, and usually from one single point to another. And it produces traffic distinguishable by some unique selectors [58], which means the disruption of one connection in one pair of nodes should not affect other connections within the same secure tunnel.

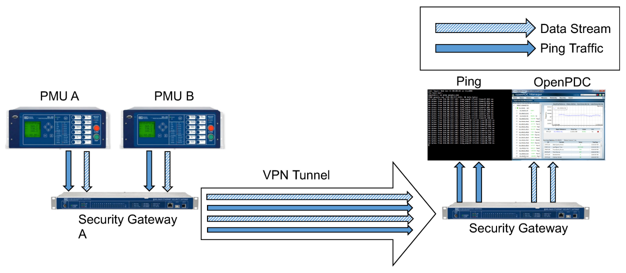
The security gateway quickly encrypts each packet and barely alters the timing features of the original traffic. Similarly, the packet size of encrypted packets is correlated to the size of the original packets. By analyzing the packet streams rather than looking at a specific packet, additional information can be inferred.

2.4 Side-Channel Analysis

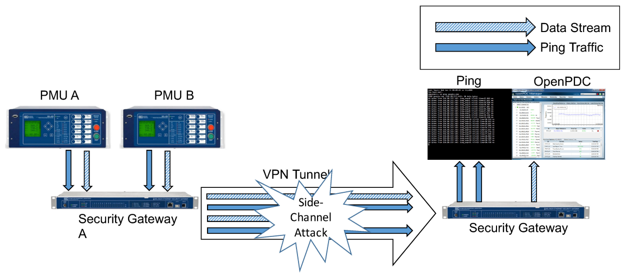
Although the encryption systems are mathematically difficult to crack, but the physical implementation of cryptographic systems leak information. Side-channel analysis extract information by observing implementation artifacts. For example, the power consumption of a computer is correlated to different operations. By monitoring power consumption, the activities of a program can be inferred [59]. For network enabled programs, it is possible to infer program operations by analyzing the network traffic. In [98], a timing side-channel vulnerability of SSH has been used to extract the system password from interactive sessions.

A VPN is a tunneled, and encrypted connection established between two private networks though public network. Packets transferred through a VPN tunnel have the same source and destination IP. The real source and destination IPs for each packet are encrypted and can not be seen by the third party. Fig. 2.5a shows a VPN tunnel that connects two private networks, multiple traffic streams are transmitted though the tunnel including two PMU measurement data streams.

In [127], PMU measurement sessions are detected even when all packets are encrypted in



(a)



(b)

Figure 2.5: An example of a DoS attack in a VPN tunnel: (a) A VPN network carries Ping traffic and data streams from two PMUs to control center; (b) During a DoS attack, a PMU data stream within the VPN tunnel is identified and dropped.

a VPN tunnel. Furthermore, packet size side-channels and inter-packet timing side-channels can differentiate between the sources of PMU packets when multiple PMU data streams are transmitted in a single encrypted VPN tunnel. In this case, an encrypted VPN tunnel is established between two subnets. Multiple PMUs in one subnet are connected to a PDC on the other subnet through this encrypted VPN tunnel. Mapping the encrypted packets to PMUs based on inspect the content of each packet is impossible. However, by using side-channel analysis, the most probable mapping between packets and PMUs can be identified and a specific PMU data stream can be blocked by an attacker without interfering with other traffic in the same VPN tunnel, shown as Fig. 2.5b. This attack is easy to perform but difficult to diagnose. The attack only identifies and blocks the PMU data transmission stream. Other traffic from PMUs e.g. PING or HTTP is not blocked, thus making the PMU appear to be connected to the network while not transmitting any data.

2.4.1 Hidden Markov Models

In previous work [13], detecting PMU measurement sessions was possible even when all packets are encrypted by performing side channel analysis on timing delay between every two packets. In that study, a Hidden Markov Model (HMM) using packet timing delays is built, where packets are captured from encrypted PMU traffic between two security gateways. Then HMM inference is used to recognize encrypted PMU traffic. HMMs are widely used for pattern recognition and detection [65].

2.5 Man-In-The-Middle Attack

A MITM attack is an cyber-attack that falls in to the category of interception attack [14]. Generally, it occurs anywhere between two legitimate nodes in a network. For example, if A and B are two legitimate nodes connected by connection AB, attacker C can take over the connection by replacing AB with connection AC and connection CB. So attacker C can intercept and modify all communications between A and B. In PMU-PDC communications, attackers disguise themselves as PMU if the packets are sent from PDC to PMU, and vice versa. It is the same for communications between a PDC node and a super PDC. This vulnerability and countermeasures are discussed in [14]. Without using security gateways or any other security countermeasures, an attacker could view, modify and redirect measurement packets easily. By using security gateways to encrypt the traffic, the chance of packets being manipulated is practically eliminated.

The MITM attack is very easy to apply and can be performed in many different ways. On the Internet, packets are redirected to the destination based on the Board Gateway Protocol (BGP) routing mechanism. BGP route hijacking is very easy to be applied and sometimes caused by human mistakes. In March 2015, traffic from Texas, US to the UK was redirected to Ukrainian and Russian telecoms for five days. Multiple sites were affected, including UK's official mail service, and nuclear weapon managing and delivering system [113]. On April 26, 2017, Russian telecom Rostelecom hijacked traffic belonging to MasterCard, Visa, and many other financial services companies for several minutes [107]. In January 2017 Iran's act of pornography censorship mistakenly redirected traffic from Russia to Hong Kong to blank pages [114]. In a local network, MITM attack can be performed in many ways such as Address Resolution Protocol (ARP) spoofing, Domain Name System (DNS) poisoning, Media Access Control (MAC) address spoofing and so on.

2.6 Summary

The terminology and background introduced in this chapter will be used in all following chapters.

Chapter 3

Side-Channel analysis of PMU Traffic

3.1 Overview

Synchrophasor devices such as Phasor Measurement Units (PMUs) are essential power system components for real-time monitoring, analysis and control. The data from several PMUs are sent to a Phasor Data Concentrator (PDC) via network connections. The PMU data are used for monitoring electric power network operations and performing real-time decision-making and control. In previous research [14], it has been pointed out that by using security gateways and Virtual Private Network (VPN) tunnels to encrypt traffic, many possible vulnerabilities can be eliminated. However, these communications are still vulnerable to side-channel analysis. In previous studies [126], side-channel inter-packet timing delays are used to map the packets to different PMUs.

In this study, packet size side-channel and inter-packet timing delay side-channels are used to differentiate the source of PMU packets in an encrypted VPN tunnel containing multiple PMU data streams. Two PMUs are connected to a PDC through an encrypted VPN tunnel. Packets transferred in this VPN tunnel have unified source and destination IP addresses for each direction and the original source IP is encrypted. Mapping packets in an encrypted VPN tunnel to PMUs based on the clear text header in each packet is impossible. However, by using side-channel analysis, the most probable mapping between packets and PMUs can be identified. The proposed methods can identify and block traffic from specific PMU in a power system.

3.2 PMU Packet Size Side-Channel Analysis

In this section, the methods of identifying the packet source in an encrypted VPN tunnel using packet size side-channel is explained, where the VPN tunnel is transporting packets from multiple PMUs. The following subsections are organized as follows: Section 3.2.1 introduces the experimental network configurations and give a brief overview of this method; Section 3.2.2 describes the method to perform packet identification using packet size side-channel; Section 3.2.3 describes methods used to evaluate the accuracy of these method.

To be consistent, 'PMU_0' and 'PMU_1' refer to two PMUs with different model numbers. 'PMU_A' and 'PMU_B' refer to two PMUs, the first packet in the main stream is assigned to 'PMU_A', the identity of 'PMU_A' is unknown; it is simply a label to distinguish between 'PMU_A' and 'PMU_B'.

To identify PMU models using packet size side channels, it is needed to build a map between encrypted or unencrypted packet size and its corresponding model number. In this experiment, there are only two PMUs sending packets through one VPN tunnel and no other applications are sending packets. To differentiate PMU models, no map is needed.

3.2.1 Experimental Setup and Procedure

All experiments and measurements are carried out in the Real-Time Power and Intelligent Systems (RTPIS) Laboratory in Clemson University [92]. At the RTPIS lab, simulating power systems in real-time and collecting data using PMUs is possible. Real Time Digital Simulator (RTDS) is used to build and perform real-time simulation of a power system. The hardware PMUs are used to take measurements. There are hardware and software PDCs, and both of them can collect data from PMUs. In this study, software, OpenPDC, is used to collect data from hardware PMUs. Also, two hardware security gateways are set up to encrypt and decrypt packets between PMUs and PDCs. Wireshark running on computers connected to the hubs in the network is used to collect packets. The network configurations are shown in Figs. 3.1, 3.2 and 3.3.

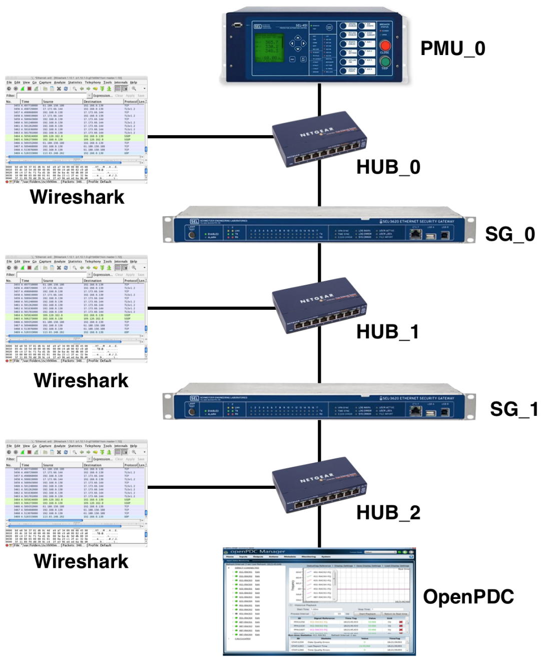


Figure 3.1: Network configuration for packet size sampling of PMU_0.

The first two network configurations (Figs. 3.1 and 3.2) are used to get packet size and inter-packet timing delay samples. The third network configuration (Fig. 3.3) is used to perform packet identification and evaluate its accuracy. PMUs and PDCs do not encrypt or decrypt packets, so the packets collected at 'HUB_0' and 'HUB_2' in three Figs. 3.1, 3.2 and 3.3 are in clear-text and the payload and IP source/destination of each packet are observable. Packets are encrypted between security gateway SG_0 and security gateway SG_1 shown in Figs. 3.1, 3.2 and 3.3. It is noticed that variation in power grid dynamics has no affect on the packet sizes between different PMUs.

Data collected under the network setting shown in Figs. 3.1 and 3.2 is used to build the map between packet sizes and PMU model numbers. The histograms of packet sizes can be found in Appendix 3.4.1. After getting the mapping of packet sizes to PMU model numbers (Table 3.3), both PMUs are added to the network, as shown in Fig. 3.3. Packet sizes to PMU model numbers map is used to determine the source of each packet collected at the hub between two security gateways, shown as 'Hub_1' in Fig. 3.3. Finally, the accuracy of the identification method is discussed.

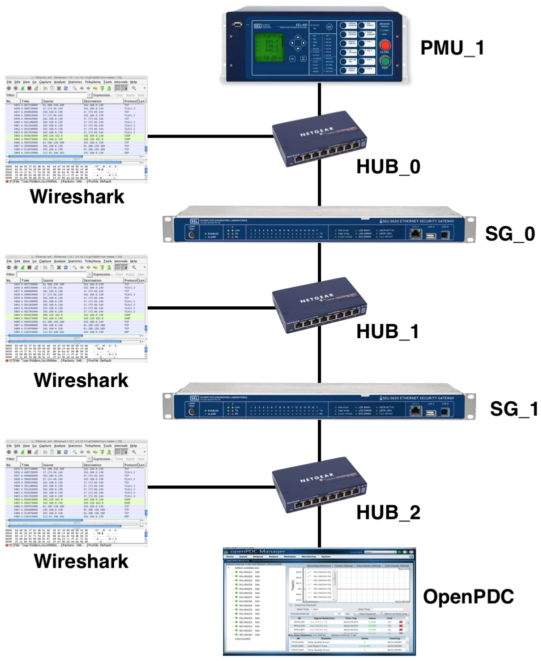


Figure 3.2: Network configuration for packet size sampling of PMU_1.

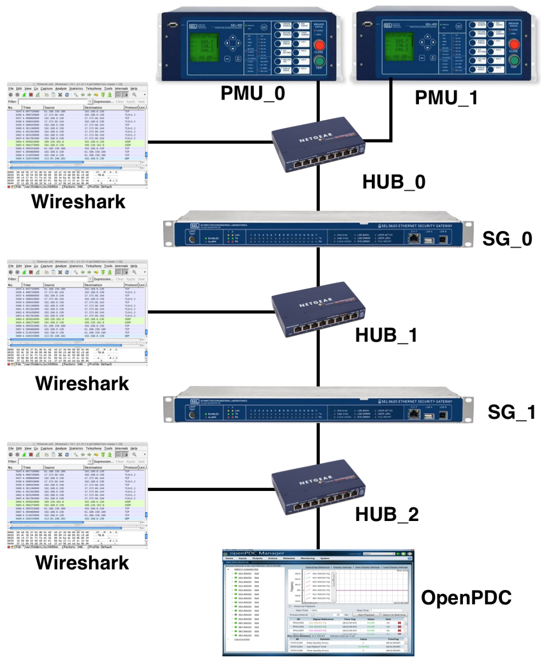


Figure 3.3: Network configuration for packet separation.

3.2.2 Packet Size to PMU Model Map

The packets generated from different PMU models have different sizes. There are two types of PMU packets essential to power grid operation: PMU measurement and PMU configuration packets. Each PMU measurement packet is composed by a fixed length header followed by synchrophasor data frames. The size of each synchrophasor data frame is related to PMU models, where different models have different synchrophasor data frame sizes. The size of a PMU measurement packet can be expressed as Eqn. 3.1

$$S_{PMU} = 54 + n \times S_{DATA} \tag{3.1}$$

Where S_{PMU} is the length of PMU measurement packet, 54 is the byte size of the header, n is the integer number of data frames contained in this packet and S_{DATA} is the length of synchrophasor data frame for this PMU model. From the data captured, a map from synchrophasor data frame size to PMU model numbers is shown as Table 3.1.

Table 3.1: Map of Synchrophasor Data Frame Sizes to PMU Model Numbers

Synchrophasor Data Frame Size	PMU Model Number
154	PMU_0
284	PMU_1

A PMU configuration packet is sent at the beginning of each PMU-PDC session. The map of PMU configuration packet size to PMU model number is shown as Table 3.2.

Table 3.2: Map of PMU configuration packet size to PMU Model Numbers

Encrypted PMU Configuration Packet Size	PMU Model Number
428	PMU_0
1008	PMU_1

Encrypted by security gateways, each encrypted packet is composed by a fixed length of header and ESP payload. The encryption algorithm used by security gate ways requires the size of data is an integer multiple of 32 bytes. Therefore, a variable size of "padding" is attached to extend the payload data. The relation between clear-text packet size and cypher-text packet size can be expressed as Eqn. 3.2

$$S_{cypher} = 54 + \lfloor \frac{S_{clear}}{16} + 1 \rfloor \times 16 \tag{3.2}$$

Where S_{cypher} is the size of cypher-text packet, 54 is the byte size of header and S_{clear} is the size of clear-text packet.

In most of the cases, each PMU packet contains one synchrophasor data frame. When packets are lost, PMU will resend packets contains multiple synchrophasor data frames. The number of synchrophasor data frames is usually between 1 to 10. Thus, a map between clear-text PMU packet sizes, cypher-text packet sizes and PMU model numbers is inferred as Table 3.3.

Table 3.3: Map of Packet Sizes to PMU Model Numbers

Packet type	Number of Synchro -phasor Data Frames	Clear-text Packet Size	Cypher-text Packet Size	PMU Model Number
C		428	486	PMU_0
M	1	208	278	PMU_0
M	2	362	422	PMU_0
M	3	516	582	PMU_0
M	4	670	726	PMU_0
M	5	824	886	PMU_0
M	6	978	1046	PMU_0
M	7	1132	1190	PMU_0
M	8	1286	1350	PMU_0
M	9	1440	1510	PMU_0
M	10	1594	1654	PMU_0
C		1008	1078	PMU_1
M	1	338	406	PMU_1
M	2	622	678	PMU_1
M	3	906	966	PMU_1
M	4	1190	1254	PMU_1
M	5	1474	1542	PMU_1
M	6	1758	1814	PMU_1
M	7	2042	2102	PMU_1
M	8	2326	2390	PMU_1
M	9	2610	2678	PMU_1
M	10	2894	2950	PMU_1

'M' denotes for Measurement packet, 'C' denotes for Configuration packet

This allows the PMU model numbers to be inferred from encrypted traffic in near real-time.

3.2.3 Comparison Between Encrypted and Unencrypted Data Streams

In this study, different PMU models are used and no other hardware in the network produces packets of the same size. Encrypted packets can be mapped to specific PMU models using the packet size side-channel with 100% accuracy rate. If the network contains packets have the same sizes with

PMU packets but are not from PMU, the accuracy rate should be lower.

3.3 Inter-Packet Timing Delay Side-Channel Analysis

In this section, the method of identifying packet source in an encrypted VPN tunnel using inter-packet timing delay side-channel is explained, where the VPN tunnel is transporting packets from multiple PMUs. The following subsections are organized as follows: Section 3.3.1 introduces the network configurations and gives a brief overview of the method; Section 3.3.2 and 3.3.3 describe the two steps of this method to perform packet separation and section 3.3.4 describes the method used to evaluate the accuracy of the proposed packet separation method.

3.3.1 Experimental Setup and Procedure

The experiments in this section using similar settings to Section 3.2. First, data is collected under the network setting shown in Figs. 3.1 and 3.2 to infer the HMM model for a single PMU, as well as, obtain the peak timing delays.

After getting peak timing delays, shown as Fig. 3.5, and the HMM model for a single PMU, shown as Fig. 3.6, both PMUs are added to the network, shown as Fig. 3.3. Algorithm 1 (Section 3.3.3) along with HMM and peak timing delays are used to determine the source of each packet collected at the 'Hub_1' between two security gateways, shown in Fig. 3.3. Finally, the accuracy of this method is checked by comparing the source of each encrypted packets inferred with the source of each unencrypted packets captured simultaneously at the 'Hub_0' on PMU side, shown in Fig. 3.3.

3.3.2 HMM for Single PMU Recognition

The HMM construction algorithm can be found [65], which infers the HMM structure and state transition probabilities from a sequence of symbolized observations. To infer the HMM structure, a sequence of symbols with length L is needed. An extension [95] is used to determine the parameter L and derive HMMs with no *a priori* information. Starting from 1, by gradually increasing L, the HMM for each value of L is constructed. A symbol-to-state mapping is built to check if the model structure stabilizes. If the inferred HMM stabilizes, the model is declared correct [95]. This occurs when with no additional statistical relevant information could be gained with larger L.

If the HMM is built from an insufficient amount of observation data, it will not statistically represent the underlying process. A model confidence test is carried out to determine if the constructed model matches the underlying process with a given level of statistical significance [122]. This approach also calculates a lower bound on the number of samples needed for building a correct model. If the number of input samples is less than the bound, more data is collected. New models inferred with more data are still checked for the confidence. This approach allows us to remove the effect of noise in the HMM inference [65]. The HMM inference approach is shown in Fig. 3.4.

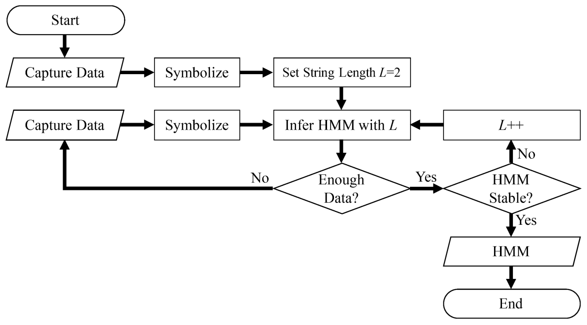


Figure 3.4: HMM Inference Process.

The first experiment collects the inter-packet delays of same PMU sent through an encrypted VPN tunnel under different scenarios. The histogram of inter-packet delays collected from a single PMU traffic is shown in Fig. 3.5, in which there are three peaks along the x-axis. The leftmost peak is omitted as it is considered insignificant compared with the other peaks during HMM inference. It is noticed that variation in power grid dynamics doesn't affect the selection of symbols and there is no significant change in peak timing delays under different power grid dynamics. The HMM built from the symbolized data sequence is given in Fig. 3.6.

3.3.3 Packet Separation Algorithm

This section considers the scenario of multiple PMUs sharing the same encrypted tunnel. Assume that the HMM used by the target PMU and the peak timing delays are known, as well as the number of PMUs. A packet separation algorithm is proposed to identify the packets generated by multiple PMUs [126]. A sub stream is extracted recursively from a given number of packets collected in the small time interval. The packets in the same sub stream should come from the same PMU. In each recursion, all the possible combinations of the packet sources are considered. For

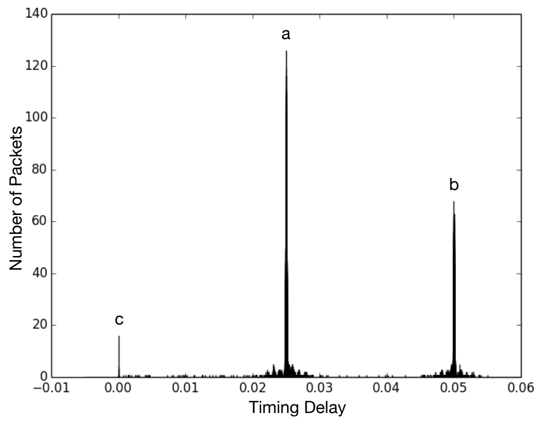


Figure 3.5: A typical histogram of encrypted single PMU traffic captured in one hour. The horizontal axis indicates timing delay, the vertical axis indicates the number of packets. Peak value of “a” is 0.024978s and peak value of “b” is 0.050002s. “c” is been trimmed during HMM inference.

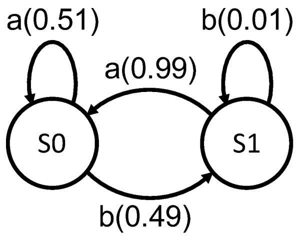


Figure 3.6: HMM of one PMU in encrypted VPN tunnel.

each combination, packets for each source are used to build sub strings. For each sub strings, the optimum mapping between the observed delays and the peak values is determined.

For delays generated by the target PMU, each of them should close to one of the peak values. If a delay is not close to any peak value, that indicates that the packets do not belong to same PMU. The square of the sum of the minimum squared value of deviations for each sub-stream in each combination is compared and the sub-stream with the minimum value is selected. The packet in the center of this interval has the highest credibility and the algorithm only takes this packet as the result and ignores the results for other packets. The operation is repeated in a sliding window over time to identify the packets sent by the target PMU.

In the simulation, the source of each packet are determined in near real-time and the average time to process each packet is smaller than the timing delay of two PMUs. An Field Programmable Gate Array (FPGA) could be used if the algorithm is used in a heavy traffic condition.

Inputs for the algorithm are: an array of packets’ arriving time A; Length of history to look

Algorithm 1 Packet Separation Algorithm

```
1: for $i = 1 \rightarrow 2^l$ do
2:     $b_i = convert\_to\_base\_2(i)$
3:     ▷ ($b_i$ is a base 2 number)
4:     $c_i = get\_and\_label\_l\_elements(l, A, b_i)$
5:     for $j = 1 \rightarrow 2$ do
6:         $c_{i_j} = substream\_from\_source\_j(c_i, j)$
7:         for $k = 1 \rightarrow number\_of\_items\_in\_c_{i_j}$ do
8:             $d_{i_{j_k}} = timing\_delay\_between\_c_{i_{j_k}} and\_c_{i_{j_{k-1}}}$
9:             for $m = 1 \rightarrow number\_of\_items\_in\_P$ do
10:                $s_{i_{j_{k_m}}} = (d_{i_{j_k}} - P_m)^2$
11:                $s_{i_{j_k}} = min(s_{i_{j_k}}, s_{i_{j_{k_m}}})$
12:            end for
13:            $s_{i_j} += (s_{i_{j_k}})^2$
14:        end for
15:        $s_i = min(s_i, (s_{i_j})^2)$
16:    end for
17:    $retrieve\_c_{i_y}\_according\_to\_s_i$
18:    $s\_min = min(s\_min, s_i)$
19: end for
20: $retrieve\_c_{x_y}\_according\_to\_s_{min}$
21: if $c_{x_y}\_accepted\_by\_HMM$ then
22:     return the label of the packet in the middle of $c_{x_y}$
23: else
24:     $retrieve\_c_{x_y}\_according\_to\_s_{second\_min}$
25:     redo HMM check
26: end if
27: remove packets in $c_{x_y}$
28: redo Algorithm 1 for the rest packets
```

at l (for simplicity, l is an odd number); Sample peak values list P; A HMM that can recognize target traffic.

3.3.4 Comparison Between Encrypted and Unencrypted Data Streams

For side-channel using inter-packet timing delays, another method is used to check the accuracy rate. Since there is no direct way to identify the source of each encrypted packet, the approach only compares the combinations and permutations of obtained packets' stream with the unencrypted stream captured at 'HUB_0', as shown in Fig. 3.3. To be more specific, if the obtained stream is 100% correct, and projections are only created between each pair of encrypted packet and its corresponding unencrypted packet. Then the projection from the set of the sources of inferred packets to the set sources of unencrypted packets is an injective function. To calculate the accuracy rate, the 100% correct injective function is built as a sample. Then among all packets pairs, the number of packet pairs that match the sample injective function are divided by the number of all packet pairs.

In practice, the so-called "Head" packets are found in both encrypted and unencrypted streams. Head packets are defined as several continuous packets that have the same combination in both encrypted and unencrypted streams. Head packets are assumed to be the packets that are corresponding between encrypted and unencrypted streams. Due to the low complexity of PMU traffic, it is relatively easy to find a wrong Head, which the combination are the same but the packets are not actually corresponded, if less than 500 packets are examined. A wrong Head may result in a 90% match rate in its following packets. In the experiment, it is found that the accuracy rate of the method is around 99%, which a Head with length more than 500 is used. So, in this study, 1000 packets are examined to determine the Head in each stream. It is said that the Head is found until all 1000 packets in both streams have the same combination; otherwise, it is said there is not a match. After finding the Head, the packets that was mistakenly inferred are counted. Another point needed to be clarified is that, after the program made a mistake, all the following packets inference are swapped. That means if the program makes a mistake on packet n, and the following inferring was based on the decision of packet n, even if the program returns correct inference after packet n, the packet inferred as "PMU_A" before packet n are inferred as "PMU_B" after packet n. To check if there is a "swap" mistake, the comparison program checks the number of continues mistakes. If the number of continues mistakes reaches to 5, which is really rare, the program reduces the count of mistake by 5 and adds one swap mistake.

3.4 Results

A Pseudo Random Binary Signal (PRBS) of small magnitude is added to the excitation system of generators G15 and G16 of the IEEE 68 bus system [89]. Typically, 5-10% of the excitation voltage is applied as a PRBS signal. The PRBS signals are injected to the system to excite all possible dynamics of the power system. The results obtained in [126] using side-channel of inter-packet delays with window sizes 11, 9 and 7 are shown in Tables 3.4, 3.5 and 3.7 respectively.

The advantage of packet size side-channel has an 100% accuracy rate and a low computation complexity. To determine PMU model numbers, it requires a priori map from encrypted or unencrypted packet size to PMU models. To differentiate PMU models, no priori map is needed.

Table 3.4: Accuracy Rate Using Inter-Packet Timing Delay Side-Channel with Window Size of 11

Experiment setting	Correct Count	Mismatch Count	Swap Count	Accuracy Rate
PRBS to Generator 15	98072	1195	733	98.072%
PRBS to Generator 16	98392	1392	216	98.392%
No PRBS	98486	1463	51	98.486%
No Data Measurement	98119	1551	330	98.119%

Table 3.5: Accuracy Rate Using Inter-Packet Timing Delay Side-Channel with Window Size of 9

Experiment setting	Correct Count	Mismatch Count	Swap Count	Accuracy Rate
PRBS to Generator 15	98972	504	524	98.972%
PRBS to Generator 16	99193	764	43	99.193%
No PRBS	98604	1351	45	98.604%
No Data Measurement	98989	561	450	98.989%

However, the method is limited to differentiating PMU models and it cannot differentiate different PMU instances. For example this method can not differentiate packet sources under the following three instances: 1, Packets sent by multiple PMUs of the same model. 2, Packets sent by other sources that have the same size with packets sent from PMU. 3, Packets sent by PMUs that have the same least common multiple packet size, e.g. the sizes of packets from PMU_a are 100, 200 and 300 while the sizes of packets from PMU_b are 50, 100, 150 and 200.

The packet size side channel differentiates between PMU models, but can not separate different PMU data streams. The timing side channel can isolate specific PMU data streams, but can not identify the PMU model. The computation complexity of using inter-packet timing delay side-channel increases exponentially with respect to the length of history to look at, and more history is needed when more PMUs are in the network.

3.4.1 Histograms of PMU Data Streams

In this section, histograms are created using packet size, where the packets are collected between two security gateways under different power grid situations. Histograms with no PRBS in power grid are shown. There is no visible difference except packet numbers could be slightly different between histograms presented in this section and histograms created under other power grid situations.

Table 3.6: Accuracy Rate Using Inter-Packet Timing Delay Side-Channel with Window Size of 7

Experiment setting	Correct Count	Mismatch Count	Swap Count	Accuracy Rate
PRBS to Generator 15	98053	1090	857	98.053%
PRBS to Generator 16	99064	873	63	99.064%
No PRBS	99678	262	60	99.678%
No Data Measurement	99329	402	269	99.329%

Table 3.7: Accuracy Rate Using Packet Size Side-Channel

Experiment setting	Accuracy Rate
PRBS to Generator 15	100%
PRBS to Generator 16	100%
No PRBS	100%
No Data Measurement	100%

3.5 Summary

In this study, two methods are presented to distinguish the packets generated by different PMUs sent through an encrypted VPN tunnel using the packet size and inter-packet timing delay side-channels. A comparison between these two methods is also presented.

The results show that the proposed approach using packet size side-channel is 100% accurate under some certain preconditions. It is demonstrated that the methods can be applied to perform real-time analysis on the traffic of two PMUs. The comparison shows that the method using packet size side-channel require less computing power to perform real-time analysis with multiple PMUs. But this method is limited to determine different PMU models rather than different PMU instances. By using inter-packet timing delay side-channel, multiple PMUs of the same model can be distinguished.

As discussed earlier, those side-channels can be exploited to redirect or drop a target network communication instance and remains accessibility to the remote host device within a VPN tunnel. It could be difficult to detect the attack since the network session is still available. A simple way to eliminate this vulnerability is by using software or hardware to make the packets size and timing delay uniform. Future research includes using both packet size side-channel and inter-packet timing delay side-channel cooperatively, PMU dropping, side-channel analysis with more PMUs and PMU side-channel analysis in a more complicated real-world implementation.

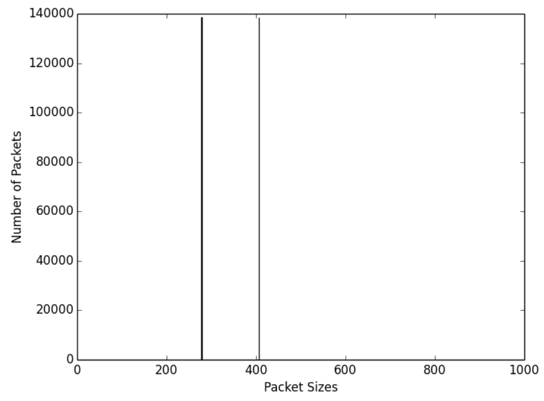


Figure 3.7: Histogram of one hour of data, which contains packets from PMU_0 and PMU_1 with no PRBS in power grid. The peak value at 278 presents packets from PMU_0. The peak value at 406 presents packets from PMU_1

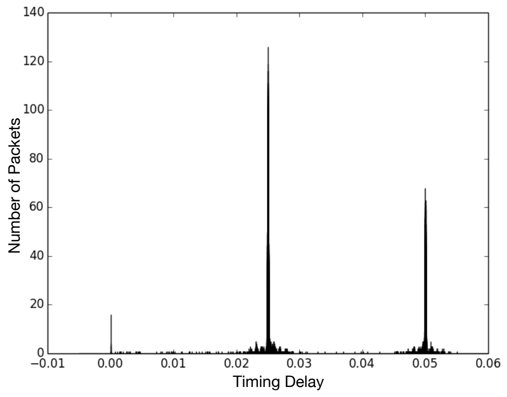


Figure 3.8: Histogram of one hour of data where packets are captured from PMU_0 with no PRBS in power grid.

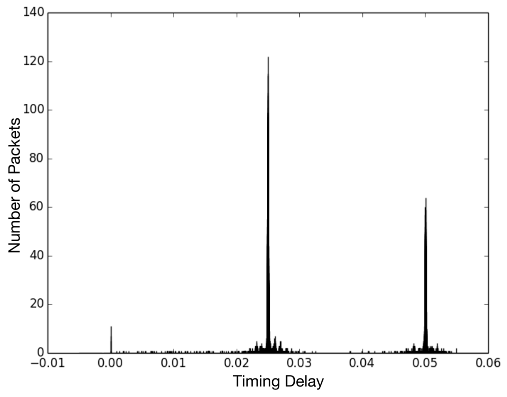


Figure 3.9: Histogram of one hour of data where packets are captured from PMU_1 with no PRBS in power grid.

Chapter 4

Traffic Camouflage

4.1 Overview

Malware authors conceal Command and Control (C&C) traffic[1] by masquerading as legitimate web browsing activity (HTTP or encrypted HTTPS) [44] or Internet relay chat (IRC). Twitter and DNS queries have also been used to hide C&C communication channels [94]. Detection techniques based on reverse engineering, side-channel analysis, etc., have emerged [16, 66]. Reverse engineering of malware develops network signatures for malicious traffic filtering [99]. Many bots use cryptography or obfuscation to disrupt signature-based detection [9], which can be foiled by blocking all unknown encrypted traffic [74]. However, this has two drawbacks: (1) discouraging the use of cryptography makes traffic less secure and (2) any false positives disrupt legitimate traffic producing denial of service. Other research uses traffic analysis for detection. Chen [65] presents timing side-channel analysis for Zbot detection using hidden Markov model (HMM) and probabilistic context-free grammar (PCFG). Passive DNS traffic analysis system for detecting and tracking malicious flux networks of a botnet is proposed in [84].

In this chapter, a *counter-countermeasure* that obscures the features used by malware traffic detection tools is presented. The idea is to camouflage malware traffic as an innocuous application protocol using format-transforming encryption (FTE) [35] and side-channel massage (SCM). FTE converts traffic flow to another application protocol using a regular expression describing the target

[1]In the context of cybersecurity, attackers use C&C infrastructure to issue C&C instructions to their victims through network.

protocol. This thwarts existing detection approaches because they detect that the infected machines are using IRC, HTTP/HTTPS or DNS to communicate. Additionally, the FTE output is processed using SCM to modify side-channel features, making the system resistant to side-channel analysis. The motivation of this study is to reveal the weakness of existing traffic detection techniques and show that more sophisticated malware traffic detection solutions are needed.

This counter-countermeasure is illustrated by camouflaging Zbot traffic. Zeus malware is notorious for stealing on-line financial information. The source code of Zeus is on Github [97]. In the experiment presented here, the Phasor Measurement Unit (PMU) communication protocol is the target protocol for the following reasons: (1) PMU traffic is in clear text [14, 104], and resistant to countermeasures that block encrypted traffic; (2) as early as 2009, the possibility of a power-grid botnet consisting of buggy smart meters was raised [42]; and (3) the synchrophasor protocol used for PMU communication is regular, simple and easy to imitate "in its entirety". The feasibility of converting botnet traffic to PMU traffic is explored in this study. Mimicking is done at the application layer of the protocol stack, which simplifies the imitation process and reduces the risk of errors. Experimental results show that the counterfeit PMU can talk with the real server without being detected. Since the same traffic can be processed to recover the C&C traffic, the botnet communications are effectively concealed.

4.2 Related Work

Malware uses traffic camouflaging to hide network traffic. Spyware Taidoor, a banking trojan discovered in 2008, used a Yahoo blog as a botnet C&C server [18], the malicious traffic was camouflaged as HTTP requests to blogs. According to Kaspersky Lab [102], ChewBacca and evolved Zeus use the Tor network to hide communications. Pushdo malware [85] included domain names of large companies or famous educational institutions in the C&C domain list, which made network traffic analysis more difficult. The Morto Trojan [94] sent false DNS requests to a DNS server, which was a C&C server, and the exchanged message was obfuscated by a simple Base64 encoding. The proposed method can transform arbitrary traffic to a target protocol, so that malware detection techniques looking for features of the original protocol won't work.

Besides hiding malware traffic, traffic camouflage can also be used to evade surveillance and censorship. The Onion Router (Tor) [90] provides an infrastructure for anonymous communication over a public network. The idea is to re-route network traffic through several browser-based short-lived proxies (called relay nodes) to evade surveillance and censorship [36]. However, as Tor becomes well-known, some unpublished relay nodes (bridges) are blocked in authoritarian countries [116].

To counter that, obfsproxy [86] is developed to circumvent censorship by camouflaging the Tor traffic between client and bridge. It supports multiple protocols, called pluggable transports, which specify how traffic is transformed. ScrambleSuit [117] is a polymorphic network protocol to obfuscate the transported application data to defend against active probing and protocol fingerprinting. SkypeMorph [75] is a Tor pluggable transport to reshape Tor packets to resemble Skype calls. StegoTorus [115] first uses chopping to change packet sizes and timing information, and then uses steganography to disguise Tor traffic as a message in an innocuous cover protocol, such as HTTP. FTE [35] evades regular-based deep packet inspection (DPI) technologies by transforming Tor traffic into a predefined format. Although FTE is effective in mimicking the contents of packets, it does not mimic timing information, which is vulnerable to side-channel analysis. The proposed SCM method 'massages' the timing side-channel to ameliorate FTE.

4.3 Background

4.3.1 Malware Traffic Detection

Network-based malware detection techniques examine the behavior of underlying malicious activities in the network traffic. Early study on network-based detection inspect the network packets for known botnet signatures. DPI technologies are usually used [1]. However, signature-based detection suffers from computational complexity and can be easily foiled by encryption and obfuscation.

Other detection approaches include anomaly [15] detection, and traffic analysis [65, 84]. Although anomaly detection might detect unknown malware, it usually has high false positive rates (FPRs). Traffic analysis exploits artifacts. Many properties of network traffic (packet size, power consumption, timing, electromagnetic radiation, etc.) remain observable after encryption and reveal valuable information. Attacks based on these properties are known as side-channel attacks. A timing side-channel attack is presented in [30], which breaks the anonymity of Tor network by matching packet timing data from any two Tor users to a model. The comparison gives a statistical likelihood that the two systems are actually communicating over Tor. In [126, 127], it is demonstrated that inter-packet delays and packet size can differentiate between packet streams coming from different

PMUs.

Chen et al. [65] show Zbot traffic can be identified via timing side-channel analysis. The timing pattern in network traffic is represented by a deterministic hidden Markov model (HMM), which can be constructed from the collected inter-packet delays using the zero-knowledge HMM inference algorithm [26, 95, 122]. Zbot traffic detection is performed by comparing the constructed HMMs of the target traffic and Zbot traffic. This method is leveraged in this study to build the HMM used for SCM.

4.3.2 Format-Transforming Encryption

A network protocol is a set of rules for communication. An implicit premise most of DPI is that regular expressions are sufficient for identifying protocols [35]. Tools, such as Snort [99], Bro [76] and L7-filter [37], use regular expressions to identify network protocols.

Format-Transforming Encryption (FTE) [35] is developed to evade regular expression based protocol identification. It extends conventional symmetric encryption by formatting the ciphertext. Arbitrary application-layer network traffic can be transformed into a target protocol using FTE. FTE is used to evade Internet censorship and surveillance, because the original protocols are obfuscated.

Although FTE is effective in hiding network protocols, it can be a challenge to write the regular expression for the target protocol. First, there is no automated software or procedures to extract regular expressions from protocols. Second, a regular expression that fully describes a protocol is usually complex. Although FTE can evade syntax-based protocol identification, it is vulnerable to side-channel analysis. Features like packet size, inter-packet delays etc., can be used for protocol identification. In this study, a SCM method is developed to obscure side-channel features and augment FTE.

The implementation of FTE includes LibFTE and FTEproxy. LibFTE [34] is a python library to transform string to ciphertext. FTEproxy [33] is a Tor [3] pluggable transport layer to resist keyword filtering, censorship and discriminatory routing policies. LibFTE (version 0.1.0) [34] is used in this study.

4.3.3 Zeus Botnet

Botnets are groups of compromised computers that botmasters (botherders) use to launch attacks over the Internet. As one of the most famous botnets, Zeus botnet is a malicious network that steals banking information with keystroke logging and screen capture [65]. It can also be used in other types of data or identity theft [71]. Since its first appearance in 2007, Zeus has been widespread and infecting Windows machines through drive-by downloads and phishing schemes. Sold on the black market, Zeus allows technically unsophisticated users to carry out cybercrimes. According to [40] Trend Micro, Zeus (2.0.8.9) source code can cost $400-500 in the Russian market. In 2011, the source code of Zeus was leaked on several underground forums over the Internet. A security researcher compiled the code and confirmed it worked 'like a charm' [88].

Zbot (version 2.0.8.9) was retrieved from Github [97] for experimentation. It contains code to build a Zeus Command and Control (C&C) server and generate executables to infect victim machines. A Zeus C&C server is a PHP server with mySQL and Apache.

4.4 Traffic Camouflage

The counter-countermeasure consists of 2 steps: (1) Zbot traffic is transformed using FTE to hide the source protocol; (2) the output of FTE is improved with SCM to modify inter- packet delays to evade side-channel analysis.

4.4.1 FTE

The input of FTE includes the message to encrypt and the regular expression describing the target protocol. The output ciphertext matches the input regular expression. There is a parameter, called $fixed\_slice$, which refers to the number of bytes that FTE receiver-side should decrypt of an incoming stream. For example, the target regular expression is '$\wedge(a|b) + \$$', which matches an arbitrary long string only comprising a or b. Given a message *helloworld* and $fixed\_slice = 512$, a string of *aaaaaaaabababaab...bababbabaaababab* containing 512 characters is obtained.

The goal is to transform Zbot traffic into a valid PMU data stream. PMU data is collected to infer a regular expression for synchrophasor protocol. Table 4.1 shows the structure of a PMU data packet [52], and the number of observed values for all fields. The extracted regular expression that fully describes observed PMU traffic is very complex. The transformation using this regular expression requires excessive computational resources.

Table 4.1: PMU packet fields in application layer

Byte Poisition	Length in Bytes	Parameter Name	Number of observed values
0 - 1	2	Synchroni-zation word	1
2 - 3	2	Framesize	1
4 - 5	2	PMU/DC ID number	1
6 - 9	4	SOC time stamp (UTC)	3958
10	1	Time quality flag	4
11 - 13	3	Fraction of second (raw)	30
14 - 15	2	Flags	4
16 - 271	8 each	Phasor #1 - #32	118713 each
272 - 275	4	Actual frequency value	52
276 - 279	4	Rate of change of frequency	7573
280 - 281	2	Digital status words	1
282 - 283	2	PMU checksum	54887

In this study, the FTE performance if improved by mapping observed PMU traffic directly to a regular expression. The Zbot traffic is first transformed with FTE using an example regular expression '$\wedge[0-9a-f]+\$$'. The output string consists of a sequence of hexadecimal symbols. There are 16 possibilities for each character position. Each character position corresponds to one PMU packet field. For each PMU packet field (Figure 4.1), the observed values of that field are divided into 16 groups and assign each group a unique symbol. Based on the predefined mapping relation from hexadecimal symbols to groups of value, for each character in the output string of FTE, a value is randomly picked from the related group.

Note that the previous procedure only applies to fields with at least 16 possible values. For fields with less than 16 possible values, a random value from the observation is used. For example, suppose the FTE output is 'a10b5d7...'. The first character 'a' corresponds to group number 10. So it can be replaced by a random value in the 10th group of the first field of PMU packet, which is 'SOC time stamp (UTC)'.

There are several advantages of the proposed mapping technique: (1) The input regular expression is straightforward. There is no need to extract regular expressions from target protocols. (2) Since regular expression extraction is not needed, the process can be easily automated. (3) The proposed mapping uses observed data, so the output is closer to real PMU traffic than traditional FTE. (4) The throughput can be increased by adjusting the number of symbols used to represent one protocol field.

4.4.2 Timing Side-Channel Massage (SCM)

Timing SCM modifies inter-packet delays to match the target protocol to evade side-channel analysis. Given collected inter-packet delays of the target protocol, a deterministic HMM representing the timing pattern is constructed. A HMM is a statistical Markov model in which the modeled system is a Markov process with unobserved (hidden) states. The standard HMM in [87] has two sets of random processes: states and outputs (observations). In a deterministic HMM, there is only a single set of random processes. In this model, the state transition labels are uniquely associated with an output alphabet.

To infer the HMM representing the timing pattern of PMU traffic, packet collection is performed on the PMU side. The time difference, $\triangle t$, is calculated by subtracting the receive time of the previous packet from the time of the current packet. In other words, $\triangle t_i = t_i - t_{i-1}$ where t_i is the receive time of packet i. Obviously, the first packet in the data sequence will not have a value, so the process start with $i = 2$. This step is done so that network latencies between the source and the destination are filtered out. Using the calculated values of $\triangle t$, the data is symbolized by grouping it into ranges and assigning anything in that range a unique symbol such as a or b. The symbolization result of is shown as Figure 4.1. Given the symbolized data sequence, the zero knowledge HMM inference algorithm introduced in [26, 95, 122] is used to create the deterministic HMM. The inferred model is shown in Figure 4.2.

Given the model, the counterfeit PMU starts the timing SCM process by randomly selecting a start state in the model. To send the packet, a transition is taken from the current state and the corresponding delay is waited before sending a packet to the PDC. If there are more than one possible transitions out of current state, the transition is chosen randomly, weighed on the probability of each

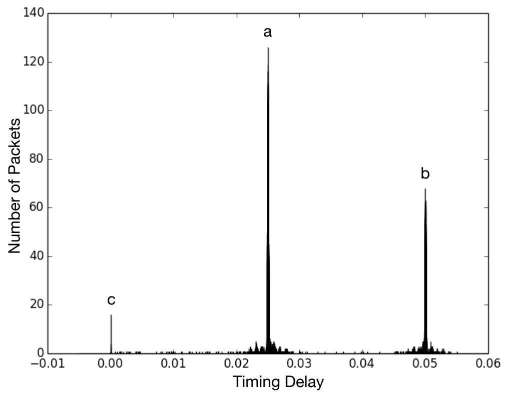


Figure 4.1: The symbolization of inter-packet timing delays of legitimate PMU traffic

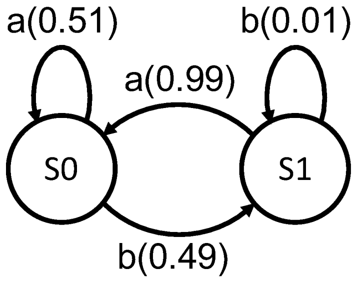


Figure 4.2: The HMM of inter-packet timing delay side-channel of legitimate PMU traffic

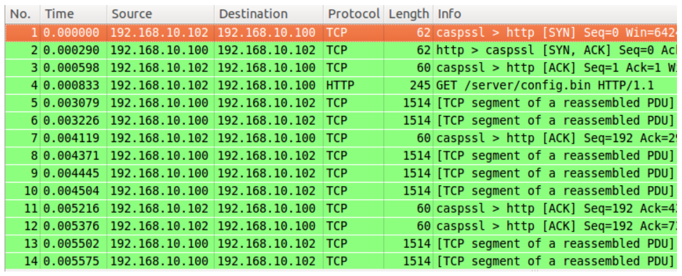
transition.

4.5 Experiment

4.5.1 Traffic Collection

A Zeus C&C server and a Zeus bot are set up in the lab on two Windows XP virtual machines. Figure 4.3 shows example botnet traffic. There are two protocols used: HTTP is for communication with the C&C PHP server, and TCP for information exchange between bot and botmaster. The volume of the collected bot traffic is 1.4MB.

In the Real-Time Power and Intelligent Systems (RTPIS) Laboratory of Clemson University [92], a real-time power system is simulated with one hardware PMU and software openPDC (as PDC). Network traffic between PMU and PDC are collected with Wireshark on the same machine



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.102	192.168.10.100	TCP	62	caspssl > http [SYN] Seq=0 Win=642
2	0.000290	192.168.10.100	192.168.10.102	TCP	62	http > caspssl [SYN, ACK] Seq=0 Ac
3	0.000598	192.168.10.102	192.168.10.100	TCP	60	caspssl > http [ACK] Seq=1 Ack=1 W
4	0.000833	192.168.10.102	192.168.10.100	HTTP	245	GET /server/config.bin HTTP/1.1
5	0.003079	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
6	0.003226	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
7	0.004119	192.168.10.102	192.168.10.100	TCP	60	caspssl > http [ACK] Seq=192 Ack=2
8	0.004371	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
9	0.004445	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
10	0.004504	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
11	0.005216	192.168.10.102	192.168.10.100	TCP	60	caspssl > http [ACK] Seq=192 Ack=4
12	0.005376	192.168.10.102	192.168.10.100	TCP	60	caspssl > http [ACK] Seq=192 Ack=7
13	0.005502	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]
14	0.005575	192.168.10.100	192.168.10.102	TCP	1514	[TCP segment of a reassembled PDU]

Figure 4.3: Screenshot of botnet traffic (part)

(windows 7 OS) running openPDC software. The traffic collection lasts around one hour, and the volume of collected traffic is 108.9MB.

4.5.2 Experiment Setup

The experiment setup is in Figure 4.4. The implementation includes a client-side program (counterfeit PMU program) deployed on the bot and a decoder integrated as a front end of C&C server. The client-side program takes the Zbot data stream as input, processes it with FTE then SCM. It then sends the false PMU data over the network. When the data arrives at the server side, it is decoded by the FTE decoder and forwarded the to C&C server.

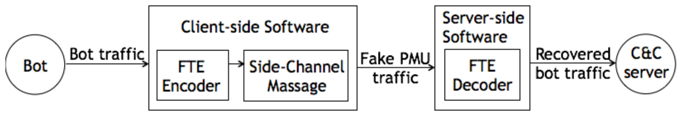


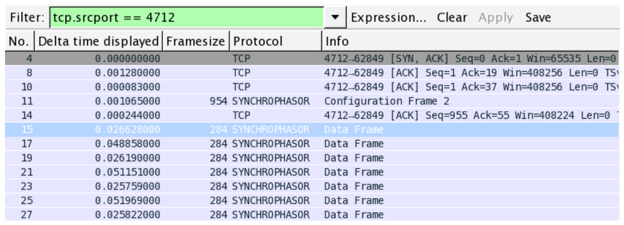
Figure 4.4: Experiment setup.

4.5.3 Experiment, Results and Analysis

It is found that the bot and its C&C server communicate fluently with the deployment of the traffic camouflage system. This indicates the false PMU data is decoded correctly before forwarded to the C&C server.

As shown in Figure 4.5a, the false PMU packets are recognized as the synchrophasor protocol, i.e., the protocol used for PMU-PDC communication. The packet details of false PMU displayed in Wireshark are shown in Figure 4.6a, which contain all the fields of a standard PMU packet as shown in Table 4.1. Values of each field are legitimate because they are consistent with the real

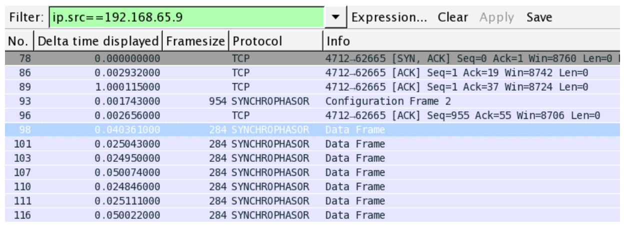
PMU packets. In particular, the voltage and current measurements are all within a reasonable range compared with real PMU packet shown in Figure4.6b.



Filter: tcp.srcport == 4712 Expression... Clear Apply Save

No.	Delta time displayed	Framesize	Protocol	Info
4	0.000000000		TCP	4712→62849 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
8	0.001280000		TCP	4712→62849 [ACK] Seq=1 Ack=19 Win=408256 Len=0 TS
10	0.000083000		TCP	4712→62849 [ACK] Seq=1 Ack=37 Win=408256 Len=0 TS
11	0.001065000	954	SYNCHROPHASOR	Configuration Frame 2
14	0.000244000		TCP	4712→62849 [ACK] Seq=955 Ack=55 Win=408224 Len=0
15	0.026628000	284	SYNCHROPHASOR	Data Frame
17	0.048858000	284	SYNCHROPHASOR	Data Frame
19	0.026190000	284	SYNCHROPHASOR	Data Frame
21	0.051151000	284	SYNCHROPHASOR	Data Frame
23	0.025759000	284	SYNCHROPHASOR	Data Frame
25	0.051969000	284	SYNCHROPHASOR	Data Frame
27	0.025822000	284	SYNCHROPHASOR	Data Frame

(a) Screen-shot of false PMU traffic (part)



Filter: ip.src==192.168.65.9 Expression... Clear Apply Save

No.	Delta time displayed	Framesize	Protocol	Info
78	0.000000000		TCP	4712→62665 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0
86	0.002932000		TCP	4712→62665 [ACK] Seq=1 Ack=19 Win=8742 Len=0
89	1.000115000		TCP	4712→62665 [ACK] Seq=1 Ack=37 Win=8724 Len=0
93	0.001743000	954	SYNCHROPHASOR	Configuration Frame 2
96	0.002656000		TCP	4712→62665 [ACK] Seq=955 Ack=55 Win=8706 Len=0
98	0.040361000	284	SYNCHROPHASOR	Data Frame
101	0.025043000	284	SYNCHROPHASOR	Data Frame
103	0.024950000	284	SYNCHROPHASOR	Data Frame
107	0.050074000	284	SYNCHROPHASOR	Data Frame
110	0.024846000	284	SYNCHROPHASOR	Data Frame
111	0.025111000	284	SYNCHROPHASOR	Data Frame
116	0.050022000	284	SYNCHROPHASOR	Data Frame

(b) Screen-shot of real PMU to PDC traffic (part)

Figure 4.5: Comparison between false PMU traffic and real PMU to PDC traffic

To further evaluate the proposed approach, side-channel analysis in [66] is launched against the false PMU data flow. The confidence interval (CI) approach in [22] is used to determine if the observed false PMU traffic matches the PMU HMM (see Figure 3.6). The inter-packet delays of the false PMU packets are computed and symbolized. Given the symbolized string, the CI method traces the data back through the HMM. Meanwhile, transition probabilities and corresponding confidence intervals are estimated by frequency counting. This process maps the observation data into the HMM structure. After the confidence intervals are estimated, the percent of transition probabilities from originally given HMM that fall into their respective confidence interval are determined. If this percentage is greater than a threshold value, it is accepted that the observations adequately match the HMM, i.e., the collected PMU traffic is legitimate in terms of timing.

Receiver operating characteristic (ROC) curves are usually used to find the optimal threshold value. By varying the threshold from 0% to 100%, the criteria for acceptance is progressively increased. Since the HMM (see Figure 3.6) in this experiment only contains 2 states and each state has one probability distribution, there are only 3 possible values for the percentage of probability distributions that match, which are 0, 50%, 100%. So there is no need for a ROC curve. The detection results using these three values are given in Table 4.2, where TPR denoted true positive rate, FPR denotes the false positive rate, and l is the threshold used for detection. TPR means the percentage of real PMU traffic that is correctly identified by Wireshark, while FPR means the percentage of false PMU traffic that is identified as PMU traffic. As shown in the results, it is impossible to obtain a high TPR and low FPR at the same time no matter how l varies. Therefore, it is difficult to perform an effective side-channel analysis.

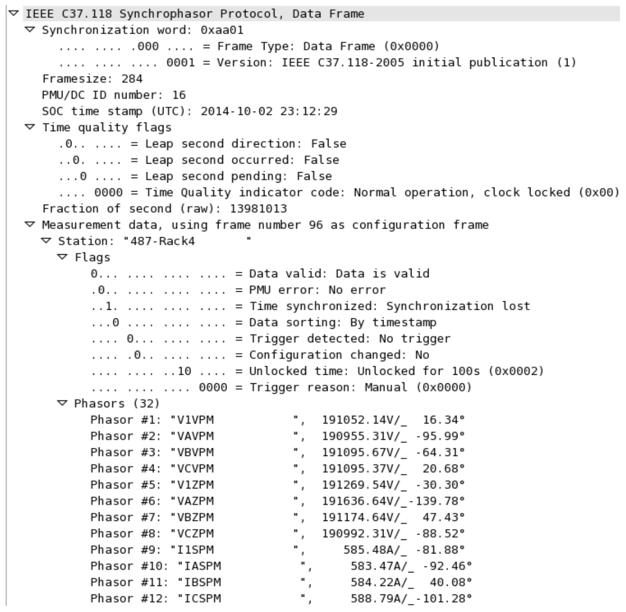
Table 4.2: TPR-FPR with different thresholds

Threshold l	$l = 0$	$0 < l \leq 0.5$	$0.5 < l \leq 1$
TPR	1	1	0.0193
FPR	1	0.9611	0.0352

A communication channel is built between the counterfeit PMU and software OpenPDC. All false PMU traffic is accepted by OpenPDC. This shows that the proposed method is capable of producing false PMU data stream close to real PMU.

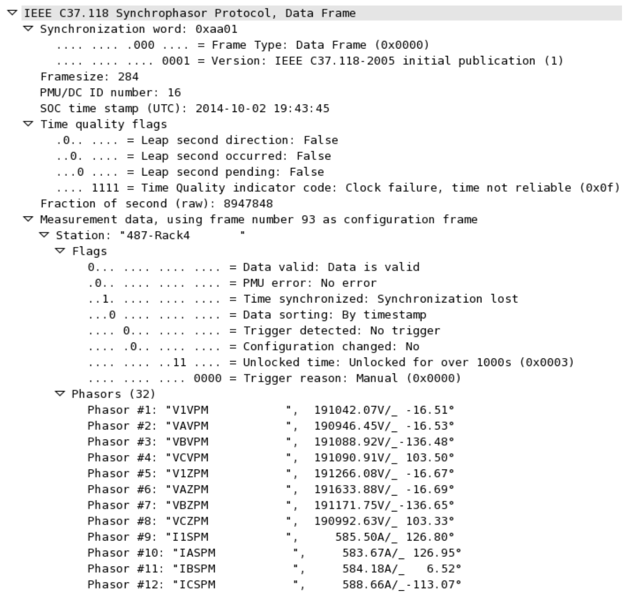
4.6 Summary

In this chapter, one protocol is transformed into another to disguise the original traffic and deceive existing network traffic detection approaches, including side-channel analysis. As an illustration, Zbot C&C traffic is transformed to PMU data stream. The experiment shows that the syntax of the counterfeit PMU packet is consistent with real PMU traffic. The proposed method is also resistant to timing side-channel analysis. False PMU packets are accepted by OpenPDC without any errors. Other target protocols can easily be used.



```
▽ IEEE C37.118 Synchrophasor Protocol, Data Frame
  ▽ Synchronization word: 0xaa01
      .... .... .000 .... = Frame Type: Data Frame (0x0000)
      .... .... .... 0001 = Version: IEEE C37.118-2005 initial publication (1)
    Framesize: 284
    PMU/DC ID number: 16
    SOC time stamp (UTC): 2014-10-02 23:12:29
  ▽ Time quality flags
      .0.. .... = Leap second direction: False
      ..0. .... = Leap second occurred: False
      ...0 .... = Leap second pending: False
      .... 0000 = Time Quality indicator code: Normal operation, clock locked (0x00)
    Fraction of second (raw): 13981013
  ▽ Measurement data, using frame number 96 as configuration frame
    ▽ Station: "487-Rack4       "
      ▽ Flags
          0... .... .... .... = Data valid: Data is valid
          .0.. .... .... .... = PMU error: No error
          ..1. .... .... .... = Time synchronized: Synchronization lost
          ...0 .... .... .... = Data sorting: By timestamp
          .... 0... .... .... = Trigger detected: No trigger
          .... .0.. .... .... = Configuration changed: No
          .... .... ..10 .... = Unlocked time: Unlocked for 100s (0x0002)
          .... .... .... 0000 = Trigger reason: Manual (0x0000)
      ▽ Phasors (32)
          Phasor #1: "V1VPM           ",  191052.14V/_  16.34°
          Phasor #2: "VAVPM           ",  190955.31V/_ -95.99°
          Phasor #3: "VBVPM           ",  191095.67V/_ -64.31°
          Phasor #4: "VCVPM           ",  191095.37V/_  20.68°
          Phasor #5: "V1ZPM           ",  191269.54V/_ -30.30°
          Phasor #6: "VAZPM           ",  191636.64V/_-139.78°
          Phasor #7: "VBZPM           ",  191174.64V/_  47.43°
          Phasor #8: "VCZPM           ",  190992.31V/_ -88.52°
          Phasor #9: "I1SPM           ",     585.48A/_ -81.88°
          Phasor #10: "IASPM           ",     583.47A/_ -92.46°
          Phasor #11: "IBSPM           ",     584.22A/_  40.08°
          Phasor #12: "ICSPM           ",     588.79A/_-101.28°
```

(a) Screen-shot of a typical camouflaged Zbot packet (part)



```
▽ IEEE C37.118 Synchrophasor Protocol, Data Frame
  ▽ Synchronization word: 0xaa01
      .... .... .000 .... = Frame Type: Data Frame (0x0000)
      .... .... .... 0001 = Version: IEEE C37.118-2005 initial publication (1)
    Framesize: 284
    PMU/DC ID number: 16
    SOC time stamp (UTC): 2014-10-02 19:43:45
  ▽ Time quality flags
      .0.. .... = Leap second direction: False
      ..0. .... = Leap second occurred: False
      ...0 .... = Leap second pending: False
      .... 1111 = Time Quality indicator code: Clock failure, time not reliable (0x0f)
    Fraction of second (raw): 8947848
  ▽ Measurement data, using frame number 93 as configuration frame
    ▽ Station: "487-Rack4       "
      ▽ Flags
          0... .... .... .... = Data valid: Data is valid
          .0.. .... .... .... = PMU error: No error
          ..1. .... .... .... = Time synchronized: Synchronization lost
          ...0 .... .... .... = Data sorting: By timestamp
          .... 0... .... .... = Trigger detected: No trigger
          .... .0.. .... .... = Configuration changed: No
          .... .... ..11 .... = Unlocked time: Unlocked for over 1000s (0x0003)
          .... .... .... 0000 = Trigger reason: Manual (0x0000)
      ▽ Phasors (32)
          Phasor #1: "V1VPM           ",  191042.07V/_ -16.51°
          Phasor #2: "VAVPM           ",  190946.45V/_ -16.53°
          Phasor #3: "VBVPM           ",  191088.92V/_-136.48°
          Phasor #4: "VCVPM           ",  191090.91V/_ 103.50°
          Phasor #5: "V1ZPM           ",  191266.08V/_ -16.67°
          Phasor #6: "VAZPM           ",  191633.88V/_ -16.69°
          Phasor #7: "VBZPM           ",  191171.75V/_-136.65°
          Phasor #8: "VCZPM           ",  190992.63V/_ 103.33°
          Phasor #9: "I1SPM           ",     585.50A/_ 126.80°
          Phasor #10: "IASPM           ",     583.67A/_ 126.95°
          Phasor #11: "IBSPM           ",     584.18A/_   6.52°
          Phasor #12: "ICSPM           ",     588.66A/_-113.07°
```

(b) Screen-shot of a typical PMU measurement packet (part)

Figure 4.6: Comparison between camouflaged Zbot packet and PMU measurement packet

Chapter 5

Denial of Service Attack on Phasor Measurement Unit

5.1 Overview

The increasing penetration of renewable energy sources such as Photovoltaics (PV) into the electric grid has advantages as well as disadvantages, specifically solar irradiance variability and uncertainty [67], [54]. The integration of renewable energy generation into the grid requires monitoring and the use of advanced control techniques [70], [60]. The deployment of PMUs, however, can improve grid management to mitigate the challenges of integrating renewable energy sources.

Concerns about the dependability and security of the communications infrastructure are affecting the adoption of synchrophasor technology, making it all the more essential to secure the network connections between PMUs and PDCs to ensure the reliability of smart grid operations [4], [6]. Side-channel analysis extracts information by observing implementation artifacts. In previous studies, where multiple PMUs transmit measurement packets through a VPN tunnel, packet size and inter-packet timing side-channels are used to differentiate packet sources [127]. This technique can be exploited to identify and selectively drop PMU traffic. When real-time PMU data is used for control, these attacks can cause the power system to become unstable.

In this chapter, the impact of a DoS attack based on tie-line bias control in a two area system are presented, followed by a description of the experiments performed in a simulated two-area four machine power system with a utility-scale PV plant. To illustrate the possible impact, a three phase-to-ground fault occurs in the power system while one PMU is blocked. The attack is performed based on side-channel analysis of PMU traffic in a VPN tunnel [127].

5.2 Two Area Four Machine Power System with Utility-Scale PV Plant and PMUs

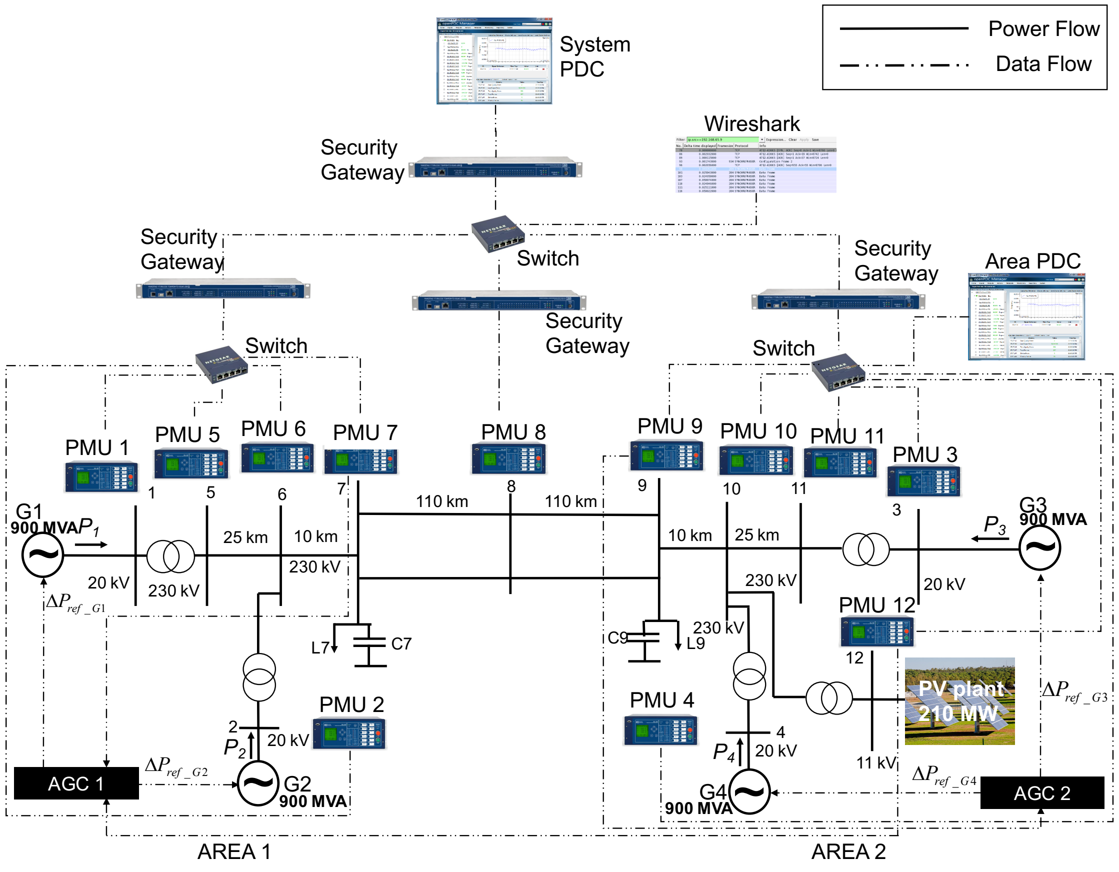


Figure 5.1: Two-area four machine power system with a 210 MW PV plant.

Fig. 5.1 shows the two-area four machine power system used in this study with each area having two synchronous generators, and with each rated at 900 MVA. A 210 MW utility-scale PV plant is integrated into Area 2. With the variable PV power generation in Area 2, there is a dynamic tie-line power flow from Area 1 to Area 2. Generator rated values and power outputs used in this study are shown in Table 1. Each area has an AGC implemented. The AGC in Area 1 (AGC 1) a

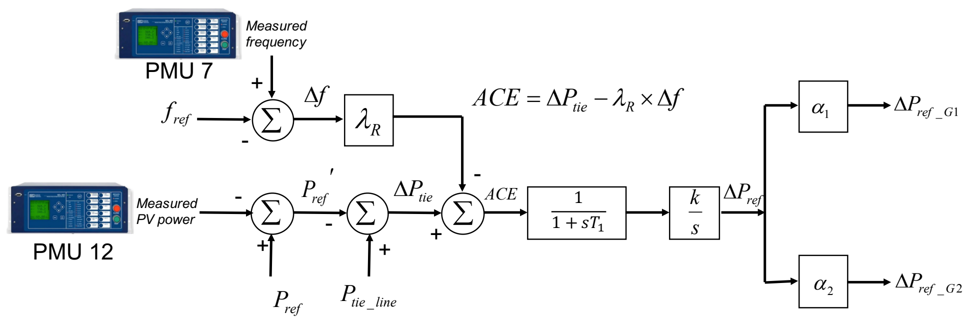


Figure 5.2: The Area 1 AGC (AGC1) diagram for tie-line bias control using PMU data.

diagram of which is detailed in Fig. 6.2 is implemented with tie-line bias control. PMU measured PV power output and Area 1 frequency values are passed as inputs to AGC 1, which decides the Area 1 generation by minimizing area control error. Tie-line power flow is regulated to optimally utilize the PV power generation in Area 1. Table 2 shows the parameters of AGCs used in this study. PMUs are placed at each power system Bus.

A dedicated communication synchrophasor network is used to transmit PMU measurement data, the configuration of which is shown in Fig. 5.1. There are four secured subnets connected by a dedicated network with a security gateway protecting each secured subnet. The attack described in this study has been found to be robust to issues related to network traffic interactions and topologies [26], [122]. For ease of presentation, simplified network is used for this study. Virtual Private Network (VPN) tunnels are established between each of the security gateways with traffic transmitted through VPN tunnels is encrypted by the security gateway, including the source and destination addresses. A system PDC is located at the control center subnet collecting data from PMUs located at Area 1 and the tie line subnet and an area PDC located at Area 2 subnet. The PMUs located at Area 2 are sending measurement data to the Area PDC. Synchronized measurement data from multiple PMUs are aligned by the system PDC and used for AGC operation.

The above model is implemented on the Real-Time Digital Simulator (RTDS) [91] facility at the Real-Time Power and Intelligent Systems (RTPIS) Lab [92] at Clemson University. The power system is simulated on RTDS, and data used for the power system control is collected using hardware PMUs.

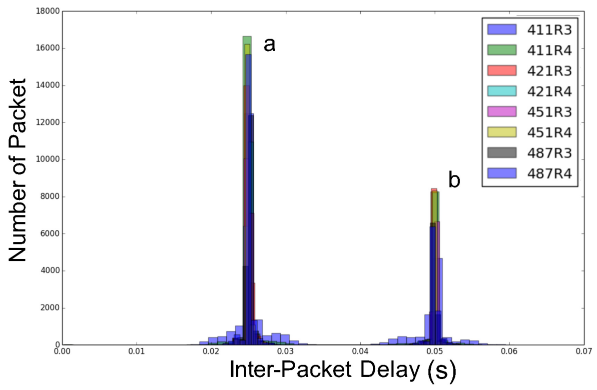


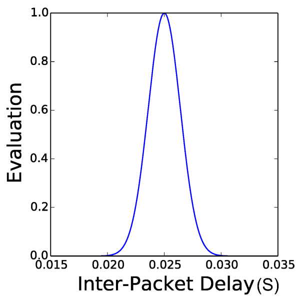
Figure 5.3: The symbolization of inter-packet timing delays of legitimate PMU traffic.

5.2.1 PMU Traffic Separation Algorithm

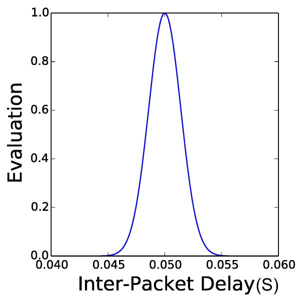
By using packet size and timing side-channels, PMU measurement traffic streams can be identified and differentiated from within a VPN tunnel. In a dedicated VPN tunnel, one or more PMU measurement traffic streams can be transferred. The timing side-channel and packet size side-channel for each PMU stream stays consistent. By using packet size side-channels, PMU streams that are not of interest can be ignored. Based on the histogram of PMU traffic, shown as Fig. 5.3, a normalized evaluation map of observed delta time for each symbol can be calculated, shown as Fig. 5.4. For each observed packet, delta times from previously received packets can also be calculated, and evaluations can be made based on the evaluation map. Unless more than one PMU is sending packets almost simultaneously, it is relatively easy to differentiate between different PMU streams. A recursive greedy algorithm is used to separate the traffic [127]. The HMM described in Section 3.3.2 is used to check the separated stream. A flowchart of the packet separation algorithm is given in Fig. 5.5.

5.2.2 DoS Attack on PMU Measurement Traffic

The system PDC collects measurements from PMUs and Area 2 PDC and send aligned data to the control center for AGC. The time stamp information in each measurement packet is then used for data alignment. In OpenPDC, an open source PDC software [7], a parameter called 'Lag Time' is used to determine the amount of time to required for the arrival of all the data for a particular time frame to arrive. If measurement packets are expected but not received within the



(a) For label ‘a’



(b) For label ‘b’

Figure 5.4: Normalized evaluation map of observed inter-packet delta time.

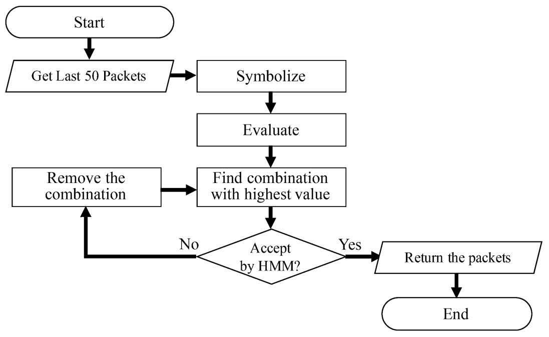


Figure 5.5: Flowchart of the separation algorithm.

time window, OpenPDC will send only the received data to the system PDC. Missing data positions are filled in order to keep the format of the aligned data packet constant. A flag in the aligned data packet indicates the data is invalid [52]. The filler is often a string of zeros, which means without any countermeasures, using value zero for AGC is harmful to the operation of the power system.

The attacker’s goal then becomes that of subverting the power grid by stealthily halting the PMU that monitors Bus 7 (Fig. 5.1) from sending measurement data to the control center. Either simply cutting the wire or blocking all encrypted traffic can be quickly diagnosed. Side-channel analysis and selectively dropping measurement packets from the target PMU is a more effective attack. Assume one node between the security gateways has been subverted by an attacker, as shown in the Wireshark screen in Fig. 5.1. The attacker performs a Man-In-The-Middle attack, then takes over the connections to the system PDC and carries out a side-channel analysis attack, during which the attacker drops all measurement data from one PMU without interrupting the connection [127]. When the PMU measurement packets are not delivered within an expected window, the PDC will send zeros to the control center for AGC. The lack of countermeasures means that AGC 1 keeps using zero as input that quickly leads to the power system instability.

The attack approach has been tested in environments with large amounts of extraneous traffic and shown to be resilient and thus not focused in this study [26], [122]. In this study, a simplified dedicated network is considered, shown as Fig. 5.1 with PMU 7 selected as the target. Data measurement traffic from PMU 7 can be identified and blocked by the attacker in a three-step process. The IP pair of interested security gateways is first identified. The packets that do not agree with the packet size side-channel are then ignored, and followed by the use of timing side-channels to identify and block packets from one PMU.

5.2.2.1 In step one, the IP addresses of interested security gateways are identified

Wireshark software is used to interpret the protocol of each packet so that security gateways send encrypted packets frequently and stably. Wireshark recognizes IP pairs that are sending and receiving encrypted packets, and three IPs are identified as the security gateways sending PMU measurements. In this study, each PMU transmits 30 packets per second (30Hz PMU rate). For three PMUs, 90 packets are transmitted from the security gateway. According to network configuration shown in Fig. 5.1, Area 1 subnet has 5 PMUs transmitting 150 packets per second; the Area 2 subnet has 6 PMUs transmitting 180 aligned packets per second; and the tie-line subnet has a single one PMU transmitting 30 packets per second. By validate the number of packets transmitted per second as well as packet size, the IP of the security gateway securing the Area 1 subnet is identified.

5.2.2.2 In step two, the PMUs are identified by packet size side-Channels

PMU 6 and PMU 7 is the same model with the packet size the same as the default setting. PMU 1, 2 and 5 are all different PMU models with the packet sizes differing from PMU 6 and PMU 7. A packet size filter is used to ignore PMU 1, 2 and 5.

5.2.2.3 In step three, the PMU packets are selectively dropped

Using the concept described in Section 5.2.1, two PMU measurement streams are differentiated. However, the separation algorithm cannot identify which stream originates from PMU 6 and which originates from PMU 7. The attacker can randomly pick a PMU measurement stream and drop all packets from that stream. In the analysis, only the results from blocking PMU 7 are presented.

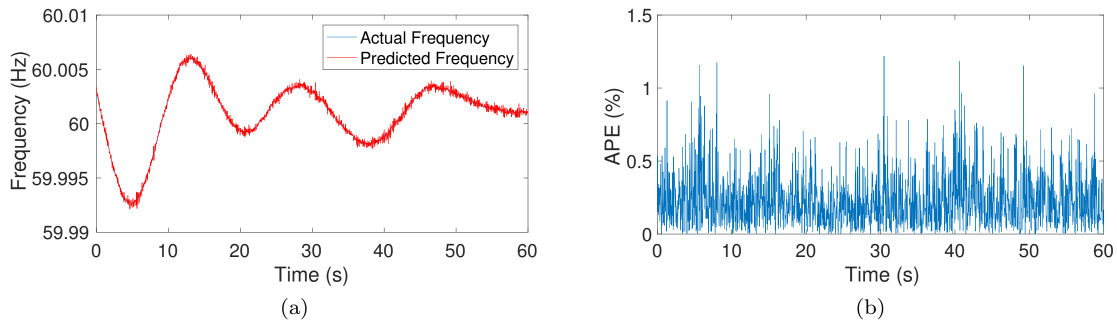


Figure 5.6: (a) Frequency predictions with PV integration. (b) Absolute Percentage Error (APE) of the PV predictions.

5.3 AGC Operation Under Attack

In this study, PMU 7 located at Bus 7 (Fig. 5.1), is used to provide frequency measurement to the AGC in Area 1, which is subjected to attack. Fig. 5.2 shows the AGC under attack.

5.3.1 Experimental Setup

All experiments and measurements are carried out in the RTPIS Lab in Clemson University. The PV plant is simulated using real-time weather data from Clemson, SC, and the hardware security gateways are configured to encrypt and decrypt packets between the hardware PMUs and OpenPDCs. Wireshark [28] (a network protocol analyzer software) is used to collect data packets, and the DoS attacks are performed between hardware security gateways.

In the RTPIS Lab, the real-time synchrophasor test bed is used for this study. Capable of integration with hardware equipment, RTDS is a simulation system that simulates power system operations in real-time. In this study, the real-time solar irradiance measured within the RTPIS Lab is used to calculate the PV power in real-time. All the test cases are carried out when the PV power generation is 88 MW and the corresponding generator power outputs are given in Table 1. The effect of solar irradiance variability and uncertainty in AGC is presented in [54]. In this study, variable solar irradiance is not considered for the benefit of comparison of test cases. Twelve Hardware PMUs are used to observe simulated measurement data from RTDS with each hardware PMU configured to monitoring the frequency, voltage and current of a Bus in the two-area four machine system, as shown as Fig. 5.1. Hardware PMUs transfer measurement data to OpenPDCs through the Clemson campus network, and the connections are encrypted with hardware security gateways, also shown in Fig. 5.1. The system PDC sends aligned measurement data back to RTDS through Ethernet connection, with the input from system PDC is configured for AGC control in the simulated power grid.

To observe the more obvious results, a fault is applied to the power system immediately after the DoS attacks are performed. A three phase-to-ground 'fault' at Bus 8 for ten cycles is implemented for each experiment. This simulates a trip of a tie-line.

The DoS attacks randomly pick either PMU 6 or PMU 7 to block. To simplify the result analysis, only the results are presented where PMU 7 is blocked. For Countermeasure A, the frequency used by the AGC depends on the last frequency received at the beginning of the attack started, which can be varied in each experiment. Two cases are analyzed for Countermeasure A, hereafter referred to as 'Case 1' and 'Case 2'. In Case 1, the last frequency received is 60.2Hz, and in Case 2, the last frequency received is 59.8Hz. The experimental results under different disturbances as well as the effectiveness of VSN are presented in this section.

5.4 Consequences of DoS Attacks

5.4.1 Fault and attack without countermeasure

The response of the two-area four machine power system after a three-phase to ground fault at Bus 8 for ten cycles with a DoS attack blocking one, five, and 100 measurement packets from PMU 7 are presented in Figs. 2, 3 and 4 respectively.

5.4.2 Analysis

The consequences and countermeasures are evaluated from the perspective of system stability and the energy change according to expected energy to be delivered or generated.

The difference of the energy either delivered or generated that expected in a one minute window after a fault under each scenario is calculated by:

$$\Delta E = \int_0^{\frac{1}{60}h} (P(t) - P_E)dt \tag{5.1}$$

Where $P(t)$ is the power measured at the tie-line or generators in each cases, and P_E is the expected power either delivered or generated. The results are shown in Table 3 and Fig. 1.

5.4.2.1 Consequences Without Any Countermeasures

The absence of one to five packets will affected the power generation and the tie-line flow. The energy delivered on the tie-line also increased dramatically as more packets are been blocked. The absence of five or more packets caused a forced oscillation on the power generation and tie-line flow. The absence of 100 or more packets results in system instability.

5.5 Summary

In this study, the consequences of DoS attacks that exploits a side-channel vulnerability in a synchrophasor network has been demonstrated. The lack of countermeasures greatly increases the expense of lost PMU measurement packets during system operation. Results indicated that the lack of additional countermeasures can render the synchrophasor networks vulnerable to a DoS attack, even if the network communications have been encrypted with security gateways.

Chapter 6

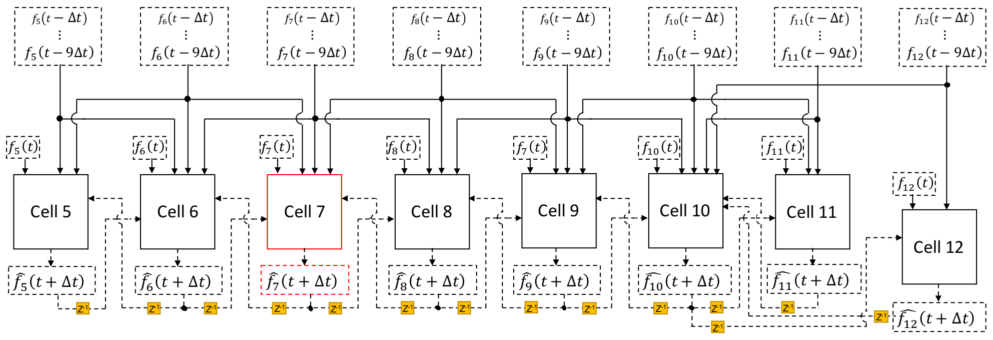
Virtual Synchrophasor Network

6.1 Overview

In this chapter, the mitigation of a DoS attack on tie-line bias control in a two area system are presented. This chapter details the creation and implementation of a Cellular Computational Network (CCN) [68] based Virtual Synchrophasor Network (VSN) for Automatic Generation Control (AGC), with the objective of mitigating the impact of PV power generation fluctuations and other disturbances on system frequencies. The topology of the VSN matches the topology of the power system, which means that during a DoS attack, missing PMU data can be inferred using the VSN. To illustrate the possible impact, a three phase-to-ground fault occurs in the power system while one PMU is blocked. The attack is performed based on side-channel analysis of PMU traffic in a VPN tunnel [127].

6.2 Cellular Computational Network

CCN is a scalable and distributed framework for predicting/estimating the dynamics of large networked systems [68], [69]. A CCN can be expressed as a directed graph. Fig. 6.1 shows an example CCN topology. Each of the nodes in a CCN is a computational cell, shown as the square boxes in Fig. 6.1. Each cell consists of a computational unit, a learning unit, and a communication unit. Each cell takes an input vector and calculates an output in one time step. Cells can learn to generate desired output using historical data. The output estimated by each cell can be passed to



$f_5(t-\Delta t)$ ⋮ $f_5(t-9\Delta t)$
$f_6(t-\Delta t)$ ⋮ $f_6(t-9\Delta t)$
$f_7(t-\Delta t)$ ⋮ $f_7(t-9\Delta t)$
$f_8(t-\Delta t)$ ⋮ $f_8(t-9\Delta t)$
$f_9(t-\Delta t)$ ⋮ $f_9(t-9\Delta t)$
$f_{10}(t-\Delta t)$ ⋮ $f_{10}(t-9\Delta t)$
$f_{11}(t-\Delta t)$ ⋮ $f_{11}(t-9\Delta t)$
$f_{12}(t-\Delta t)$ ⋮ $f_{12}(t-9\Delta t)$
$f_5(t)$
$f_6(t)$
$\hat{f_7}(t)$
$f_8(t)$
$f_7(t)$
$f_{10}(t)$
$f_{11}(t)$
$f_{12}(t)$
Cell 5
Cell 6
Cell 7
Cell 8
Cell 9
Cell 10
Cell 11
Cell 12
$\hat{f_5}(t+\Delta t)$
$\hat{f_6}(t+\Delta t)$
$\hat{f_7}(t+\Delta t)$
$\hat{f_8}(t+\Delta t)$
$\hat{f_9}(t+\Delta t)$
$\hat{f_{10}}(t+\Delta t)$
$\hat{f_{11}}(t+\Delta t)$
$\hat{f_{12}}(t+\Delta t)$
z^{-1}

Figure 6.1: CCN based VSN architecture for frequency prediction in 2-area 4-machine power system with PV plant.

other connected cells as input(s) for further computation. A cell can provide more accurate outputs from the information provided by interconnected cells, which can be deployed either on a single computer or multiple computers connected through a network. The type of the computational unit in a cell depends upon the application, with the goal of predicting/estimating the required output using the available information. A cell has the ability to learn autonomously, for example computational intelligence (CI) paradigms such as neural networks and fuzzy systems.

The advantage of CCN is that the topology of a CCN can be designed to match the topology of the objective power system, permitting the deployment of the cell to local systems. CCN is an appropriate scalable computational framework to learn and predict/estimate the behavior of a power system, the topology of which can be mapped to a CCN, where the Buses of the power system can be simulated by cells. Cells can be learned to follow the characteristics of each component/Bus, and for each cell, predictions can be made based on information gathered from adjacent connected cells. By carefully designing a CCN that matches the topology of the objective power system, it is possible to make predictions based on the dynamics of the power system. In [111], dynamic state estimation (DSE) of a power system can be made by the CCN even if multiple PMUs' data is missing.

When the information of a part of the objective power system is not available, the missing information can be estimated using CCN. Power systems must be closely monitored during unexpected situations such as security attacks; CCN can provide a better situational awareness/intelligence for power system operation and control.

In this study, a CCN based VSN architecture is implemented for the two-area four machine power system by mapping each Bus of the system (Fig. 5.1) to a cell. Cells are connected to their neighboring cells (mapping adjacent connected Buses), which communicate with each other. Fig.

6.1 shows the CCN based VSN proposed for this study with arrows indicating the input and output data flow directions of cells. PMUs are deployed at each Bus of the power system. Implemented cells use near-real-time and historical data from the PMUs (indicated by solid arrows) and interconnected cells' previous time step predicted values (indicated by dash arrows with yellow boxes) to predict the next state of the PMU data (indicated by dash arrows). Z^{-1} indicates a single time step delay that occurs with the acquisition of data from neighboring cells. Each cell in the CCN includes an extreme learning machine (ELM) as the computational unit. ELM is a type of single hidden layer feedforward network, where the input to hidden layer weights is assigned randomly. In an ELM, input to hidden layer weights are kept consistent during the learning process. Thus, ELM is extremely fast in learning input-output mapping [54]. ELM is more suitable for non-linear time series predictions. Specifically, if a PMU is attacked, the communication capability with adjacent connected Buses aids each cell to provide accurate predictions at which point the missing data values are estimated by a VSN and used for the AGC operation. ELM based CCN frequency prediction approach for two-area four-machine power system is presented in [53], which clearly explains the development and application of ELM including its training and testing methods used in this study.

6.3 AGC Operation Under Attack

In this study, PMU 7 located at Bus 7 (Fig. 5.1), is used to provide frequency measurement to the AGC in Area 1, which is subjected to attack. Fig. 6.3 shows the flowchart for the test cases. From each time stamp t, the PMU packet receiver waits Δt time till a packet arrives. At time $t+\Delta t$ a flag status is checked and different test cases are followed based on the status of the flag. If a packet has arrived before $t+\Delta t$, the flag is set to 0, and the received value is used for AGC control. To study the consequence of PMU packet dropping without any countermeasure, the flag is set to 0, and the frequency value 0 is sent to the AGC if a PMU packet is not received until $t+\Delta t$. To study the effect of countermeasures, if the packet is not received on time, the flag status is set to either 1 or 2, and a countermeasure is implemented to replace the missing data. There are two types of countermeasures, A and B. Countermeasure A uses the last received data to replace the missing data, and Countermeasure B uses the CCN based VSN estimated data to replace the missing data.

The two countermeasure approaches are implemented and deployed on a computer in which OpenPDC is installed at the control center. Fig. 6.2 shows the AGC switching behavior based on

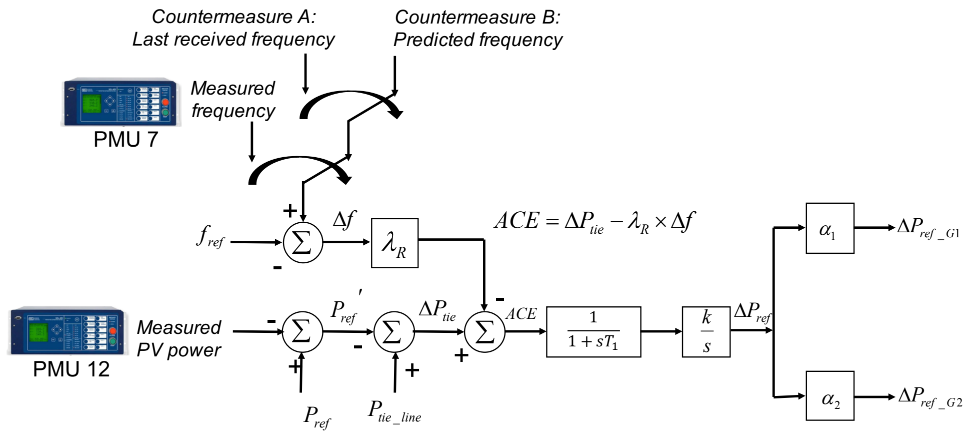


Figure 6.2: The Area 1 AGC (AGC1) diagram for tie-line bias control using PMU data, old data, or VSN data.

the frequency input value of the flag.

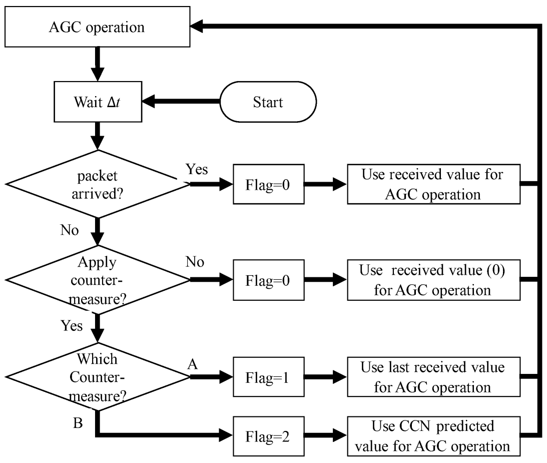

AGC operation
Wait Δt
Start
packet arrived?
Yes
No
Flag=0
Use received value for AGC operation
Apply counter-measure?
No
Yes
Flag=0
Use received value (0) for AGC operation
Which Counter-measure?
A
B
Flag=1
Use last received value for AGC operation
Flag=2
Use CCN predicted value for AGC operation

Figure 6.3: The flowchart for AGC operations under DoS attacks. This includes four possible cases studied: 1) packet arrived before $t + \Delta t$ and used for AGC; 2) packet did not arrive before $t + \Delta t$ and value '0' is used for AGC; 3) packet did not arrive before $t + \Delta t$ and last arrived value is used for AGC; 4) packet did not arrive before $t + \Delta t$ and VSN estimated value is used for AGC.

6.3.1 Countermeasure A

For Bus i, the last received data $f_i(t)$ at time t is sent to replace the missing data at time $t + \Delta t$. If the system keeps missing data over multiple continuous time intervals, then the data received at time t is sent to AGC continuously until the flag status changes.

6.3.2 Countermeasure B

For Bus i, VSN predicted frequency $\hat{f}_i(t+\Delta t)$ is used to replace the missing data at time $t+\Delta t$. Similar to Countermeasure A, if the system continues to missing data for multiple continuous time intervals, the VSN predicted frequency value $\hat{f}_i(t{+}2\Delta t)$ based on previously predicted frequency value $\hat{f}_i(t+\Delta t)$ is used for AGC.

6.3.2.1 Single time step predictions

The input vector for cell i is given by (6.1)

$$\begin{aligned} f_{input(i)}(t) = \{&f_i(t), f_i(t-\Delta t), ..., f_i(t-9\Delta t), ..., \\ &\hat{f_j(t)}, f_j(t-\Delta t), ..., f_j(t-9\Delta t)\} \end{aligned} \tag{6.1}$$

where $f_i(t)$ is measured i^{th} Bus frequency at time t, $\hat{f_j(t)}$ is the predicted j^{th} Bus frequency at time t, $f_j(t-\Delta t)$ is measured j^{th} Bus frequency at $t-\Delta t$, j represents all the neighboring Buses, and Δt is the prediction time step. In this study Δt is 33 ms.

The output of cell i is the predicted frequency $\hat{f}_i(t+\Delta t)$ at time $t+\Delta t$.

For example, the input for frequency prediction at cell 7 at time $t+\Delta t$ is using Bus 7 historical frequency values and neighboring Buses historical frequency values as given by (6.2).

$$\begin{aligned} f_{input(7)}(t) = \{&f_7(t-9\Delta t), ..., f_7(t), f_6(t-9\Delta t), ..., \\ &\hat{f_6(t)}, f_8(t-9\Delta t), ..., \hat{f_8(t)}\} \end{aligned} \tag{6.2}$$

The output of cell 7, which is the predicted frequency data $\hat{f}_7(t+\Delta t)$ at time $t+\Delta t$, is calculated according to (6.3)

$$\hat{f}_7(t+\Delta t) = sig(W_{in} \times f_{input(7)}(t) + bias) \times W_{out} \tag{6.3}$$

where W_{in} and W_{out} are the respective input and output weights of the ELM. The sigmoid function is used as the activation function.

Fig. 5.6 shows the single time step ahead (33ms) frequency prediction results obtained for the frequency at Bus 7 with PV power variability.

6.3.2.2 Recursive frequency predictions

If the system keeps missing data packets for multiple Δt time intervals, VSN recursively uses its predicted values in input function. Fig. 6.4 shows an example of recursive VSN prediction procedure for Bus 7 for two consecutive Δt time intervals. The data from PMU 7 at time $t+\Delta t$ and $t+2\Delta t$ are not delivered on time. Please note that at time $t+\Delta t$, data from neighboring PMUs are delivered and used for prediction at time $t+2\Delta t$.

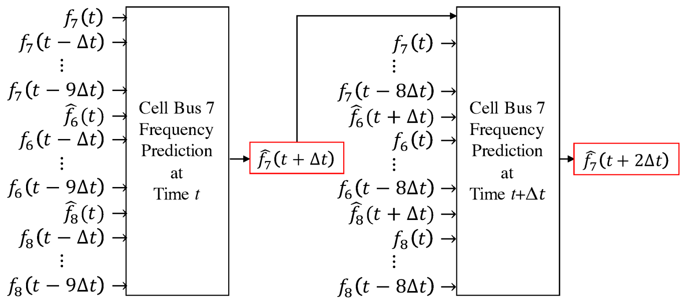


Figure 6.4: Recursive VSN prediction procedure for Bus 7 for two consecutive Δt time intervals.

6.4 Experiment and Results Analysis

6.4.1 Experimental Setup

The attacks presented in this chapter shares the same setup with the previous chapter5.3.1. The DoS attacks randomly pick either PMU 6 or PMU 7 to block. To simplify the result analysis, only the results are presented where PMU 7 is blocked. For Countermeasure A, the frequency used by the AGC depends on the last frequency received at the beginning of the attack started, which can be varied in each experiment. Two cases are analyzed for Countermeasure A, hereafter referred to as 'Case 1' and 'Case 2'. In Case 1, the last frequency received is 60.2Hz, and in Case 2, the last frequency received is 59.8Hz. The experimental results under different disturbances as well as the effectiveness of VSN are presented in this section.

6.4.1.1 Fault and attack with Countermeasure A deployed

The response of the two-area four machine power system, where Countermeasure A is deployed, after a three-phase to ground fault at Bus 8 for ten cycles with a DoS attack blocking 5 and 300 measurement packets from PMU 7 are presented in Figs. 5 and 6 respectively.

6.4.1.2 Fault and attack with Countermeasure B deployed

The response of the two-area four machine power system, where Countermeasure B is deployed, after a three-phase to ground fault at Bus 8 for ten cycles with a DoS attack blocking 5 and 300 measurement packets from PMU 7 are presented in Figs. 7 and 8 respectively.

6.4.2 Analysis

The countermeasures are evaluated from the perspective of system stability and the energy change according to expected energy to be delivered or generated.

The difference of the energy either delivered or generated that expected in a one minute window after a fault under each scenario is calculated by:

$$\Delta E = \int_{0}^{\frac{1}{60}h} (P(t) - P_E)dt \tag{6.4}$$

Where $P(t)$ is the power measured at the tie-line or generators in each cases, and P_E is the expected power either delivered or generated. The results are shown in Table 3 and Fig. 1.

6.4.2.1 Countermeasure A

The use of old frequency data to replace the missing measurement data stabilized the power system, and no forced oscillation in power generation and tie-line power flow is observed. However, the occurrence of a DoS attack at a very high system frequency of 60.2Hz deceived the AGC in Area 1 which reduced the power generation in Area 1. To mitigate this power loss, the power generation at Area 2 is increased. Similarly, the AGC in Area 1 is again deceived by a very low system frequency of 59.8Hz, which in this case increased the power generation in Area 1. The power delivered on tie-line is increased, and power generated in Area 2 is reduced. Although not experimentally observed, this increase in power either delivered or generated can trip the tie-line or synchronous generators.

6.4.2.2 Countermeasure B

The use of CCN predicted frequency data to replace the missing measurement data showed no obvious effects on power system operation, power generation and tie-line flow.

6.5 Summary

In this chapter, the CCN is used to develop a VSN that estimate the missing data during DoS attacks. The ability to accurately estimate the frequency change trends during the DoS attack makes it possible to deliver the results demonstrated in this chapter.

The VSN provides an innovative solution for adopting PMU measurement data for closed-loop control. The VSN is realized using a cellular computational network to predict frequency at the power system buses when real-time data is not available. The real-time implementation of DoS attacks and mitigation of damage using VSN validated the effectiveness of the proposed method for frequency prediction. A countermeasure that simply uses last received data is found ineffective in mitigating damage especially the change of energy delivered or generated during a severe event.

The original intention in deploying synchrophasor devices is for reducing the system operation cost while increasing both revenue and service quality. However, the cost of a guaranteed delivery of real-time PMU measurement data is expensive. According to the results presented in this study, the CCN based VSN prediction is validated as a practical solution that effectively reduces both the cost and risk of adopting PMU in closed-loop applications. Also, this proposed CCN based VSN can increase the resiliency of power system operations to DoS attacks, thus enhancing the reliability with variable generation systems (e.g. large solar farm integrated power system). This novel VSN technology in turn makes it possible to minimize the penalties from the North American Electric Reliability Corporation (NERC). Further study shall be performed to evaluate the reliability improvement by applying the proposed VSN to smart grid operations [93].

Chapter 7

VSN Resiliency

7.1 Overview

In the previous study, a Virtual Synchrophasor Network (VSN) based on a cellular computational network (CCN) that infers missing PMU measurement data during a DoS attacks is used. VSN is proven effectively mitigate damage during a DoS attack against a single PMU that concurrent with a three-phase touch ground fault. The system can stabilize even after 600 packets have dropped during an attack [131].

In this study, the resiliency of the VSN is further tested by mitigating DoS attacks on multiple PMUs at the same time. This study tested the VSN technique on both the previously studied two-area four machine power system and the IEEE 68 bus system. The computation unit in each cell is trained to further adapt to harsher situations such as a severe system fault during multiple DoS attacks.

7.2 Virtual Synchrophasor Network

The CCN based VSN shown in Fig. 5.1 with dash arrows indicating the input and output data flow directions of Cells. The topology of the VSN follows the topology of the two area four machines network with each Cell mapped to a PMU that provides frequency measurements. The power system takes frequency data from Bus 7 and Bus 9 as AGC 1 and AGC 2 input repeatedly. Thus, that two PMUs are considered as the primary layer PMUs. PMU 6, PMU 8 and PMU 10 are secondary layer PMUs, which are the direct neighbors to the primary layer PMUs. Finally, PMU 5, PMU 11 and PMU 12 are considered as tertiary layer PMUs.

The computational unit in each Cell is a one hidden-layer feedforward neural network, with nine historical measurement data from both the corresponding PMU and neighboring PMUs, one current measurement data from the corresponding PMU and one inferred frequency data from each of the neighboring Cells. The number of neurons in the hidden-layer is three times of the number of input neurons. The input layer and output layer of the neural are using linear activation whereas the hidden-layer uses sigmoid activation function. Under cases such as DoS attack s or network delays where the measurement data from PMUs are not immediately available, the input for the next iteration of inference will switch to the previously inferred frequency data.

There are two stages of learning to develop the proposed VSN. In the first stage, the VSN learns with data captured during operations without DoS attacks. In the second stage, some of the input streams of VSN is disabled to create simulated DoS attack scenarios. For both stages, the learning data is generated using the two area four machine system simulated by RTDS simulation system. The learning data is captured during system operations without DoS attacks but with several system faults. The faults are 10 cycle three phase to ground fault at Bus 8.

Very few learning epochs are performed in the first stage to avoid local minimization. In the second stage, several attack patterns are designed and listed in Table 7.1. In each epoch, an attack pattern is selected from the table. The PMUs streams under simulated attacks are disabled during that epoch. The VSN has to switch the input to previously inferred data for those data that are not available. At the beginning of each epoch, a random value is used as the input to avoid a deadlock, since no output is generated from any Cells yet.

The developed VSN is then integrated with the hardware PMUs and RTDS for simulation, where the VSN is taken input from the hardware PMUs and the output from VSN Cell 7 and Cell 9 are then sent to the AGCs in RTDS real-time simulation. The AGCs select either the hardware measurement data when it is available or the VSN inferred data when the hardware measurement data is not immediately available.

Table 7.1: Simulated DoS patterns for the second training stage.

PMU ID	5	6	7	8	9	10	11	12
Pattern 1 (P* layer)			x					
Pattern 2 (P layer)					x			
Pattern 3 (S layer)		x						
Pattern 4 (S layer)				x				
Pattern 5 (S layer)						x		
Pattern 6 (T layer)	x							
Pattern 7 (T layer)							x	
Pattern 8 (T layer)								x
Pattern 9 (PST layer)		x	x	x	x	x	x	x
Pattern 10 (PST layer)	x		x	x	x	x	x	x
Pattern 11 (PST layer)	x	x		x	x	x	x	x
Pattern 12 (PST layer)	x	x	x		x	x	x	x
Pattern 13 (PST layer)	x	x	x	x		x	x	x
Pattern 14 (PST layer)	x	x	x	x	x		x	x
Pattern 15 (PST layer)	x	x	x	x	x	x		x
Pattern 16 (PST layer)	x	x	x	x	x	x	x	
Pattern 17 (PT layer)	x		x		x		x	
Pattern 18 (ST layer)		x		x		x		x

*'P', 'S' and 'T' are the abbreviations of 'Primary', 'Secondary', and 'Tertiary' respectively. PMU Data feeds marked with x are disabled in the respective training pattern.

7.3 System Resiliency with VSN

7.3.1 VSN Resiliency on Two Area Four Machines System

Fig. 5.1 shows the experiment environment including the physical layer, the VSN layer, and the two AGCs. In the hardware layer, the two-area four machine power system is used in this study. Each area has two synchronous generators, and with each rated at 900 MVA. A 210 MW, utility-scale PV plant, is integrated into Area 2. With the variable PV power generation in Area 2, there is a dynamic tie-line power flow control from Area 1 to Area 2. An AGC is implemented in each area.

PV power output and Area 1 frequency values from PMU 7 are passed as inputs to AGC 1, which dispatches the Area 1 generation by minimizing area control error. Tie-line power flow is regulated to utilize the PV power generation in Area 1 optimally. The Area 2 frequency measured by PMU 9 is sent to AGC 2.

The DoS attack experiments exams the VSN by gradually increasing the number of layers of PMUs been attacked as well as the attack time in each experiment concurrent with a 10 cycle three phase to ground system fault at Bus 8. Fig. 7.1 shows the time frames of each stage of the attack. The frequency response, total energy change, and the cumulate absolute percentage error (CAPE) are the three factors used to evaluate the performance of the VSN under each attack.

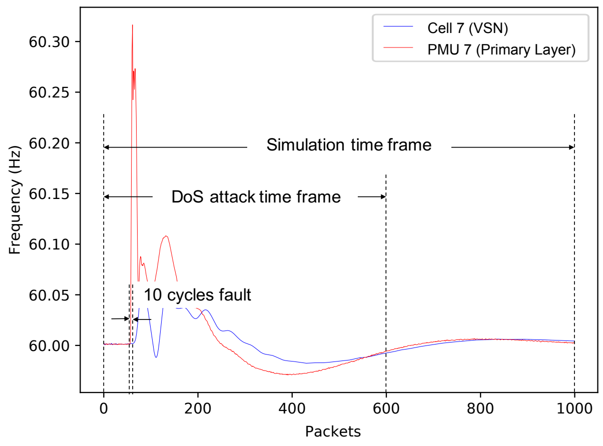


Figure 7.1: The illustration of time frames of the DoS attack experiments.

The attack experiments are performed in the following fashion: First, consider the number of PMUs been attacked. There are four groups of attacks, start from attacking at single PMU at the primary layer. Then attack the primary layer. Then attack the primary layer and secondary layer. Finally attacks against all three layers of PMUs except PMU 12. Then, consider the duration of each DoS attack. In each group of attacks, the duration of the DoS attacks performed is 2 seconds (60 packets drop), 5 seconds (150 packets drop), 10 seconds (300 packets drop) and 20 seconds (600 packets drop). There is a total of 12 experiments performed. In all experiments, the system fault is applied one second following the DoS attack.

7.3.2 VSN Resiliency on IEEE 68-Bus System

In addition to the test on the two area four machine system, a VSN for IEEE 68-bus system was trained and tested. Similar to the two area four machine system, Figure 7.2 shows the IEEE 68-Bus System with VSN that having the same topology. Similarly, the 68-bus system uses some of the PMUs for AGC. Those PMUs are considered as the primary PMUs. In this system, there are four layers of PMUs determined based the number of hops from a PMU to any of the primary PMUs.

For this system, we tested under ten different PV penetration levels, shown in Table 7.2, and 100 different attacks under each operating condition for a total of 1000 experiments. Start from no PMUs under attack; every two following attack conditions attacks one more PMU. For example, no PMUs are attacked in case 0 and 1. One of the PMUs is attacked in case 2, and 3. 49 PMUs are attacked in case 98 and 99. PMUs closer to primary layer will be attacked first. In this system, 11 PMUs are used for AGC. Those 11 PMUs will be attacked first in attack case 2 to 23. In those attack cases, the PMUs under attack are randomly selected among primary PMUs. There are 14 PMUs in the secondary layer. For attack case 24 to 51, all 11 PMUs in the primary layer is attacked, PMUs in the secondary layer is randomly selected. So on and so forth.

Table 7.2: Ten Operating Conditions.

Operating Condition Case ID	PV1(MW)	PV2(MW)
0	0	0
1	85	170
2	125	250
3	165	330
4	205	410
5	245	490
6	285	570
7	325	650
8	365	730
9	405	810

7.4 Results and Analysis

7.4.1 VSN Resiliency Results on Two Area Four Machines System

Before the attacks, the total energy transferred on the tie-line in one minute E_E is recorded as the benchmark to calculate the total energy change after each attack. The total energy change δ

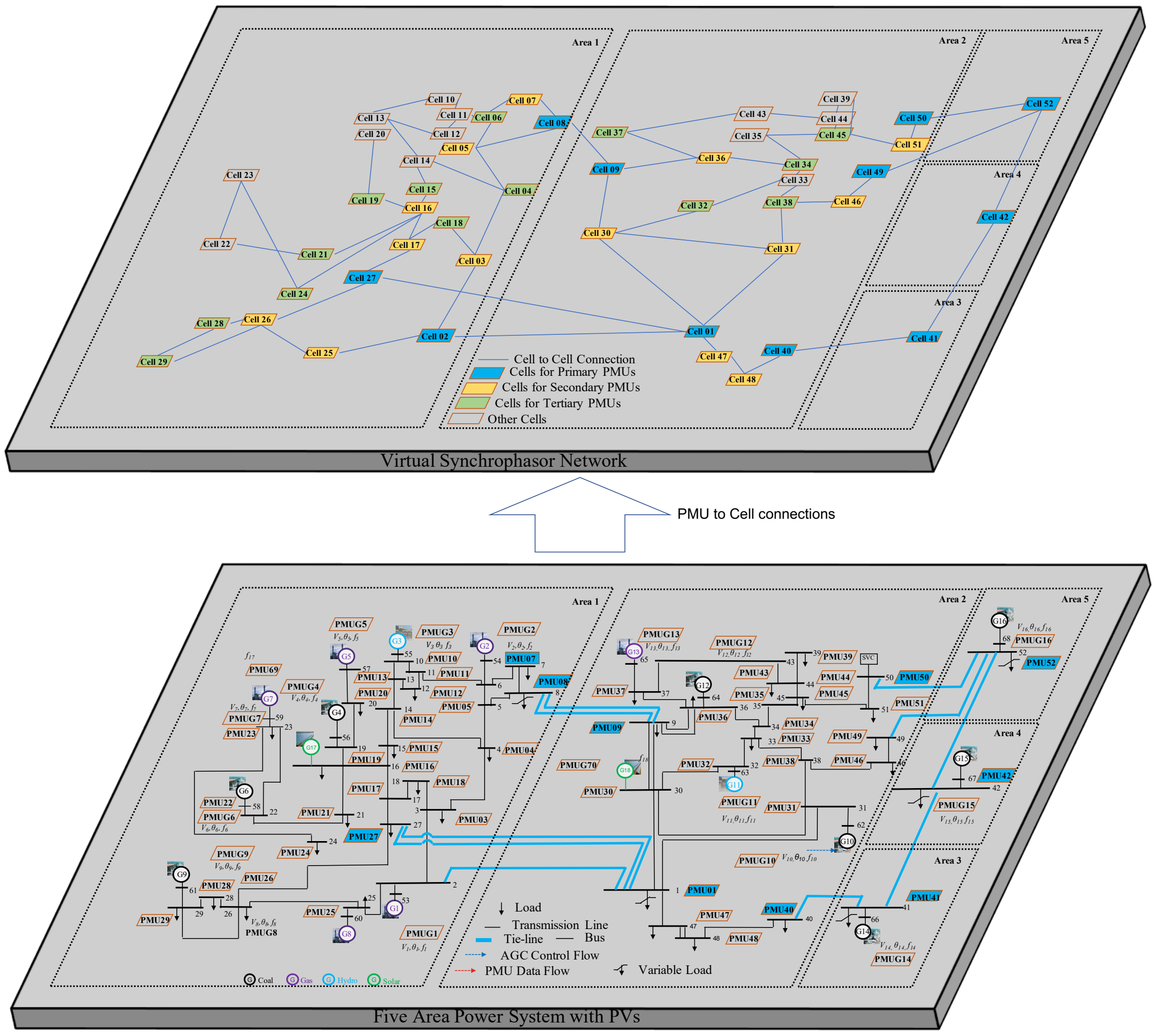


Figure 7.2: The illustration of IEEE 68-Bus System with VSN integrated.

is calculated as follows:

$$\delta = (\int_0^{\frac{1}{60}h} P(t)dt - E_E)/E_E \times 100\% \tag{7.1}$$

Where $P(t)$ is the power measured at the tie-line. The CAPE is calculated only during the DoS attacks and as follows:

$$CAPE = \int_0^{T_{Attack}} |\hat{f_i(t)} - f_i(t)|/60 \times 100\% dt \tag{7.2}$$

Where $f_i(t)$ is the system measured frequency at PMU i, $\hat{f_i(t)}$ is the VSN inferred frequency at Cell i, T_{Attack} is the duration of the attack.

RTDS simulates power system in real-time, and each attack experiment last around 10 minute. It is not feasible to calculate confident interval with hundreds of simulations. To get relatively reliable results, all variables mentioned in above formulas are calculated using an average of the results produced in 10 experiments.

The average of total energy transfered on the tie-line in one minute with a fault but without any attack is 3.3321 MWh.

Figs. 7.9 and 7.10 show the average of the CAPE for each experiment. As expected, the longer the attack, the higher the CAPE. However, with more PMUs under attack, the CAPE is not necessarily getting higher in some cases. By comparing with Figs. 7.3 to 7.8, it is obvious that the VSN can perform very well when only primary layer PMUs are attacked. With more PMUs and even all PMUs except PMU 12 attacked, the VSN can still imply the actual frequency trend.

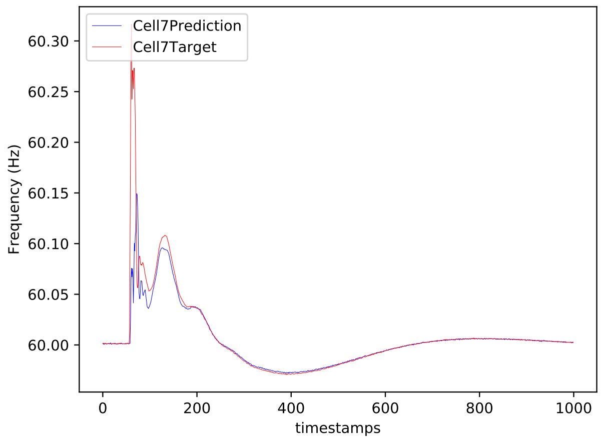


Figure 7.3: Frequency response at Bus 7 during attack on primary layer PMUs for 20 seconds.

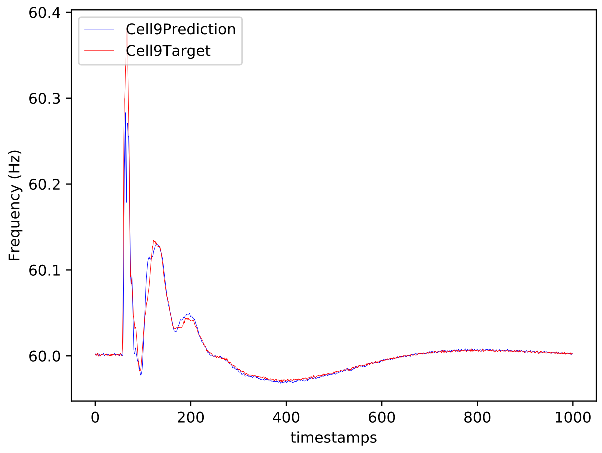


Figure 7.4: Frequency response at Bus 9 during attack on primary layer PMUs for 20 seconds.

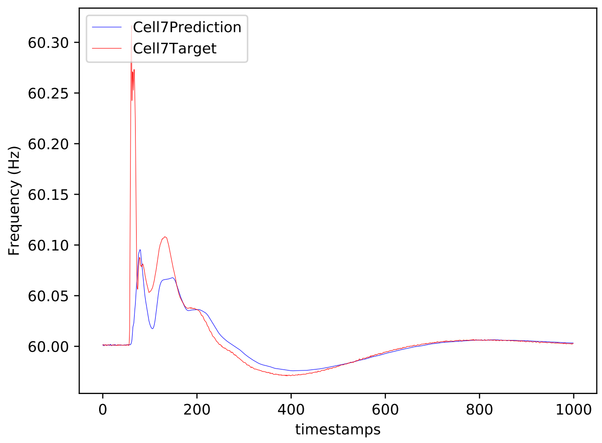


Figure 7.5: Frequency response at Bus 7 during attack on primary layer and secondary PMUs for 20 seconds.

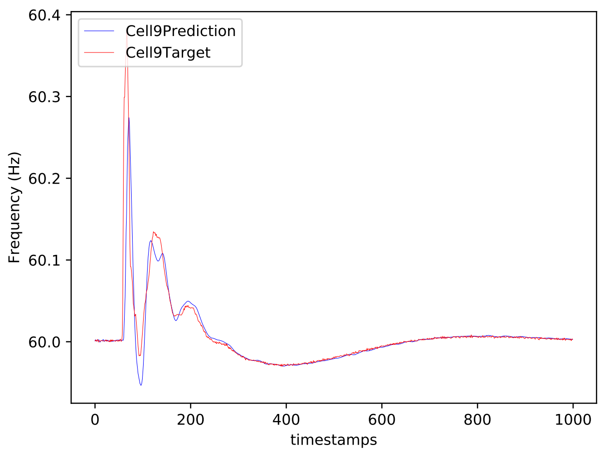


Figure 7.6: Frequency response at Bus 9 during attack on primary layer and secondary PMUs for 20 seconds.

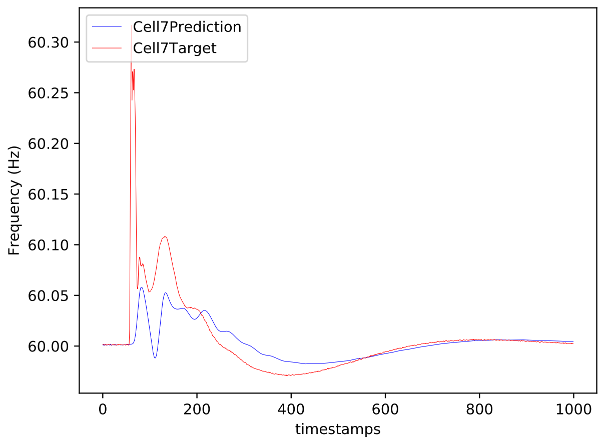


Figure 7.7: Frequency response at Bus 7 during attack on all PMUs except PMU 12 for 20 seconds.

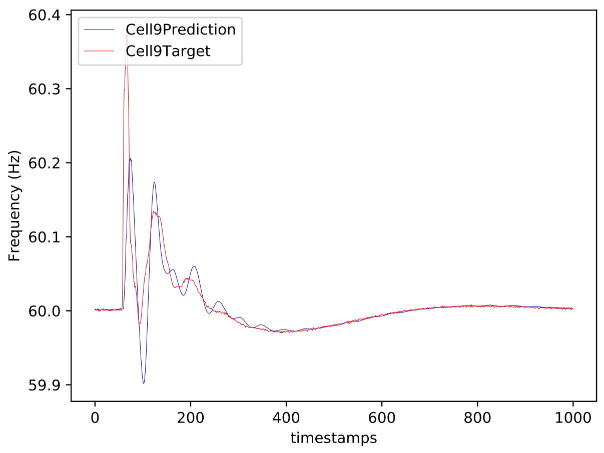


Figure 7.8: Frequency response at Bus 9 during attack on all PMUs except PMU 12 for 20 seconds.

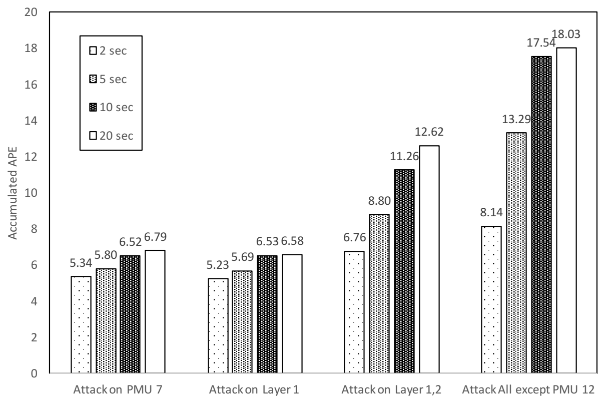


Figure 7.9: The accumulate absolute percentage error of Cell 7 frequency predictions under each attack case.

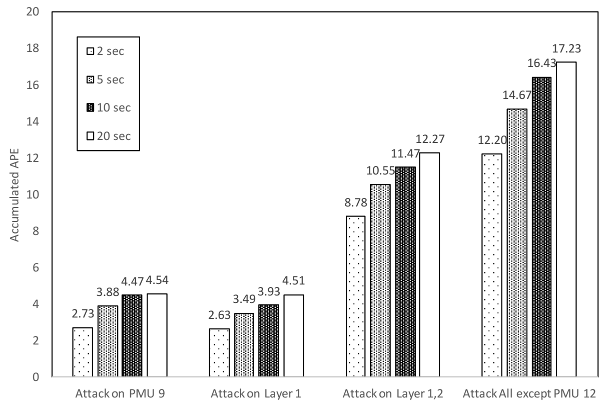


Figure 7.10: The accumulate absolute percentage error of Cell 9 frequency predictions under each attack case.

7.4.2 VSN Resiliency Results on IEEE 68-Bus System

The one minute Control Performance Standards CPS1 was used to evaluate the results in this section. The one minute CPS1 is calculated using the following equation:

$$ACE_{1min}/(-10 * Beta) * MeanDeltaFrequency$$

where ACE_{1min} is the average ace over one minute window; $MeanDeltaFrequency$ is the average of absolute delta frequency over one minute window.

The results are presented in heat-map Figs 7.11 to 7.11. The vertical axis presents different operating conditions. The horizontal axis presents different percentage of PMUs under attacked. The brighter the color, the higher the CPS1. Similar to the results in the two are four machine system, there is no observable increase of CPS1 as more PMUs under attack. That means, the VSN successfully mitigated multiple DoS attacks in IEEE 68 Bus system.

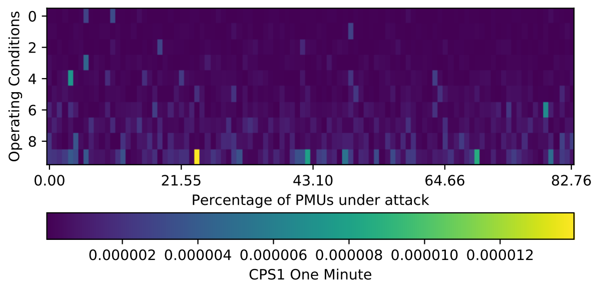


Figure 7.11: CPS1 of area one under different operation conditions and attacks.

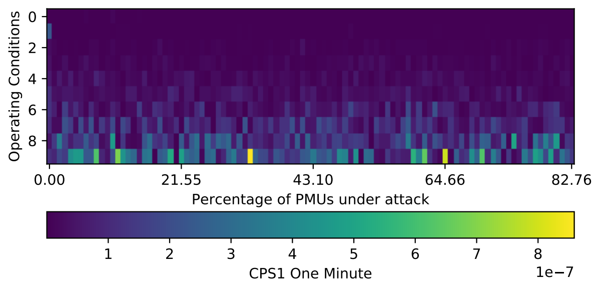


Figure 7.12: CPS1 of area two under different operation conditions and attacks.

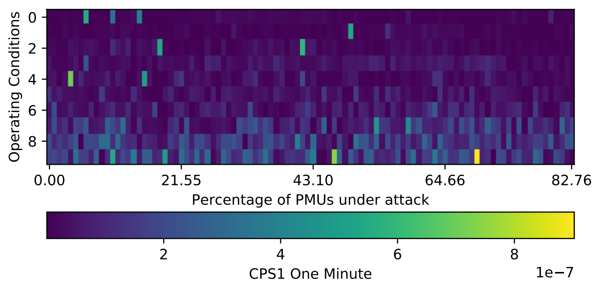


Figure 7.13: CPS1 of area three under different operation conditions and attacks.

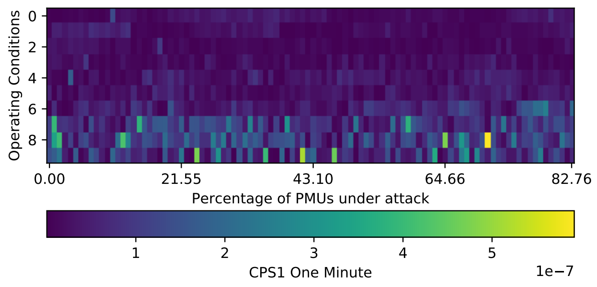


Figure 7.14: CPS1 of area four under different operation conditions and attacks.

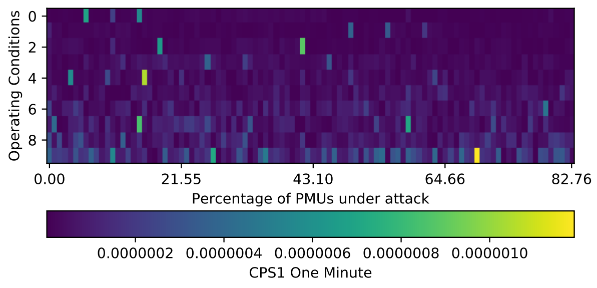


Figure 7.15: CPS1 of area five under different operation conditions and attacks.

7.5 Summary

In this study, two VSNs has been developed to mitigate concurrent DoS attacks during a power system disturbance in two area four machines system and IEEE 68-bus system respectively. A set of DoS attacks are performed against the developed VSN with system faults to evaluate that the VSN can be used to mitigate DoS attacks. The experiments and results prove the robustness of the VSN technology. The VSN is an ideal countermeasure to response to delay or missing frequency measurement data.

With more than 80% of PMUs disconnected in either systems for 20 seconds during system faults, the VSN can still infer the system frequency response, and further, makes the tie-line bias control resilient to DoS attacks. The experiments presented in this study demonstrated the capability of the VSN under server DoS attacks. With 7 out of 8 PMUs disconnected for 20 seconds during a system fault, the VSN can still infer the trend of the system frequency response, and further, makes the AGC able to control the power generation and eventual get the system back to stable.

Chapter 8

Border Gateway Protocol Hijacking Detection

8.1 Overview

The Internet is a public network; encryption methods are commonly used to ensure the privacy and authenticity. However, the encryption does not protect the communication route from Man-In-The-Middle (MITM) attacks and Denial of Service (DOS) attacks and does not guarantee the latency of the communication. Any unexpected change of the variance of the packet delay can cause instability in the smart grid.

For communication between different IP prefixes, Border Gateway Protocol (BGP) routing mechanism determines the route. BGP is used by Autonomous Systems (AS) to exchange routing information. Every BGP update message can affect the routing of the entire Internet. Usually, the traffic sender and receiver will not be aware in advance of the change caused by BGP traffic engineering. Also, the BGP routing mechanism is subject to MITM attacks. This is usually called BGP hijacking attack.

BGP attacks usually follow with other attacks, such as Denial of Service (DoS) attacks. Thus, the BGP hijacking detection can be used as a pre-alarm for possible attacks shortly. BGP hijacking is one of the easiest ways to perform Man In The Middle (MITM) attacks. Establish air gaped networks for the smart grid is not practical, and PMU to PDC communication is subject to

BGP hijacking attack either intentional or unintentionally. However, no existing method can quickly detect a BGP route change.

In this chapter, we present experiments to study the performance of using CUSUM and F-Test algorithms to detect BGP hijacking attacks on the PEERING testbed[1]. The PEERING testbed owns multiple BGP routers and IP prefixes on the Internet. Thus, this unique testbed is capable of performing real attacks on the real Internet without causing real damages. The attacks hijacked a PMU to PDC communication route and caused additional latency to the data transmission. We used absolute delay side-channel and inter-packet delay side-channel to detect BGP route update.

8.2 Border Gateway Protocol in Smart Grid

In a distributed system such as smart grid, information needs to be shared between subsystems. Those messages need to be delivered timely in-order for the system to make real-time decisions. For long distance communication, the Internet is a common choice of communication media. Network traffic over a long distance will travel through a backbone network. A backbone network might be shared among different service providers. The network traffic for smart grid communication will be transmitted through those backbone networks and thus shared with other network traffic.

The network traffic route on the backbone network is subject to change. The route selection depends on a variety of parameters, such as bandwidth, latency, load, pricing, contract, and even misconfigurations or BGP hijacking attacks. Packets travel on the Internet will go through many devices. Many of the devices are ignoring network probing packets for security reasons. Thus it is impossible to know if a route is updated. Once a network route is changed, the latency usually changes as well. Figs 8.1 and 8.2 shows some incidences of latency change observed when transmitting PMU measurement traffic over the Internet.

8.3 Border Gateway Protocol Hijacking

The BGP hijacking attacks are performed using the PEERING testbed. The PEERING owns multiple IP prefixes, AS numbers and BGP routers on the Internet. The testbed provides authorized users to access and control the BGP routers controlled by PEERING. Authorized users

[1]PEERING is a system that provides safe and easy access for researchers and educators to the Internet's BGP routing system, enabling and inspiring transformational research. https://peering.usc.edu [83]

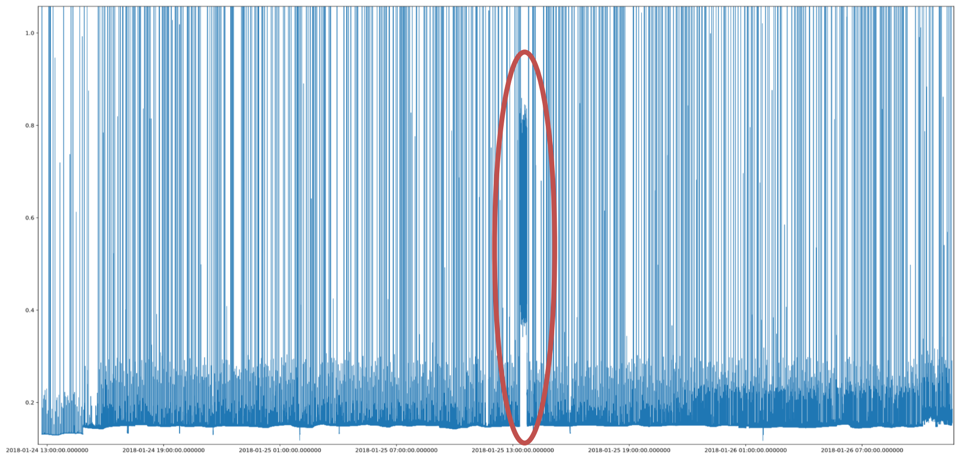


Figure 8.1: The latency is changing over time.

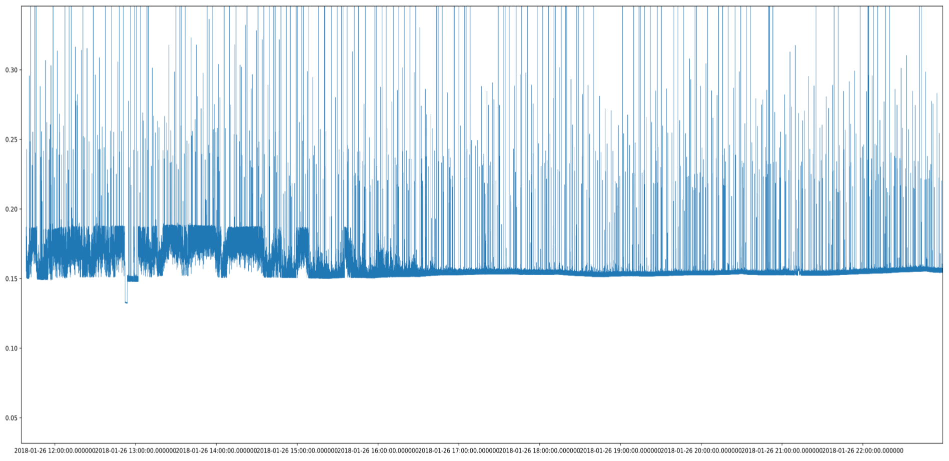


Figure 8.2: An incidence of an unexpected latency change that might cause by a network traffic route change. The average latency increased to 0.6 seconds. This incidence lasted for about an hour.

can change the routing table and broadcast BGP messages to the Internet through those BGP routers. The messages are checked by the PEERING testbed before send. If the BGP message can potentially harm the Internet, the message will be blocked before sending. In this study, we are allocated to use two prefixes: 184.164.244.0/24 and 184.164.245.0/24. We used two BGP routers to perform the BGP hijacking attack.

During the experiment, we captured the PMU messages and the arriving time of each message. The PMU messages are transmitted by PMUs to an OpenPDC running in a computer using the Synchrophasor Protocol. Each of the synchrophasor messages contains a GPS time tag indicating the time that the packet was sent out. The computer receives the synchrophasor messages can calculate the inter-packet delay and absolute delay of the message. Any pattern change in the latency or inter-packet delay may indicate a possible BGP hijacking attack.

Figs 8.3 shows the four stages during the BGP hijacking attack experiment. In stage one, Fig 8.3a, we announced both prefixes from the BGP router at Phoenix. We configure PMU A to send measurement data to 184.164.245.100 and PMU B to send measurement data to 184.164.244.100. In the second stage, Fig 8.3b, we simulate a BGP hijacking attack by announcing the 184.164.244.0/24 prefix at the BGP router at Amsterdam without withdrawing that prefix from the BGP router at Phoenix. In the third stage, the announcement takes effect, and the traffic from PMU B is routed through Amsterdam.

As not all the broadcast messages are going through the PEERING testbed, we use the arriving time series from PMU A as the control group.

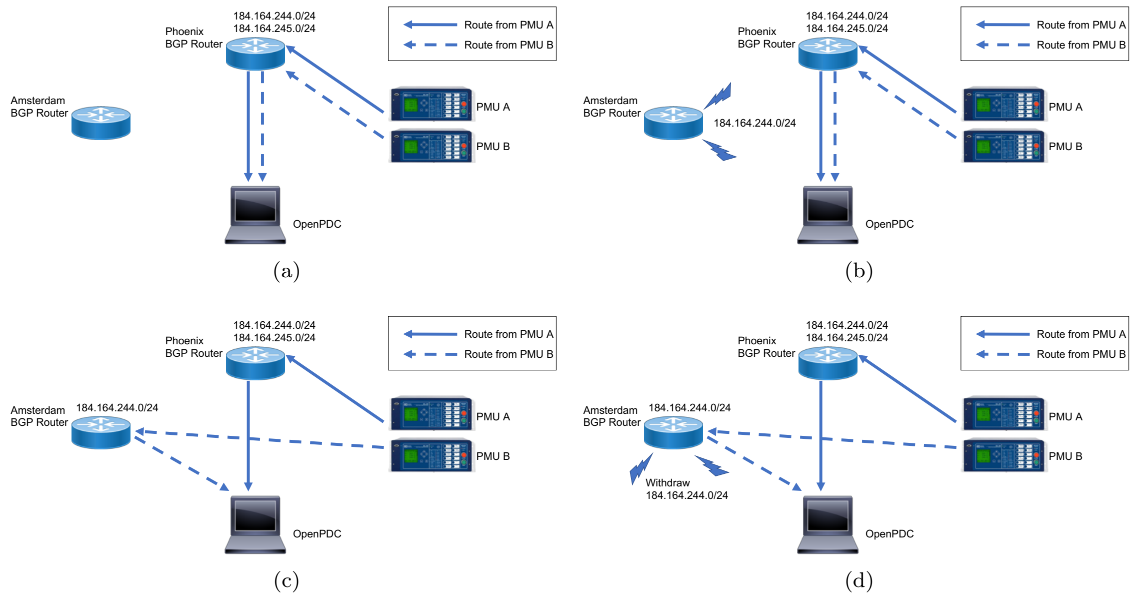


Figure 8.3: The four stages of BGP hijacking attack experiments performed in this study.

8.4 Packet Timing Side-Channel Analysis

In practice, BGP route change might not have a significant effect on the absolute delays. It is necessary to analyze both the absolute delays and inter-packet delays.

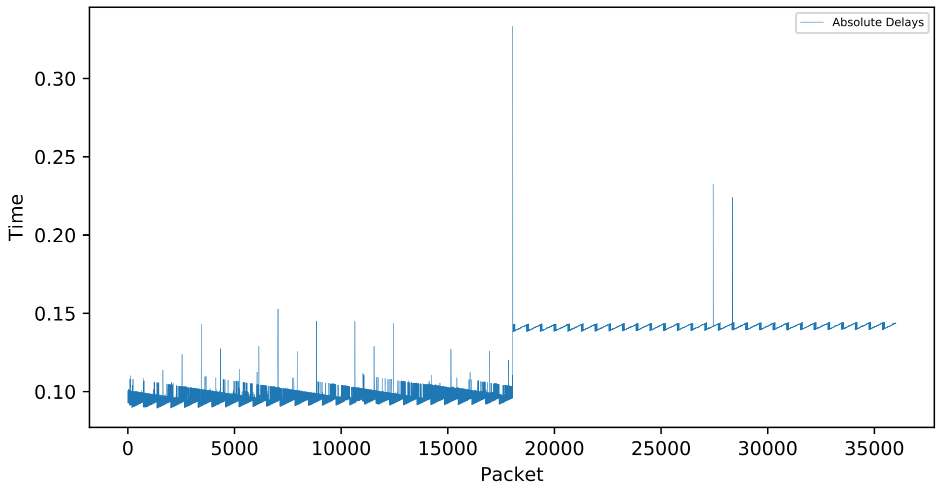


Figure 8.4: The absolute delays of PMU traffic before and after the BGP routing switched from Phoenix to Amsterdam.

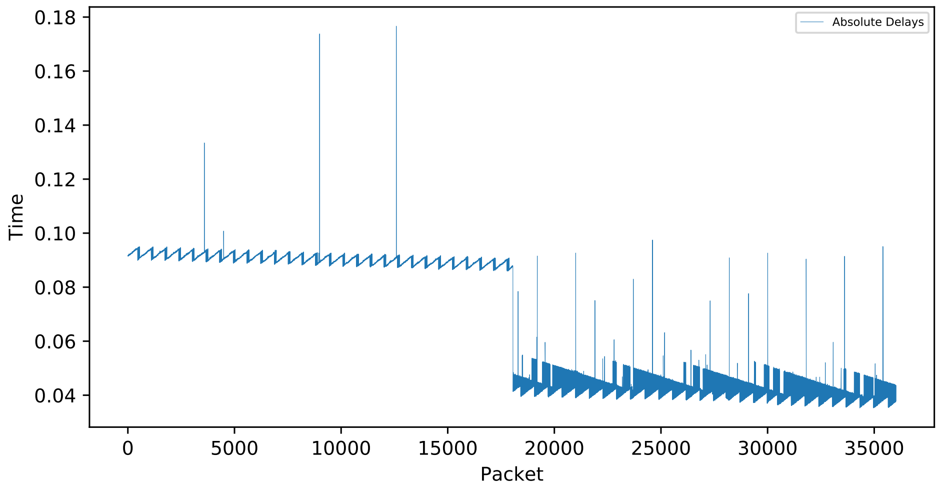


Figure 8.5: The absolute delays of PMU traffic before and after the BGP routing switched from Amsterdam to Phoenix.

Figs 8.4, 8.5, 8.6 and 8.7 shows typical patterns of absolute delays and inter-packet delays during a BGP route change occurred at around 18000 packets. In Figs 8.4 and 8.6, the route from the victim PMU is changed from going through Phoenix to going through Amsterdam. The absolute delays increased; the variance of inter-packet delay decreased. Similarly, in Figs 8.5 and 8.7 shows a decrease of absolute delays; and increase of the variance of inter-packet delay after the BGP hijacking attack finished.

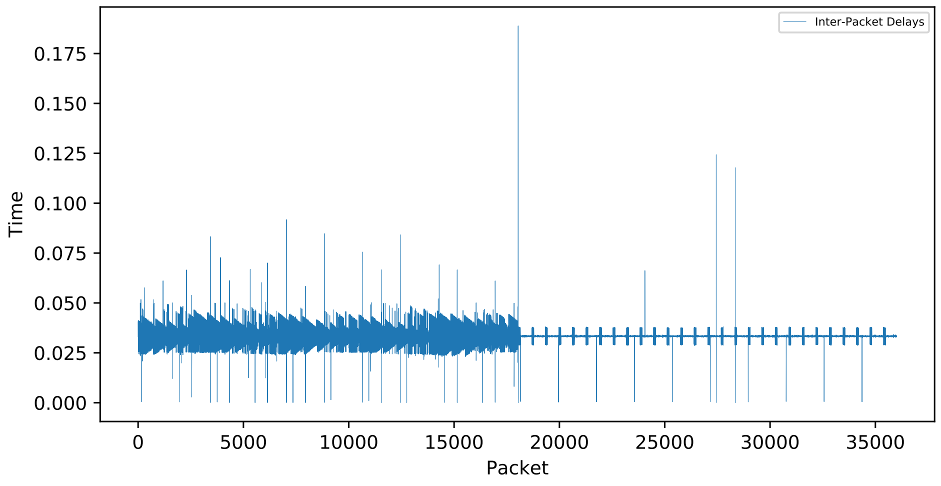


Figure 8.6: The inter-packet delays of PMU traffic before and after the BGP routing switched from Phoenix to Amsterdam

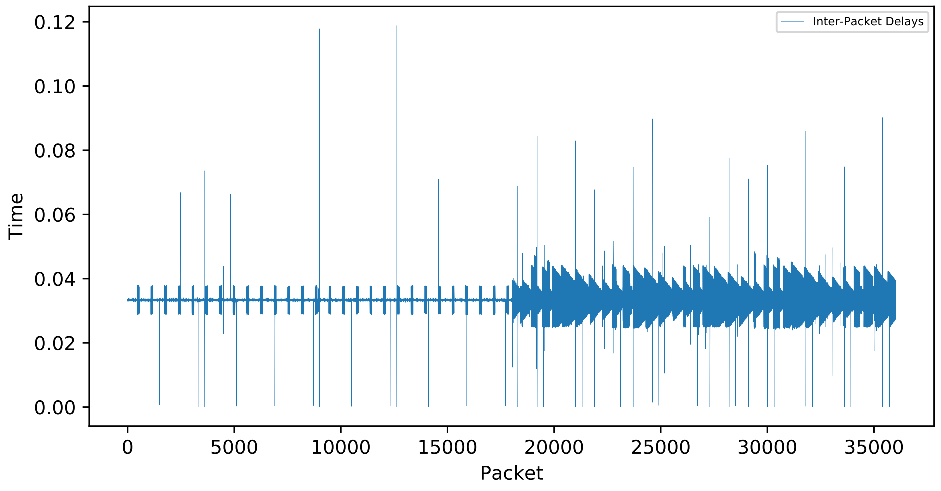


Figure 8.7: The inter-packet delays of PMU traffic before and after the BGP routing switched from Amsterdam to Phoenix.

We use two well-studied algorithms CUSUM [105] and F-Test [55] to perform the analysis. The CUSUM algorithm has been used for Distributed Denial of Service (DDoS) attack detection [81, 106]. The F-Test is a popular statistical test used to detect sample variance change.

8.4.1 Cumulative Sum (CUSUM)

The CUSUM algorithm is used to detect extreme traffic volume increases hidden in bursty background traffic. In the CUSUM algorithm, attacks are detected using a sequential probability ratio test (SPRT) [25].

Here we use a modified version of the CUSUM algorithm based on [81]. Let $L[t]$, $t > 0$, is the latency of packet arrived at time t. If a BGP hijacking starts at time λ, the latency will change:

$$L[t] = \begin{cases} L^0[t] & 0 < t < \lambda \\ L^1[t] & t \geq \lambda \end{cases}$$

If we assume $L^0[t]$ and $L^1[t]$ are independent and their means are $m^0[t]$ and $m^1[t]$ respectively.

The original sequential probability ratio test requires pre and post-change information to solve the problem. To address this issue, we use a modified version based on [106]. The method calculates the difference between the current and long-term average of the observations. If the current average changes faster than the long-term average, the CUSUM coefficient also increases. The CUSUM coefficient goes back to zero when the difference between the two averages is small. When the CUSUM coefficient exceeds the chosen threshold, it indicates a packets latency change which may be caused by a BGP hijacking attack. This CUSUM process can be summarized as:

$$S[t] = max\{0, S[t-1] + |m^S[t] - m^L[t]|\}; \quad S[0] = 0$$

were $S[t-1]$ is the old CUSUM value, $m^S[t]$ is the short window average of packets' latency, $m^L[t]$ is the long-term average of packets' latency, calculated with a given long term average memory parameter ε, $0 < \varepsilon < 1$:

$$m^L[t] = \varepsilon m^L[t-1] + (1-\varepsilon)m^S[t]; \quad m^L[0] = 0$$

To reduce high frequency noise, we use local averaging memory α as a low-pass filter:

$$\tilde{m}^S[t] = \alpha m^S[t] + (1-\alpha)\tilde{m}^S[t-1]; \quad \tilde{m}^S[0] = 0$$

High values for ε make the average emphasize the long term average value in $m^L[t]$. Larger values for α make the average emphasize the current value in $m^S[t]$.

With the above equations, we have the modified CUSUM algorithm:

$$\tilde{S}[t] = max\{0, (\tilde{S}[t-1] + |\tilde{m}^S[t] - m^L[t]| - C)\}; \quad \tilde{S}[0] = 0$$

where C is multiplication of $m^L[t]$ and correction parameter ce which forces the cusum coefficient values to 0by adding more weight to long term average $m^L[t]$.

As an example, we take the absolute latency data presented in Figs 8.4 and 8.5 as inputs for the proposed CUSUM algorithm. Figs 8.8 and 8.9 show the output of the CUSUM algorithm. The output from the proposed CUSUM algorithm generate a high peak quickly after a BGP route change.

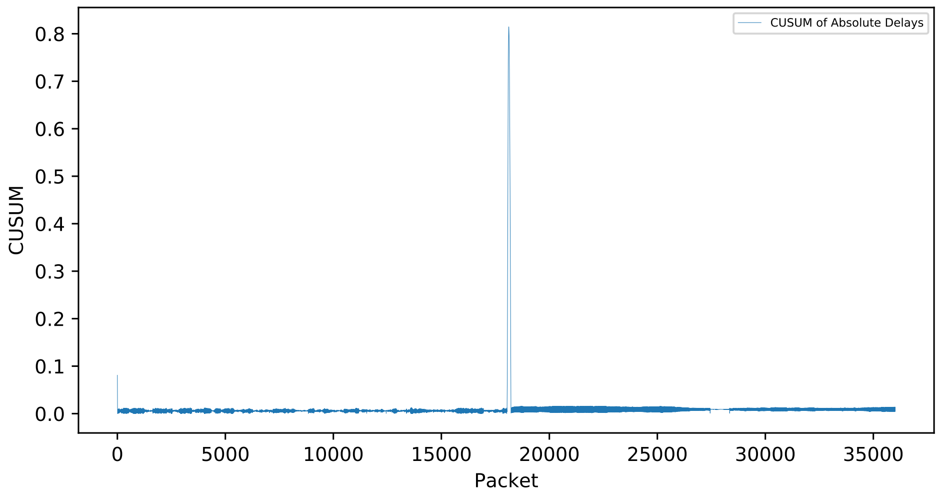


Figure 8.8: The CUSUM of absolute delays of PMU traffic before and after the BGP routing switched from Phoenix to Amsterdam.

8.4.2 F-Test

F-Test compares the variance of two data sets. We use similar approach as the modified CUSUM algorithm. We let $v^S[t]$ be the variance of the data in a short term window at time t, $v^L[t]$ be the variance of the data in long term window at time t. Then we have our F-Test algorithm as

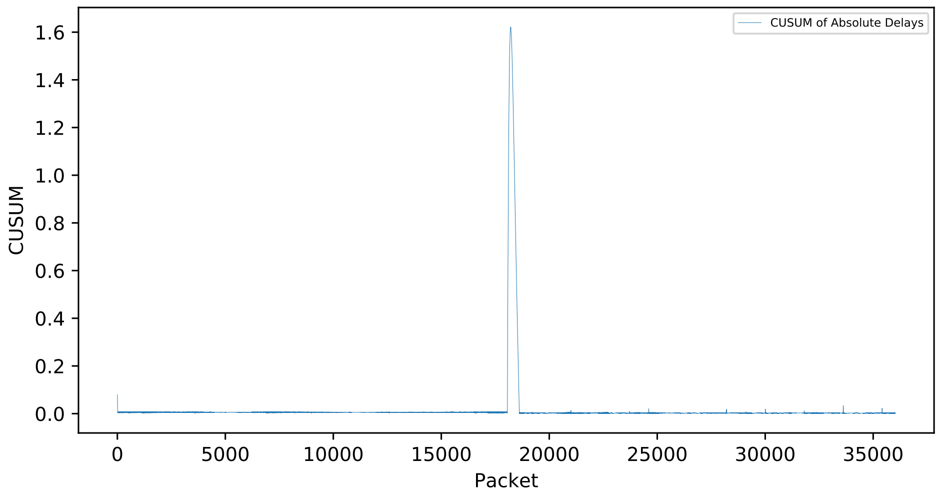


Figure 8.9: The CUSUM of absolute delays of PMU traffic before and after the BGP routing switched from Amsterdam to Phoenix.

follows:

$$F[t] = |v^S[t] - v^L[t]|$$

As an example, we take the absolute latency data and inter-packet delay data presented in Figs 8.4, 8.5, 8.6 and 8.7 as input for the F-Test algorithm. The results are shown in Figs 8.10, 8.11, 8.12, and 8.13. The F-Test generate a high peak a moment later after a BGP route change. However, the F-Test generates a lot of noises. We can see some small peaks before the BGP route change, shown in Fig 8.11 in. By looking at the original inter-packet delay data shown in Fig 8.6, we can see that the variance are gradually changing before the BGP route change.

8.5 Results

We apply the above described CUSUM algorithm and F-Test algorithm on the absolute delay and inter-packet delay of the captured traffic. ROC curves are created based on the attack records, shown in Figs 8.14 to 8.17.

The ROC curves show that the proposed CUSUM algorithm is capable of detecting the BGP route change using the absolute delays of traffic with high True Positive Rate (TPR) and low False Positive Rate (FPR). However, as the mean of inter-packet delays is not affected much by the BGP route change, the CUSUM algorithm provides no meaningful information when analyzing

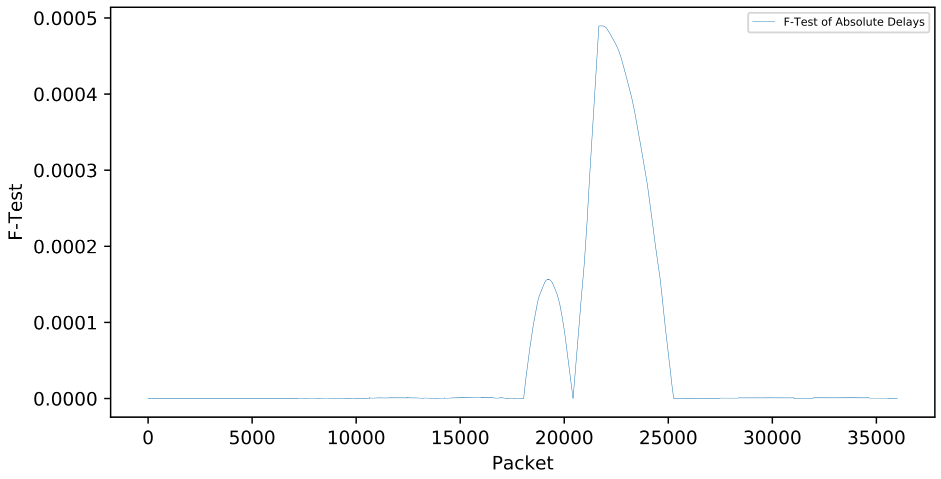


Figure 8.10: The F-Test of absolute delays of PMU traffic before and after the BGP routing switched from Phoenix to Amsterdam.

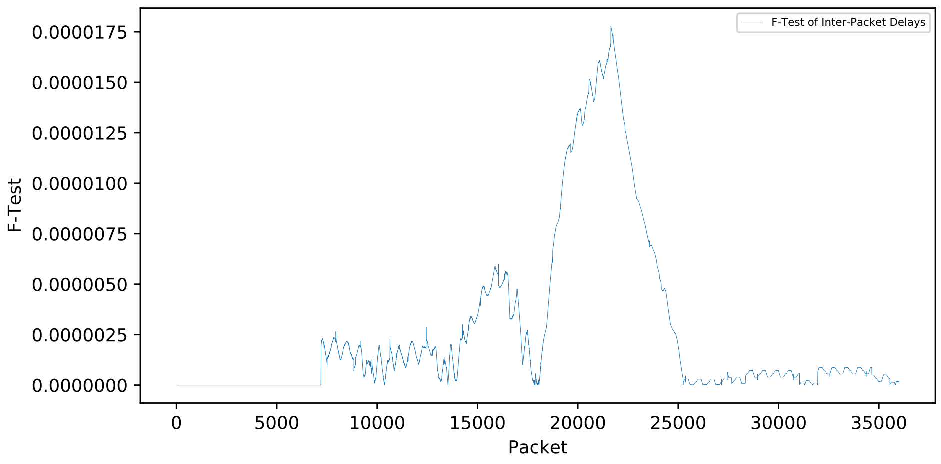


Figure 8.11: The F-Test of inter-packet delays of PMU traffic before and after the BGP routing switched from Phoenix to Amsterdam.

the inter-packet delay data. The proposed F-Test can detect BGP route change on both the inter-packet delay data and the absolute delay data. Compare to the CUSUM test; the F-Test provides higher TPR and FPR on the inter-packet delay data. Further analysis is needed to combine the two algorithms.

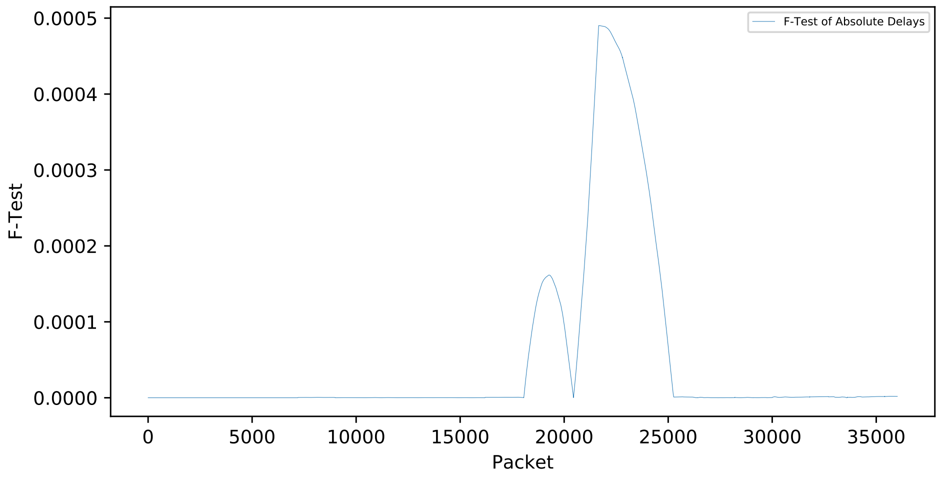


Figure 8.12: The F-Test of absolute delays of PMU traffic before and after the BGP routing switched from Amsterdam to Phoenix.

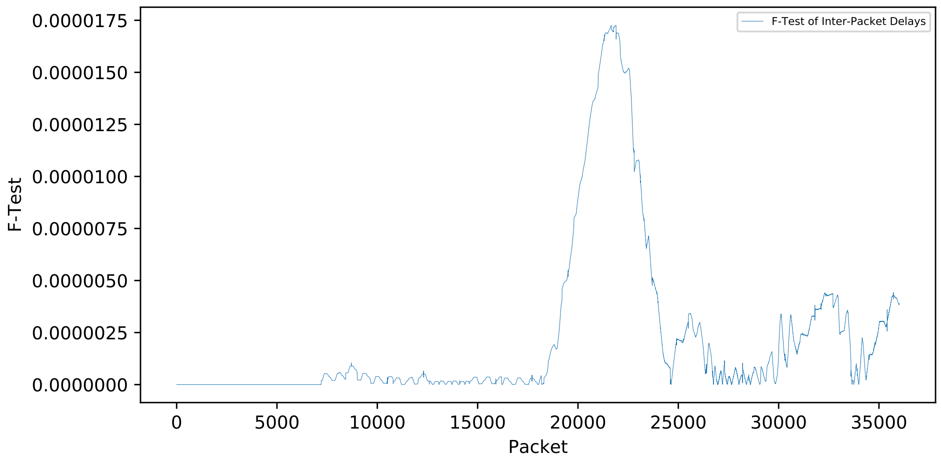


Figure 8.13: The F-Test of inter-packet delays of PMU traffic before and after the BGP routing switched from Amsterdam to Phoenix.

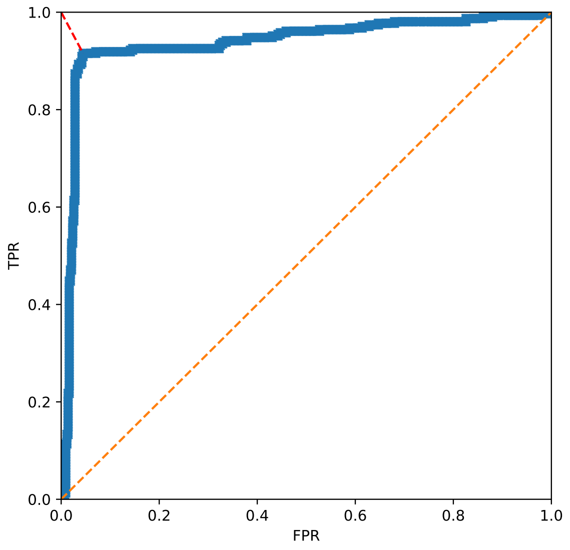


Figure 8.14: ROC of applying CUSUM algorithm on the PMU packets' absolute delay for BGP hijacking detection. The best operation point gives 91.56% TPR and 4.47% FPR

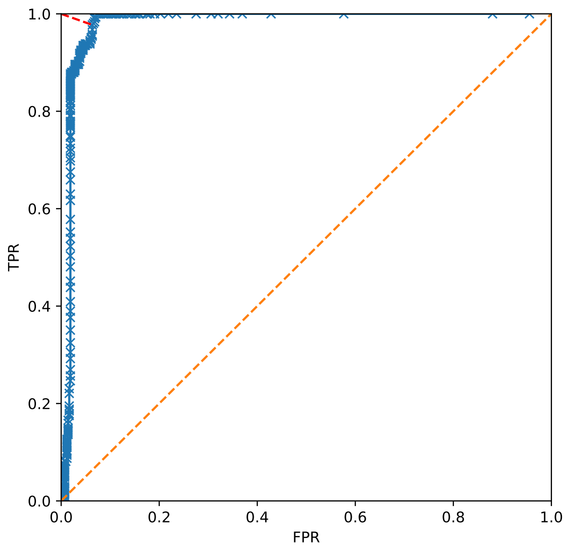


Figure 8.15: ROC of applying F-Test algorithm on the PMU packets' absolute delay for BGP hijacking detection. The best operation point gives 97.72% TPR and 6.35% FPR

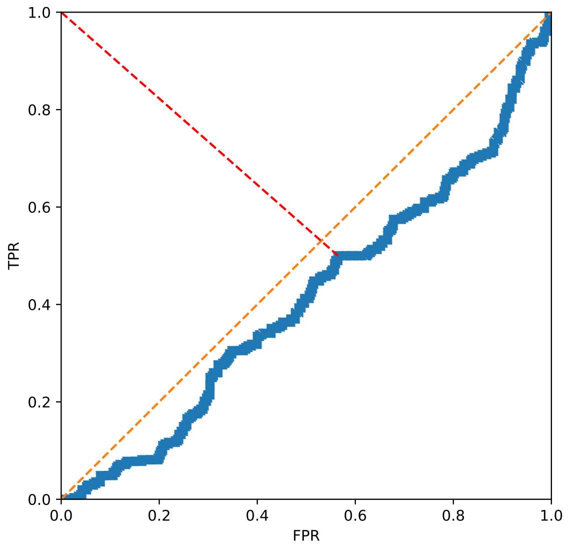


Figure 8.16: ROC of applying CUSUM algorithm on the PMU packets' inter-packet delay for BGP hijacking detection. The best operation point gives 50% TPR and 56% FPR

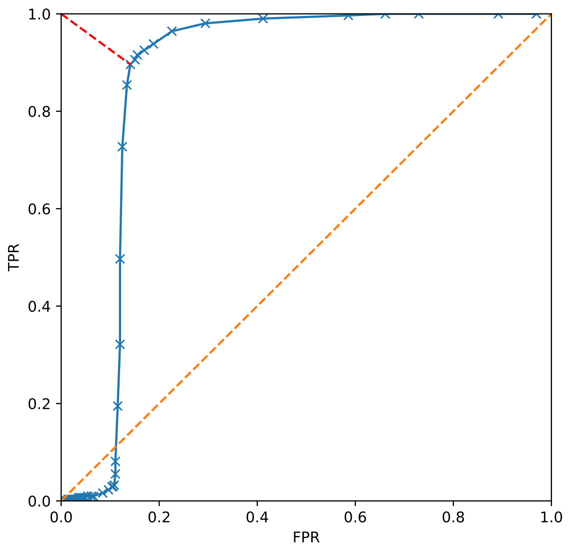


Figure 8.17: ROC of applying F-Test algorithm on the PMU packets' inter-packet delay for BGP hijacking detection. The best operation point gives 89.61% TPR and 14.12% FPR

8.6 Summary

The absolute delays and the inter-packet delays of the captured traffic shown some interesting pattern. As expected, the absolute delays of traffic when routed through Amsterdam is generally longer than it takes when routed through Phoenix. However, the Phoenix route has higher variance, and the timing patterns are changing over time. That pattern could be the result of an artifact of that specific path. Due to the lack of data source, we could only perform experiments using the above-mentioned routes. More data should be acquired in future research to establish a more concrete conclusion.

Chapter 9

Game Theory Analysis

9.1 Overview

The smart grid is a distributed system that expected to be capable of satisfying stringent system safety and performance requirements. The communication system in the smart grid is usually maintained by the third party service providers. In the previous chapters, we discussed several communication network vulnerabilities in a smart grid and mitigation methods such as VSN. In this chapter use the game theory-based methodology to develop a mitigation strategy that can tolerate the worst case scenario.

A smart grid relies on a variety of data inputs, including remote PMU measurements. The PMU measurements are transmitted through the Internet, which presents a large attack surface. Our threat model takes into account the data communication route hijacking and several following attacks after a successful BGP hijacking attack. As discussed in Chapter 8, BGP route hijacking is very easy to apply. This type of attacks are frequently happing and sometimes caused by human mistakes. During such attacks, the latency between two subsystems can change abruptly.

To find the best countermeasure strategy, we perform a classic Two-Person Zero-Sum (TPZS) game [29] analysis. We established multiple attack strategies and mitigation strategies. As AGC requires accurate and timely frequency data, we use the frequency estimation error to calculate the payoff matrix.

9.2 Two-Person Zero-Sum (TPZS) game

Two-Person Zero-Sum (TPZS) game is a classic non-cooperative finite game theory model. In this model, two players have opposite interests. Each player has one or more strategies. With one player's strategies counterparts the other player's strategies, a payoff matrix can be built. For each combination of strategies from both players, the sum of one player's gain and the other player's loss is zero.

In this study, we assume the frequency data are used for AGC. The measured or estimated frequency used for AGC could be different from the real frequency. We use the accumulated frequency differences over 90 seconds, Δf, as the evaluation metrics for the payoff matrix. Smaller Δf will lead to better power system stability. The attacker wants to disturb the power system and thus prefer larger Δf.

To solve the game, we need to test if a saddle point exists for the available attacks and countermeasures. A saddle point is where the strategy combination caused the minimax Δf among all attacks is also the same combination that caused the maximin Δf among all countermeasures. In another word, neither party in the game can do better that the combination. If there is no saddle point, the game can be solved by finding Nash Equilibrium strategies.

9.2.1 Payoff Matrices

Here we give a formal definition of the payoff matrices. In a power system, a set of PMUs P are deployed at some Buses B measuring frequencies. At each time point t, for each Bus i in B, the real frequency, or the ground truth, is denoted as $F_g^i(t)$. At the same time point, a frequency value for each Bus can be retrieved from PMU, or estimated using one of the mitigation strategies. This estimated frequency at time t is denoted as $F_e^i(t)$. The delta between the estimated frequency and the real frequency at time t for Bus i can be calculated by $|F_e^i(t)-F_g^i(t)|$. Then the accumulated delta frequency among all Buses over 90 seconds of operation can be calculated as the payoff function. The above mentioned payoff function can be expressed as the following formula:

$$\sum_{t=0}^{t=90} \sum_{i \in B} |F_e^i(t) - F_g^i(t)|$$

9.2.2 Attacks and Countermeasures

The analyzed strategies for the attackers are BGP hijacking attack; drop PMUs; random packet delays. We consider the artifact of BGP hijacking attack causes longer delay. The drop of PMUs attack drops victim PMUs' data measurements. The random delay might happen when the attacker is performing some analysis on the hijacked traffic.

The available countermeasures are: use zero to replace missing or delayed frequency data; use sixty to replace the missing or delayed data; use the last available data to replace the missing or delayed data; use the SCADA system; and finally using VSN's output to replace the missing or delayed data. The SCADA system uses Remote Terminal Units instead of PMUs and operates at $0.5Hz$. The SCADA system will not be affected hen PMUs are attacked.

Please note that, without any attack, packets can still be delayed. In such a case, one of the countermeasure strategies can be applied to fill the delayed data. The frequency of such delay is estimated using the real PMU operation data captured in Chapter 8.

9.2.3 Z-Test

The evaluation of each attack and countermeasure pair are estimated based on the simulation of the IEEE 68 Buses System. First, we perform power system simulations under 100 different operating conditions without any attack or countermeasure. We captured the frequency measurements on 58 PMUs located throughout the power system. Then, we replay the captured measurement data and apply simulated attacks and countermeasures. The simulated attacks alter the packets' delivery time. The modified traffic is then used as the input for mitigation strategies. Each attack and mitigation pair is evaluated under all 100 different operation conditions. The average, variance and the 95% confidence interval are calculated for each combination.

To determine whether one strategy is better than the other, Z-Test is used. Z-Test can be used to estimate the null hypothesis of two data set. With two independent data set X and Y, the hypothesis of $\mu_x = \mu_y$ against $\mu_x \neq \mu_y$ of the means of X and Y with unknown variances σ_x^2 and σ_y^2:

1. Determine the level of significance α and the critical value $Z_{a/2}$ such that $\Phi(Z_{a/2}) = 1 - a/2$, where $\Phi(Z)$ is the standard normal distribution. Here, we pick $\alpha = 0.05$, and thus the critical value $Z = 1.96$

2. Compute the means $\overline{x}$ and $\overline{y}$, squared variances, s_x^2 and s_y^2, of the samples

3. Compute the test statistic $z = \frac{\overline{x}-\overline{y}}{\sqrt{\frac{s_x^2+s_y^2}{100}}}$

4. If $|z| > za/2$, then reject the hypothesis. Otherwise, do not reject the hypothesis.

Then, if the hypothesis is rejected, two data set can be compared using the mean. Otherwise, both data sets are considered having similar performance.

9.3 Results and Analysis

For each attack and countermeasure pair, 100 experiments were performed. The mean, variance, and confident interval of the gain and lost for both attackers and power system operators can be calculated using the payoff function described in Sec9.2.1.

The payoff matrices are shown in Tables 6 to 12. Tables 6 to 10 each presents the results for one type of countermeasure against different attacks. After located the dominant attacks among all countermeasures, Tables 11 and 12 present the dominant attack against different countermeasures.

The red cells in each table show the dominant attack strategy under different countermeasures. The attack "Drop All PMUs" dominates all other attacks for any countermeasures. Table 11 shows the payoff matrix of all countermeasures against the attack "Drop All PMUs". The SCADA system performs the best among all countermeasures. Please note that this study only focuses on attacks against PMUs. As SCADA is also a cyber-physical system, it is possible for attackers to attack the SCADA communication system, which is out of the scope of this paper.

Thus, the saddle point is found. Where the attacker can attack and drop all PMUs to force the power system using SCADA.

However, in practice, the power system is a distributed system, it is difficult to successfully take down all PMUs. Thus, we perform the analysis by considering less powerful attacks. We found that, when "Primary and Secondary PMUs" or fewer PMUs are attacked, the VSN performs the best. With "Primary Secondary and Tertiary PMUs" under attacked, SCADA performs better. Table 12 shows the payoff matrix of all countermeasures against the second performed attack "Drop Primary Secondary and Tertiary PMUs". In this scenario, the best mitigation strategy is using the VSN.

When no attacks occur, simply use sixty to replace the missing data gives the best results.

9.4 Summary

In this chapter, we performed game theory analysis on multiple attack and mitigation strategy combinations. We identified that to achieve the best attack performance; the attacker should drop as many PMUs as they can. When all PMUs in Primary Secondary and the Tertiary layer or more are dropped, the SCADA system performs the best as the SCADA system is using a different communication channel and not been affected. With all PMUs in the Primary and Secondary layer or less are attacked, VSN provides the best solution. However, when no attack is presented, if only one or two packets are delayed during normal operation, using sixty to fill the gap could be a better solution. As mentioned in previous sections, attacking SCADA communication system is also possible, which is not focused in this study. Further investigation is needed for the case of SCADA system under attack.

Chapter 10

Conclusions

10.1 Overview

With more penetration of renewable energy into the grid, the use of PMUs makes the power system sensitive to communication network service quality. In this dissertation, we show multiple synchrophasor network vulnerabilities as well as mitigations. With the proposed VSN solutions, the resiliency of the power system is improved.

10.2 Chapter Summaries

10.2.1 Side-Channel analysis of PMU Traffic

Two side-channel vulnerabilities were discovered in the PMU to PDC connection. Those vulnerabilities make it possible for attackers to identify encrypted synchrophasor data measurement traffic without decryption. This further leads to the ability to drop certain PMU packets.

10.2.2 Traffic Camouflage

Traffic camouflage technique was proposed to evade the side-channel analysis mentioned earlier. The traffic camouflage technique takes any traffic as input, and transfers the traffic into another protocol. This technique makes it very difficult to identify the protected traffic.

10.2.3 Denial of Service Attack on Phasor Measurement Unit

The effect of dropping PMU connections to the smart grid operation was investigated. The experiment results show that, without any proper mitigation measurements, the power system will quickly go unstable under such attack.

10.2.4 Virtual Synchrophasor Network

VSN was proposed to infer dropped PMU measurement data using historical data as well as neighboring data. The proposed VSN technique is proven effective with one PMU under attack for multiple seconds.

10.2.5 VSN Resiliency

The resiliency of VSN was further investigated in a larger system with more PMUs under attack. In one test case, after 87% of the PMUs are disconnected, the VSN is still capable of producing usable data for AGC.

10.2.6 Border Gateway Protocol Hijacking Detection

BGP hijacking attack and detection experiments bring the above-mentioned experiments to the real Internet. During the experiments, the proposed method identified suspicious BGP hijacking attacks.

10.2.7 Game Theory Analysis

The analysis results identified the best countermeasure and attack strategies. In the worst attack scenario where all PMUs are disconnected, the SCADA system can be used to provide low-resolution data. When all PMUs in the Primary and Secondary layer or less are attacked, the VSN provides the best solution. When no attack is presented, using sixty to replace the occasional delayed packet usually provides a good estimation.

10.3 Future Research Directions

The following topics can be further investigated. As the SCADA system relying on network communication, it is possible that the SCADA system has similar vulnerabilities. Virtual SCADA network can be investigated using the CCN technology. The resiliency of VSN to power system topology changes. A hybrid virtual system that combines the Virtual SCADA and VSN. More BGP traffic experiments with more data point to get a more concrete conclusion about the BGP hijacking detection methods.

10.4 Summary

In this dissertation, the side-channel vulnerabilities in PMU to PDC communication were discovered. By exploit those vulnerabilities, DoS attacks were performed. Traffic camouflage was proposed and implemented to evade side-channel analysis. VSN was proposed and evaluated to be effective to mitigate DoS attacks against PMUs. The resiliency of the above mentioned VSN was then evaluated. BGP hijacking attacks were performed to bring the above attack and mitigation experiments to the real Internet. Finally, game theory analysis gives a full picture of the study and concludes the best mitigation strategies under different attack scenarios.

Appendices

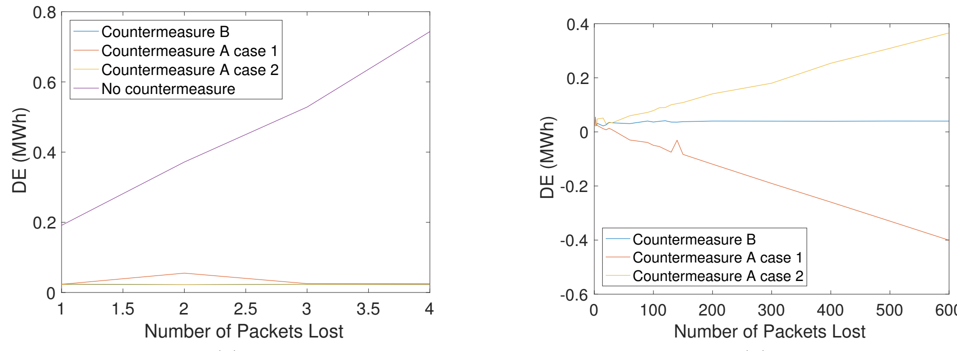
Appendix A Consequences and Mitigation of DoS Attack on Single PMU with VSN

Table 1: Generator Rated values and power outputs

	Rated (MW)	Activate Power (MW)	Reactive Power (MVArs)
Generator G1	900	654	107
Generator G2	900	654	181
Generator G3	900	710	120
Generator G4	900	702	192
PV	210	88	0

Table 2: AGC Parameters

	λ_R	T	k	α_1	α_2
AGC 1	20	0.5	-0.007	0.5	0.5
AGC 2	20	0.5	-0.007	0.5	0.5



(a) (b)

Figure 1: Tie-line energy change ΔE in one minute window after a DoS attack under different scenarios. Please note that the Countermeasure B and Countermeasure A case 2 in figure (a) are overlapped. (a) Four or less packets are blocked. (b) Five or more packets are blocked.

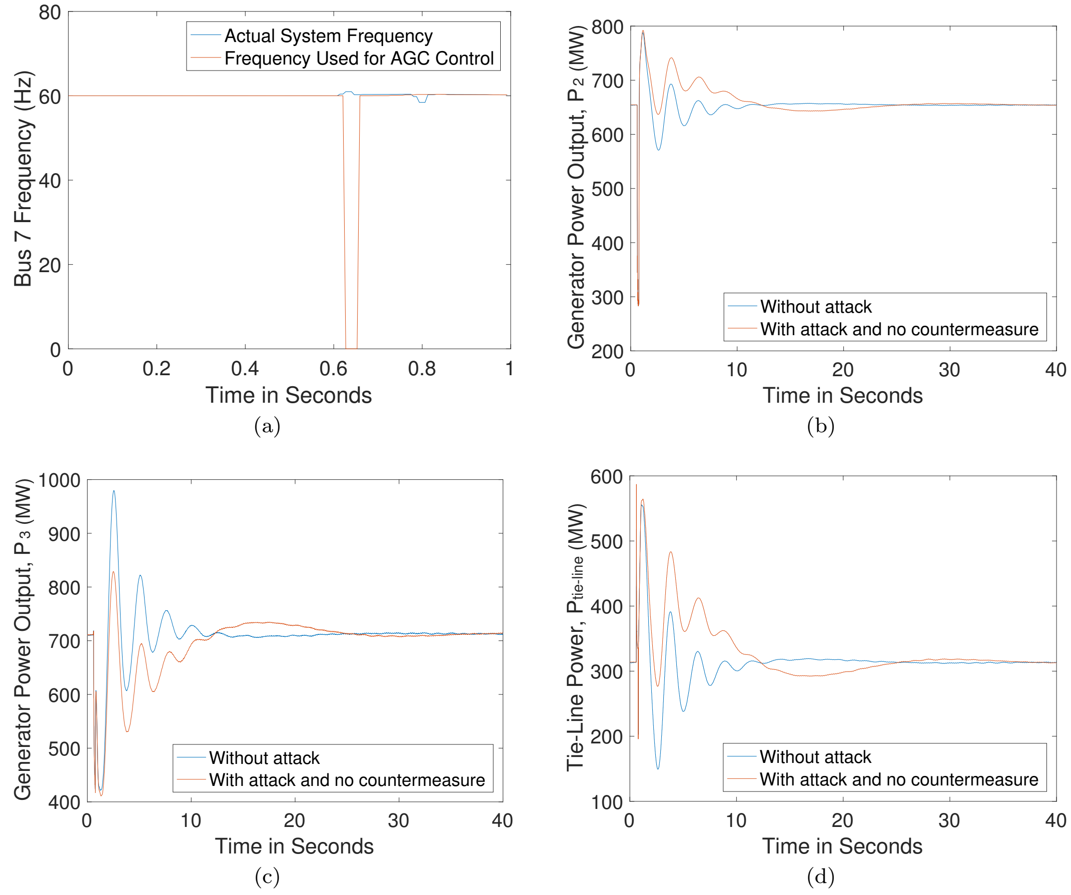
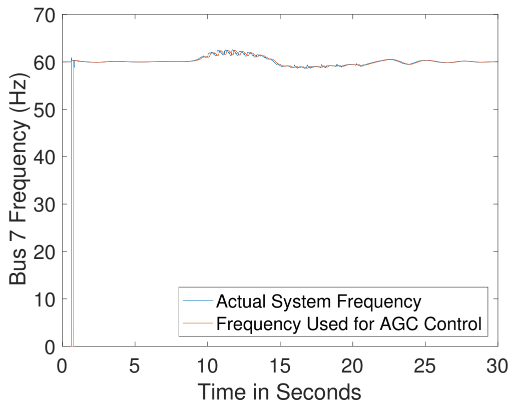
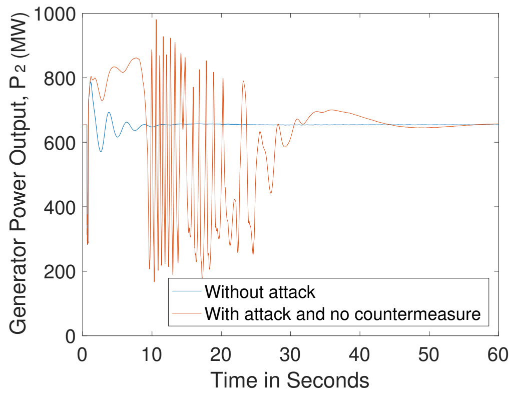


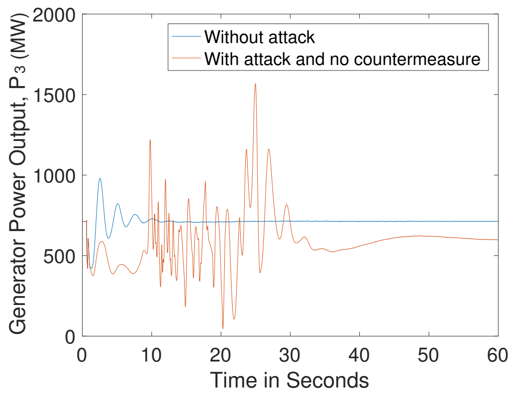
Figure 2: System responses after a fault. One packet from PMU 7 is blocked by a DoS attack. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.



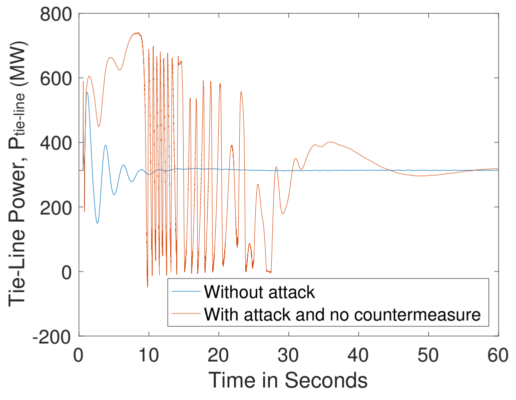
(a)



(b)



(c)



(d)

Figure 3: System responses after a fault. Five packets from PMU 7 are blocked by a DoS attack. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.

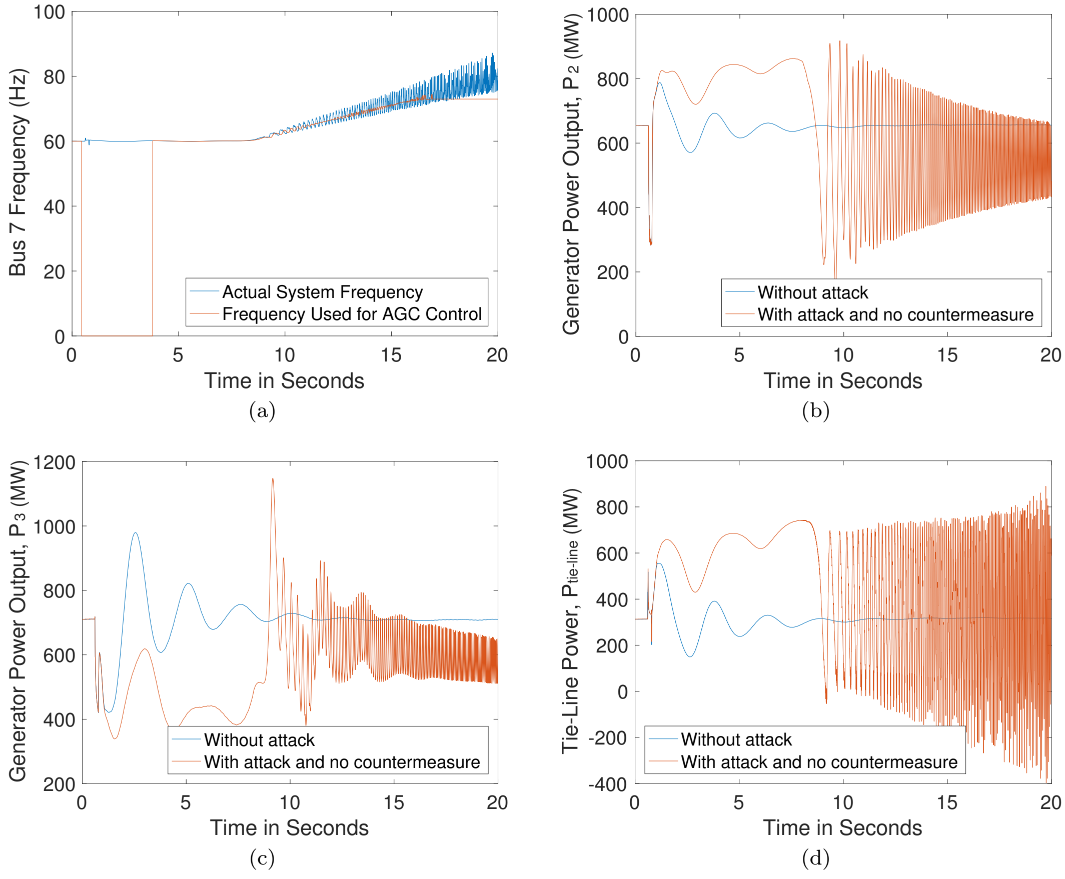


Figure 4: System responses after a fault. 100 packets from PMU 7 are blocked by a DoS attack. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.

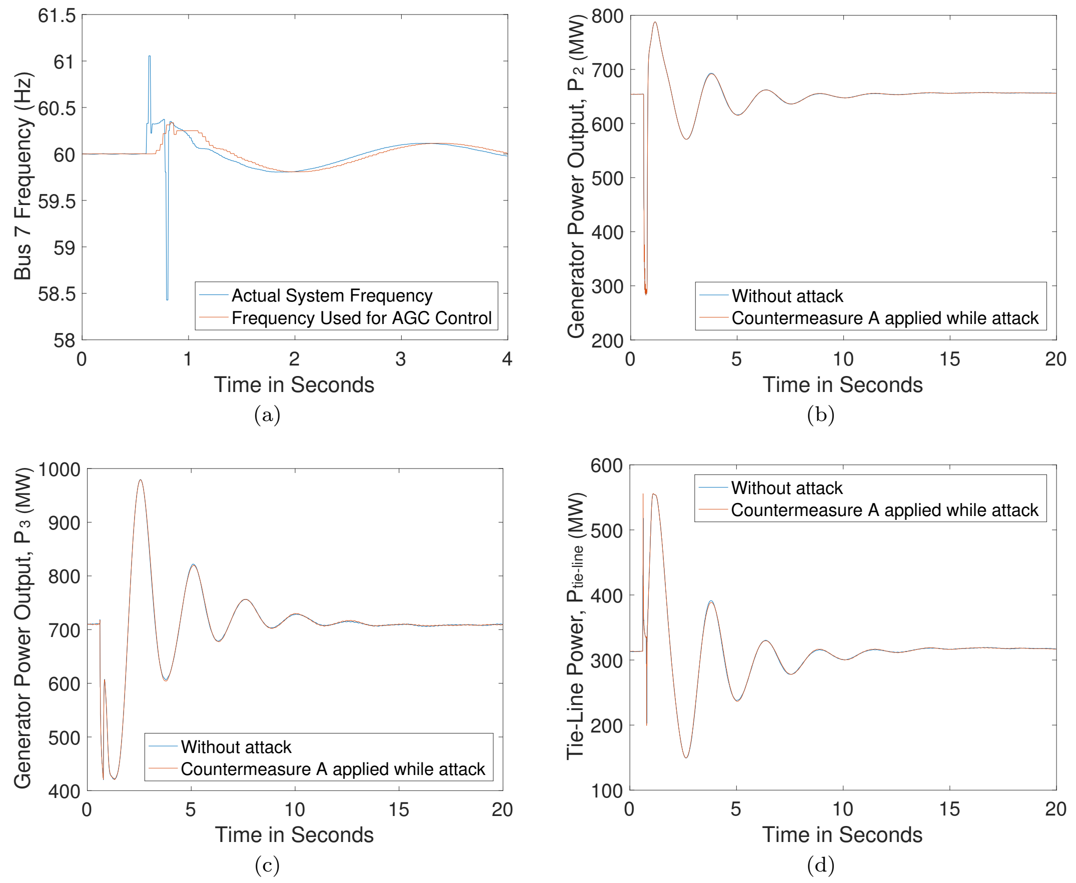
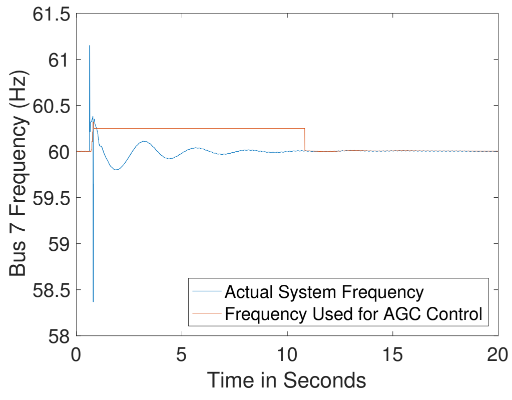
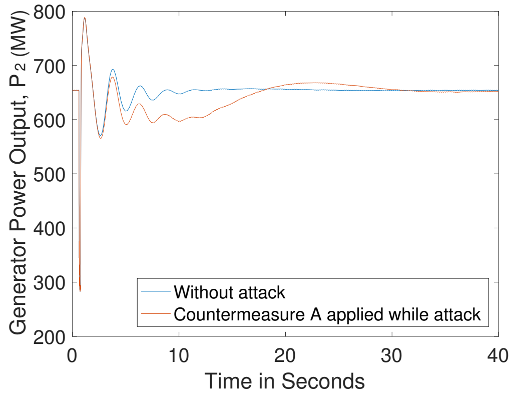


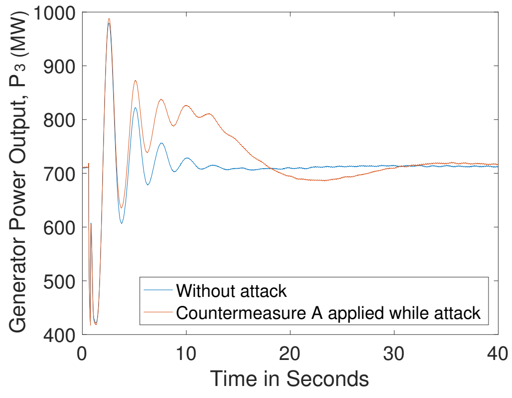
Figure 5: System responses after a fault. Five packets from PMU 7 are blocked by a DoS attack with Countermeasure A applied. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.



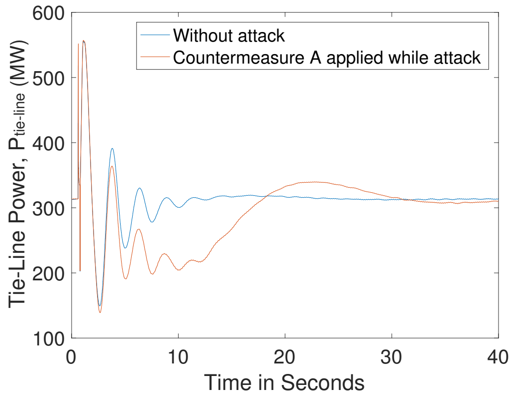
(a)



(b)

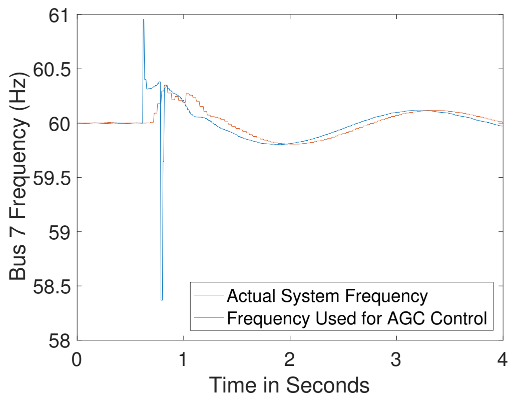


(c)

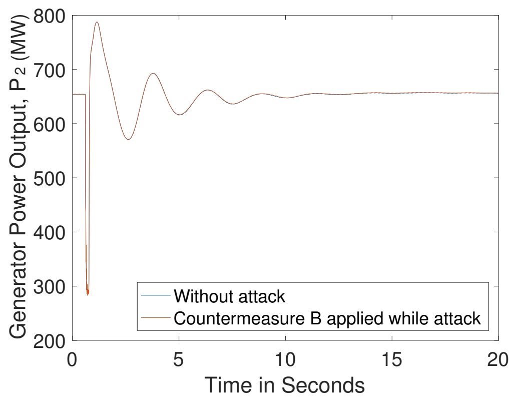


(d)

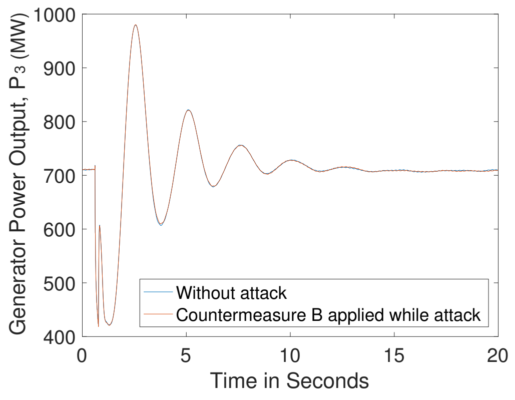
Figure 6: System responses after a fault. 300 packets from PMU 7 are blocked by a DoS attack with Countermeasure A applied. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.



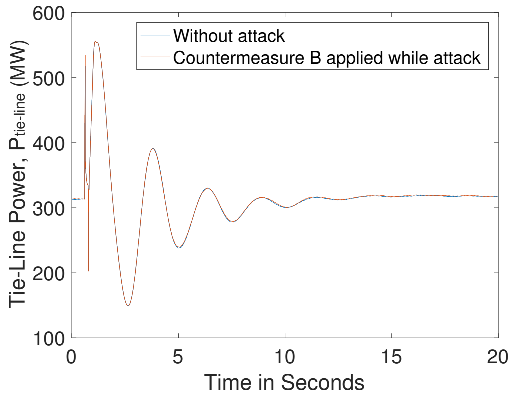
(a)



(b)

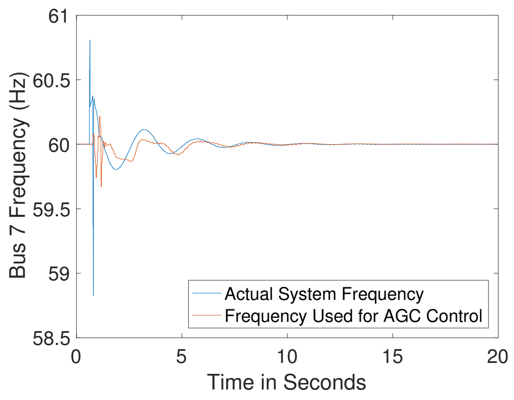


(c)

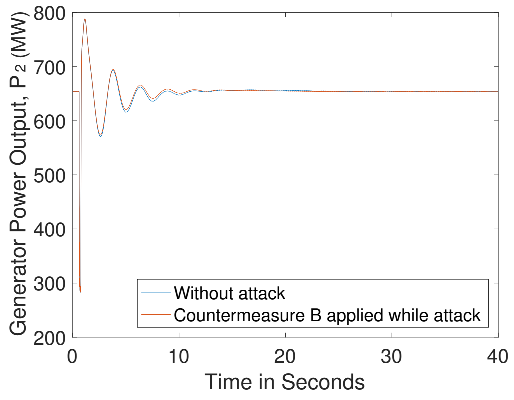


(d)

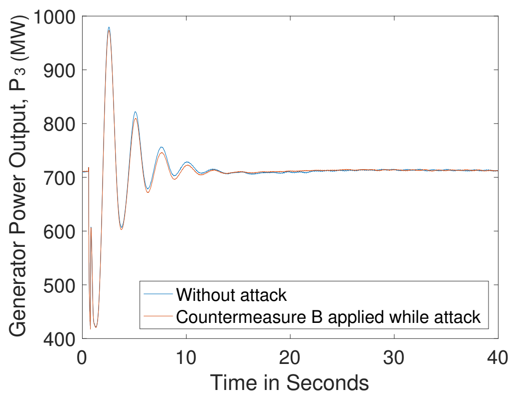
Figure 7: System responses after a fault. Five packets from PMU 7 are blocked by a DoS attack with Countermeasure B applied. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.



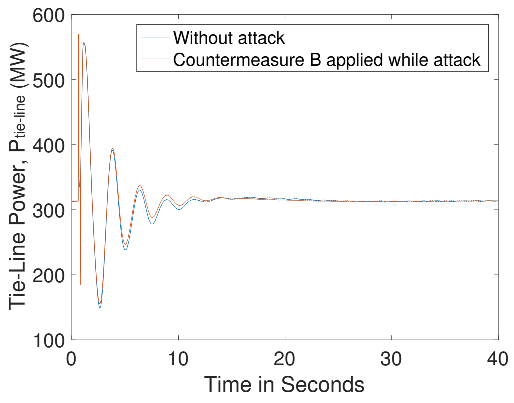
(a)



(b)



(c)



(d)

Figure 8: System responses after a fault. 300 packets from PMU 7 are blocked by a DoS attack with Countermeasure B applied. (a) Frequency response at Bus 7. (b) Power generation of generator 2. (c) Power generation of generator 3. (d) Tie-line power flow.

Table 3: The Change of Energy In One Minutes Window After A Fault

	Number of Packets Lost	Tie Line (MWh)		Generator 1 (MWh)		Generator 2 (MWh)		Generator 3 (MWh)		Generator 4 (MWh)	
No Attack	N/A	0.0240		-0.0128		-0.0138		0.0255		-0.0295	
With Attack	1	0.1912		0.0750		0.0739		-0.1337		-0.0349	
	2	0.3717		0.1699		0.1687		-0.2937		-0.0383	
	3	0.5284		0.2517		0.2494		-0.4195		-0.0402	
	4	0.7434		0.3615		0.3582		-0.5844		-0.0399	
With Attack And Counter -measure A		Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2	Case 1	Case 2
	1	0.0232	0.0228	-0.0132	-0.0135	-0.0142	-0.0145	0.0273	0.0282	-0.0312	-0.0314
	2	0.0551	0.0222	0.0035	-0.0138	0.0025	-0.0148	0.0319	0.0323	-0.0317	-0.0342
	3	0.0252	0.0231	-0.0122	-0.0134	-0.0132	-0.0144	0.0263	0.0282	-0.0298	-0.0325
	4	0.0248	0.0233	-0.0126	-0.0131	-0.0136	-0.0141	0.0265	0.0300	-0.0299	-0.0338
	5	0.0239	0.0480	-0.0128	-0.0002	-0.0138	-0.0012	0.0310	0.0312	-0.0346	-0.0325
	15	0.0119	0.0512	-0.0195	0.0015	-0.0204	0.0005	0.0360	0.0254	-0.0284	-0.0300
	20	0.0077	0.0292	-0.0212	-0.0102	-0.0221	-0.0112	0.0465	0.0201	-0.0357	-0.0302
	25	0.0133	0.0357	-0.0185	-0.0066	-0.0195	-0.0076	0.0405	0.0141	-0.0335	-0.0304
	30	0.0082	0.0334	-0.0212	-0.0078	-0.0222	-0.0088	0.0446	0.0194	-0.0337	-0.0339
	60	-0.0299	0.0597	-0.0412	0.0058	-0.0421	0.0048	0.0803	-0.0060	-0.0317	-0.0341
	90	-0.0390	0.0721	-0.0458	0.0125	-0.0468	0.0115	0.0954	-0.0225	-0.0358	-0.0310
	100	-0.0496	0.0787	-0.0517	0.0160	-0.0527	0.0150	0.0988	-0.0249	-0.0285	-0.0349
	110	-0.0540	0.0892	-0.0540	0.0214	-0.0550	0.0204	0.1035	-0.0029	-0.0289	-0.0647
	120	-0.0647	0.0902	-0.0592	0.0218	-0.0601	0.0207	0.1289	-0.0356	-0.0415	-0.0350
	130	-0.0750	0.1003	-0.0649	0.0272	-0.0658	0.0262	0.1254	-0.0466	-0.0297	-0.0329
	140	-0.0308	0.1046	-0.0477	0.0296	-0.0486	0.0286	-3.6573	-0.0500	3.6189	-0.0336
	150	-0.0831	0.1085	-0.0691	0.0316	-0.0701	0.0305	0.1345	-0.0546	-0.0295	-0.0335
	200	-0.1194	0.1408	-0.0881	0.0487	-0.0891	0.0477	0.1706	-0.0837	-0.0288	-0.0352
	300	-0.1906	0.1801	-0.1258	0.0696	-0.1267	0.0685	0.2417	-0.1281	-0.0276	-0.0444
	400	-0.2596	0.2539	-0.1619	0.1082	-0.1628	0.1071	0.3138	-0.1942	-0.0301	-0.0334
	500	-0.3299	0.3091	-0.1987	0.1372	-0.1996	0.1361	0.3813	-0.2456	-0.0263	-0.0373
	600	-0.4006	0.3660	-0.2359	0.1669	-0.2367	0.1658	0.4514	-0.3038	-0.0255	-0.0344

Table 4: The Change of Energy In One Minutes Window After A Fault

	Number of Packets Lost	Tie Line (MWh)	Generator 1 (MWh)	Generator 2 (MWh)	Generator 3 (MWh)	Generator 4 (MWh)
	1	0.0238	-0.0131	-0.0141	0.0311	-0.0349
	2	0.0221	-0.0139	-0.0149	0.0267	-0.0302
	3	0.0233	-0.0132	-0.0142	0.0256	-0.0290
	4	0.0228	-0.0133	-0.0143	0.0430	-0.0459
	5	0.0317	-0.0088	-0.0098	0.0269	-0.0299
	15	0.0221	-0.0137	-0.0147	0.0307	-0.0344
	20	0.0261	-0.0117	-0.0127	0.0273	-0.0339
	25	0.0347	-0.0072	-0.0082	0.0151	-0.0307
	30	0.0336	-0.0080	-0.0090	0.0186	-0.0334
With Attack And Counter -measure B	60	0.0306	-0.0096	-0.0106	0.0177	-0.0291
	90	0.0403	-0.0044	-0.0054	0.0240	-0.0345
	100	0.0365	-0.0062	-0.0072	0.0157	-0.0331
	110	0.0393	-0.0051	-0.0061	0.0087	-0.0280
	120	0.0412	-0.0039	-0.0049	0.0088	-0.0307
	130	0.0364	-0.0064	-0.0074	0.0166	-0.0341
	140	0.0362	-0.0064	-0.0074	0.0127	-0.0307
	150	0.0378	-0.0056	-0.0066	0.0112	-0.0301
	200	0.0401	-0.0044	-0.0054	0.0131	-0.0331
	300	0.0398	-0.0046	-0.0056	0.0125	-0.0321
	400	0.0395	-0.0048	-0.0058	0.0147	-0.0345
	500	0.0401	-0.0044	-0.0054	0.0121	-0.0322
	600	0.0400	-0.0045	-0.0055	0.0115	-0.0315

Appendix B PhasorToolBox – A Python Package for Synchrophasor Application Prototyping

In this section, a new synchrophasor software development package using Python3 is proposed. The package collects and parses data from multiple synchrophasor devices using the protocol defined in the IEEE C37.118-2011 standard [52], and provides easy-to-use interfaces for development.

The Python programming language is infamous for its poor performance, specifically in that it requires more resources than C and C++. For example, similar implementations of C++ are much more efficient, which is a topic of discussion in this section. This proposed package overcomes the performance issue through the use of single-threaded asynchronous programming, a customized parser and a simplified configuration frame.

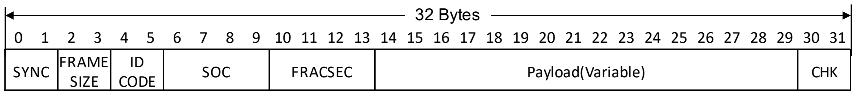


Figure 9: Synchrophasor message frame format. The size of payload varies depends on frame type and the configuration frame. Please note the unit is byte.

B.1 Related Work

The OpenPDC [7] is a synchrophasor application that functions as a PDC and has many features including interfaces for customized applications. Although OpenPDC is mostly implemented in C#, which is no an ideal languages for research and prototyping given the lack of scientific archival support and extensive development times[1].

Matlab [73] is one of the most commonly used programs in the power system community for research. It has many out-of-the-box tools for simulating a power system and capable of implementing new applications including neural networks. There is an open source repository on the GitHub supports parsing the synchrophasor protocol [79]. Despite these advantages, the regular updates to Matlab, with a new version published annually combined with poor backward compatibility requires that the user ensure compatibility prior to development.

[1]According to [7], the The language composition of OpenPDC is: C# 87.3%, SQLPL 7.6%, PLSQL 2.8%, JavaScript 1.5%, Batchfile 0.3%, CSS 0.2%, Other 0.3%

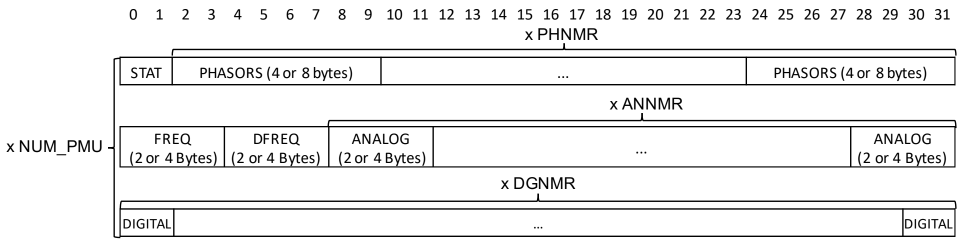


Figure 10: The payload format in a synchrophasor data frame. 'NUM_PMU ', 'PHNMR', 'ANNMR', and 'DGNMR' are obtained from the configuration frame. Please note the unit is byte.

One of the first few synchrophasor application toolkits created to help developers to implement synchrophasor applications in the LabView regime was created in 2013 [109]. Since that time, many researchers have used that toolkit to prototype new ideas. The same authors later created an enhanced version of the development toolkit, named BabelFish (BF) for the rapid prototyping of Wide Area Monitoring Protection and Control (WAMPAC) applications in LabView, C++, and Active X [8].

Packet sniffer packages in Python such as 'pyshark' [43] and 'scapy' [17] can parse synchrophasor messages. However, those packages do not provide methods to connect and demand data transmission from synchrophasor devices. Also, the performance of those packages are generally poor and sometimes not meet the requirement defined in the standard.

Rather than replace the OpenPDC or any other tools in different programming languages, the purpose of this proposed Python package ensure that diversity of languages so that the community may improve synchrophasor applications

B.2 Review of Synchrophasor Protocol

In this section, the synchrophasor message frame format and the communication methods through IP network of the synchrophasor protocol is briefly reviewed. Please refer to the standard for more details.

As of May 2018, the most recent version of the synchrophasor protocol is detailed in the IEEE C37.118.2-2011 standard, which while modifying the standard for measuring the synchrophasor keeps the synchrophasor communication protocol in place [5, 52]. In this standard, the synchrophasor, measurement method, synchrophasor message frame formats, communication methods, and their relative performance requirements are all defined. The C37.118.2-2011 standard superseded the C37.118-2005 standard [51] with a slight change in the definitions for some fields and added the configuration three frame. The C37.118.2-2011 standard is compatible with the C37.118-2015 standard.

B.2.1 Message Frames

Four message frame types are defined in the standard: data, configuration, header, and command. The data, configuration, and header frame types are transmitted from the data source and the command frame is transmitted to the data source. The general format of a synchrophasor message frame is shown in Fig. 9. Except for the variance of the format of payload varies on the frame type, all other fields share the same format, and except for the data frame, all other frames can be parsed based on the frame itself. There are three types of configuration frames, 'CFG-1', 'CFG-2', and 'CFG-3'. 'CFG-1' frames contain all available fields a synchrophasor device can provide, whereas 'CFG-2' frames contain currently available fields a synchrophasor device can provide. Although the 'CFG-3' frame need not be implemented on synchrophasor device, the 'CFG-2' frame must be implemented instead.

The format of the data frame is defined in the configuration frames, which contain information on the number and names of stations, and the number, names and format for all measurements. Thus, the configuration frame must be obtained by the receiver prior to parsing the data frames.

The format of the payload in a data frame is shown in Fig. 10, which contain measurements from one or multiple stations. The number of stations is defined in the 'NUM PMU' field in the configuration frame. In each station, 'PHNMR', 'ANNMR', and 'DGNMR' are defined in the related fields in the configuration frame. The configuration frame also determines the format of each field.

B.2.2 Communication Methods for IP

Although either serial or IP communication is used to apply the synchrophasor protocol, for purposes of this communication, the IP protocol is detailed here. There are four communication methods used: the 'TCP-only' method, the 'UDP-only' method, the 'TCP/UDP' method, and the 'Spontaneous Data Transmission' method. A 'server' is defined as the device providing data, and a 'client' is defined as the device receiving data.

In the 'TCP-only' method, a single TCP connection is used to transfer all frame types. The client must know the server's IP address and TCP port.

In the ‘UDP-only’ method, a single UDP channel is used to transfer all frame types. The client must know the server’s IP address and UDP port.

In the ‘TCP/UDP’ method, the UDP channel is used to transfer data frames, and the TCP connection is used to transfer all other frames. The client must know the server’s IP address and TCP port, and the server must know the client’s receiving port. The client’s address is optional if the UDP channel is using broadcast.

In the ‘Spontaneous Data Transmission’ method, the server sends data frames to a pre-configured destination and port continuously without listening to any incoming traffic.

B.2.3 Communication Sequence

Ten types of commands are defined in three criteria in the C37.118.2- 2011 standard: i) the ‘Turn off transmission of data frames’ ii) the ‘Turn on transmission of data frames’, and iii) the ‘Send CFG-2 frame’. The commands for each are the three commands to start, the request configuration frame, and the stop transmission respectively. The essential sequence to start and stop a data transmission session using ‘TCP-only’, ‘UDP-only’, or the ‘TCP/UDP’ communication method are shown in Fig. 11. In the ‘Spontaneous Data Transmission’ method, the data source keeps sending data frames. The configuration information may or may not be transferred using the ‘CFG-2’ frame in the same channel in a constant interval.

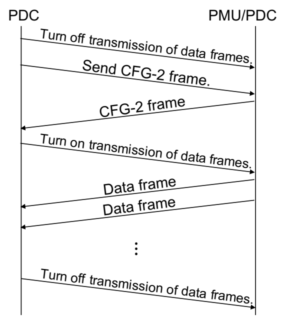

PDC
PMU/PDC
Turn off transmission of data frames.
Send CFG-2 frame.
CFG-2 frame
Turn on transmission of data frames.
Data frame
Data frame
Turn off transmission of data frames.

Figure 11: The communication sequence of a data transmission session using the ‘TCP-only’, ‘UDP-only’, and ’TCP/UDP’ communication methods.

B.2.4 Delays

Rather than providing a required time limit for message processing the C37.118.2-2011 standard instead provides an estimated delay for each link in the synchrophasor network. According to the standard, the PDC processing & alignment delay is in the range of 2ms to 2+s, and the communication system I/O delay is in the range of 0.05ms to 30ms.

B.3 PhasorToolBox

In this section, the modules of the proposed package are introduced, followed by an example configuration detailed at the end of this section.

The proposed package contains a parser module, a client module, and a PDC module, all of which must be instantiated before use. Before introducing each of the modules, the asynchronous programming technique is first introduced so that the client and PDC modules may communicate with multiple servers concurrently without any multi-threaded programming overhead.

All modules provide detailed documentation-‘help(module_name)’ for use in printing the detailed usage of each module.

B.3.1 Parser Module

The parser module is a stand-alone module that provides a ‘parse’ method that takes a binary string as input and returns a list of structured message frames. The returned list of structured message frame provides an easy-to-use interface to obtain the value of any desired fields.

For any synchrophasor message frame, the ‘frame type’ and ‘id code’ can be obtained without prior knowledge. Except for the data frame, the parser directly resolves the binary into the relative structured message frame.

As described in Section B.2, the parsing of synchrophasor data frame does require a priori knowledge of the configurations, which means that a piece of memory is used to store resolved configuration frames. If the ‘frame type’ is either ‘CFG-2’ or ‘CFG-3’ frame, then the resolved configuration frame is stored in the parser’s memory. If the ‘frame type’ is ‘data’, then the information in the resolved configuration frame with the same ‘id code’ is used to parse the data frame. If no configuration frame with the same ‘id code’ found in the memory, the data frame is discarded. The steps needed to parse a binary string and return the parsed frames are detailed in Fig. 12.

To observe the structure and the values of each field of a parsed message in the returned list, the ‘show()’ method is used on the message. Assume that a list of 100 messages ‘msgs’ is received. As shown in Fig. 13, part of the output of the ‘show()’ method applied on the 10th message by executing the helper method ‘msgs[9].show()’. This method is used by developers to determine the

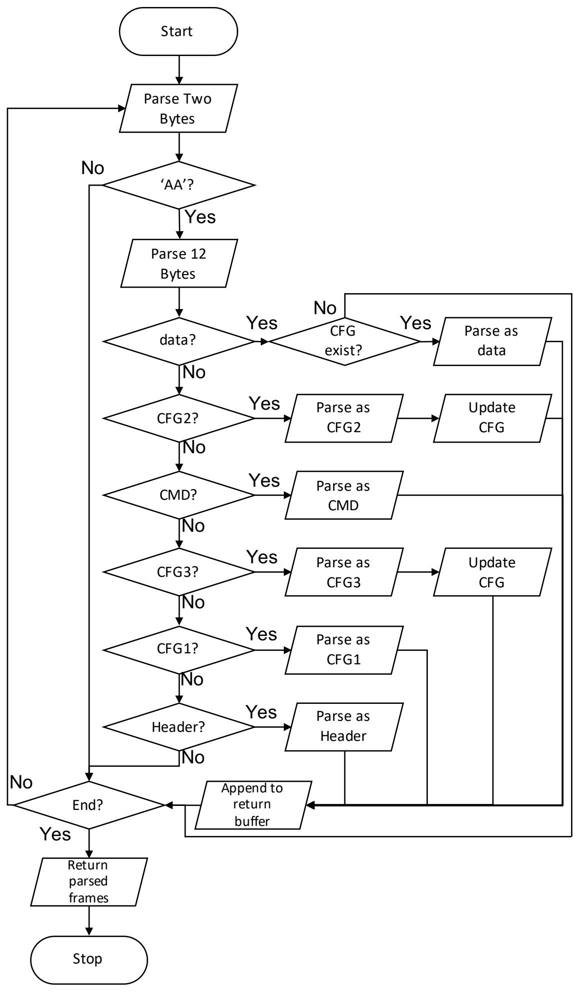


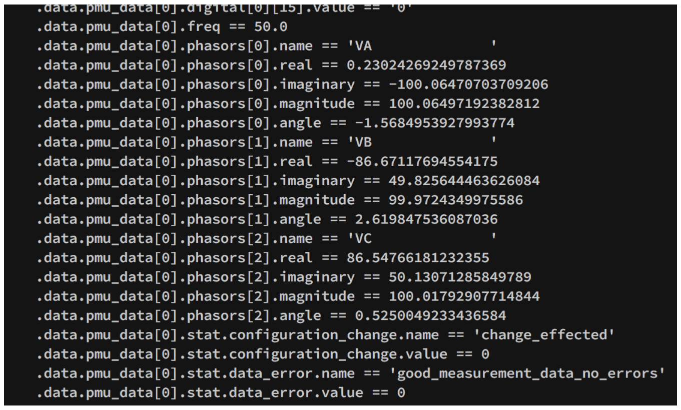
Figure 12: The flowchart of the parser module's parsing method.

proper procedure for the ready acquisition of the desired values. For example, to get the frequency value of the first PMU station in the 10th message, one can use 'msgs[9].data.pmu_data[0].freq'.

A feature of specific importance is the presentation of the phaser in the original data message in either 'rectangular' or 'polar' format, that varies with the configuration frames. In the parsed message, both presentation methods are available to the developer, and no further transformation is needed.

B.3.2 Client Module

The client module is also a stand-alone module that requires the server and device IP codes to function correctly. Different information is required depending upon the communication method,



```
    .data.pmu_data[0].digital[0][15].value == '0'
    .data.pmu_data[0].freq == 50.0
    .data.pmu_data[0].phasors[0].name == 'VA              '
    .data.pmu_data[0].phasors[0].real == 0.23024269249787369
    .data.pmu_data[0].phasors[0].imaginary == -100.06470703709206
    .data.pmu_data[0].phasors[0].magnitude == 100.06497192382812
    .data.pmu_data[0].phasors[0].angle == -1.5684953927993774
    .data.pmu_data[0].phasors[1].name == 'VB              '
    .data.pmu_data[0].phasors[1].real == -86.67117694554175
    .data.pmu_data[0].phasors[1].imaginary == 49.825644463626084
    .data.pmu_data[0].phasors[1].magnitude == 99.9724349975586
    .data.pmu_data[0].phasors[1].angle == 2.619847536087036
    .data.pmu_data[0].phasors[2].name == 'VC              '
    .data.pmu_data[0].phasors[2].real == 86.54766181232355
    .data.pmu_data[0].phasors[2].imaginary == 50.13071285849789
    .data.pmu_data[0].phasors[2].magnitude == 100.01792907714844
    .data.pmu_data[0].phasors[2].angle == 0.5250049233436584
    .data.pmu_data[0].stat.configuration_change.name == 'change_effected'
    .data.pmu_data[0].stat.configuration_change.value == 0
    .data.pmu_data[0].stat.data_error.name == 'good_measurement_data_no_errors'
    .data.pmu_data[0].stat.data_error.value == 0
```

Figure 13: The output of ‘show()’ method.

however. This proposed client module supports all four communication methods.

The client module also has the ‘run’ and ‘callback’ methods within. The ‘run’ method connects to a server to receive data and the ‘callback’ is an empty method that is executed upon acquisition of a data frame. A developer can implement any method and assign that customized function to the callback method. The user implemented function should take a parsed data frame as input. The ‘run’ method takes a positive integer as an optional parameter to stop the execution and close the connection after the number of time stamps(s) is(are) received. Without the parameter, the ‘run’ method will run until a key interrupt is initiated.

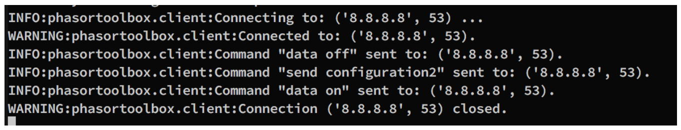
Here is an example usage to connect to a TCP server:

```python
from phasortoolbox import Client
pmu_client = Client(remote_ip='10.0.0.1',remote_port=4712, idcode=1, mode='TCP')
pmu_client.run(100)
```

:

After calling the ‘run’ method, once the client instance connected to the remote server, the client instance automatically initiates the communication sequence described in Section B.2 to start the data transmission. A parser instance embeds in the client instance continue with a parsing of the received binary messages described in Section B.3.1. Once the client receives and successfully parses a data frame, the callback method will be called with the argument of the parsed data frame.

By default, the callback method does not do anything. However, when the instance by ‘run’ method is initiated, the instance will attempt to connect the remote server. Although only critical messages are displayed with the default log level, user can change the log level to ‘DEBUG’, which will provide more information for purposes of checking network availability. The output of a successful but immediately closed connection with no data frame received is detailed in Fig 14. Here, although the device is available and reachable through the network, either the 'id_code' is incorrect or the device is not a synchrophasor device. The following script can be used to change the logging



```
INFO:phasortoolbox.client:Connecting to: ('8.8.8.8', 53) ...
WARNING:phasortoolbox.client:Connected to: ('8.8.8.8', 53).
INFO:phasortoolbox.client:Command "data off" sent to: ('8.8.8.8', 53).
INFO:phasortoolbox.client:Command "send configuration2" sent to: ('8.8.8.8', 53).
INFO:phasortoolbox.client:Command "data on" sent to: ('8.8.8.8', 53).
WARNING:phasortoolbox.client:Connection ('8.8.8.8', 53) closed.
```

Figure 14: The logs while trying to connect a non synchrophasor device.

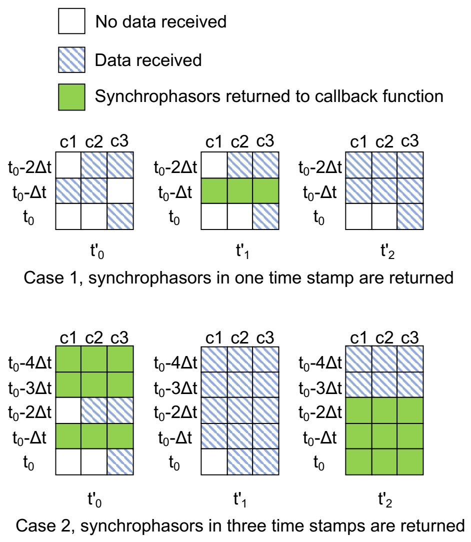
level:

```python
import logging
logging.getLogger().setLevel(logging.DEBUG)
```

:

The coroutine methods 'coro_run' and 'coro_close' methods are provided for asynchronous programming.

B.3.3 PDC Module



No data received
Data received
Synchrophasors returned to callback function
c1 c2 c3
t_0-2Δt
t_0-Δt
t_0
t'_0
t'_1
t'_2
Case 1, synchrophasors in one time stamp are returned
t_0-4Δt
t_0-3Δt
Case 2, synchrophasors in three time stamps are returned

Figure 15: An illustration of the 'buffer' status change under different configurations. 't' presents the GPS time stamp of the measurement data. 'Δt' presents the timing interval between two synchrophasor measurements. 't´ ' presents the clock time. 'c' presents the client instances.

The PDC module must work with client modules, specifically by gathering parsed frames from the client instances, aligning the frames using the GPS time stamps and returning the aligned synchrophasors.

A ‘run’ method is also within the PDC module, which is a coroutine wrapper. Upon execution of the run method, all clients assigned to the PDC instance connect to their assigned server for data retrieval

Similar to the Client module, the ‘run’ method takes a positive integer as an optional parameter to halt the execution and close all connections after receiving the number of time stamps. Without the parameter, the ‘run’ method will continue until a key interrupt.

Here is an example of an assigned series of client instances to a single PDC instance:

```
from phasortoolbox import Client, PDC
pmu_client1 = Client(remote_ip='10.0.0.1',remote_port=4712, idcode=1, mode='TCP')
pmu_client2 = Client(remote_ip='10.0.0.2',remote_port=4713, local_port=4713, idcode=2, mode='UDP')
pdc = PDC(clients=[pmu_client1, pmu_client2])
pdc.callback = lambda synchrophasors: print(synchrophasors[-1].time)
pdc.run()
```

:

Similar to the client module, a ‘callback’ method initiates when some synchrophasors are ready. For some computations such as the use of multiple time stamps of synchrophasors to infer a near future measurement, multiple time stamps of synchrophasors can be returned simultaneously. This option can be changed by assigning a positive integer value to the ‘history’ option of the PDC instance to the desired number of time stamps needed in each callback.

The ‘return_on_time_out’ is a boolean flag that determines if a time stamp with not all synchrophasors received should be returned upon time out. By default, only time stamps with all synchrophasors received will be returned.

The returned time stamps are always in sequence with the last returned time stamps never returned out of sequence. For example, if some synchrophasors with a older time stamp are created after synchrophasors with a newer stamp are already sent, caused by possible network delays, the synchrophasors with older time stamp that was just created will be held for a later return. This step by step process is detailed in Fig. 15. In case 1, where the PDC is configured to return time stamps in sequence, the synchrophasor with an older time stamp is not returned because a synchrophasor with more recent stamp was returned earlier. In case 2, where the PDC is configured to return three time stamps simultaneously, the older time stamp is returned if synchrophasors with a more recent time tag is also created.

The 'coro_run' and 'coro_close' coroutines are also provided for asynchronous programming.

B.3.4 The Callback Methods

The proposed development package provides a callback based interface in which the callback method in the Client module and the PDC play key roles to fully utilize this proposed software package fully. For a Client instance, the callback uses a single data frame as input, and for a PDC instance, the callback takes a list of synchrophasors as input.

To solve complexed problems, a developer must define the required memory space and parameters in another object, and implement a method for the object that uses as input that measurement data to solve the problem. The method is then assigned to either a PDC or CLient instance before running.

Please refer to the PhasorToolBox code repository for tutorials and examples of specific uses.

B.3.5 An Example Architecture

An example synchrophasor network built with all three modules is detailed in Figure Fig. 16.

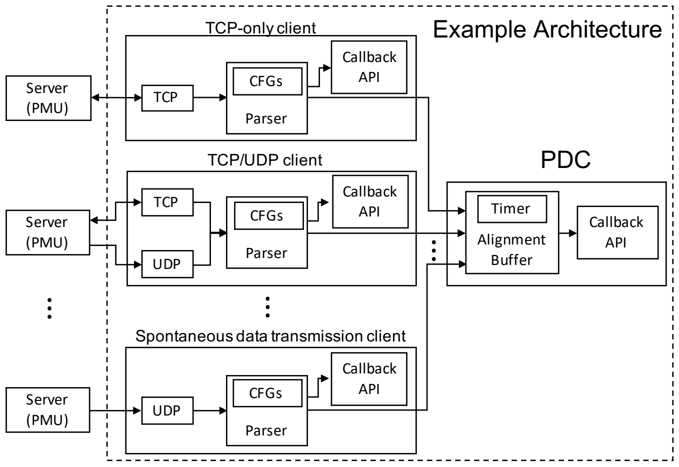


Figure 16: An illustration of a synchrophasor network built with the proposed package.

In the example setup, multiple remote servers are connected to the same number of client instances that use one of the communication types concurrently. Depending upon the connection method, the client module automates the connection and initializes a data transmission session. The received binary messages are sent to the parser instance in the client instance. Then, the client instances concurrently sends the parsed message to a PDC. The PDC aligns the received message and then calls another callback function.

B.3.6 A Real-time Frequency Meter

The program presented in this section is an example implementation using the proposed package. The program implemented a frequency meter that shows 300 historical frequency measurements of two PMUs in real time. Fig. 17 shows the screen shot of the program interface while running.

```
from phasortoolbox import PDC,Client
import matplotlib.pyplot as plt
import numpy as np
import logging
logging.basicConfig(level=logging.DEBUG)

class FreqMeter(object):
    def __init__(self):
        x = np.linspace(-10.0, 0.0, num=300, endpoint=False)
        y = [60.0]*300

        plt.ion()
        self.fig = plt.figure()

        ax = self.fig.add_subplot(211)
        self.line1, = ax.plot(x, y)
        plt.title('PMU1 Frequency Plot')
        plt.xlabel('Time (s)')
        plt.ylabel('Freq (Hz)')

        ax = self.fig.add_subplot(212)
        self.line2, = ax.plot(x, y)
        plt.title('PMU2 Frequency Plot')
        plt.xlabel('Time (s)')
        plt.ylabel('Freq (Hz)')

        plt.tight_layout()

    def update_plot(self, synchrophasors):
        y_data = [[],[]]

        for synchrophasor in synchrophasors:
            for i, msg in enumerate(synchrophasor):
                y_data[i].append(msg.data.pmu_data[0].freq)

        self.line1.set_ydata(y_data[0])
        self.line2.set_ydata(y_data[1])

        self.fig.canvas.draw()
        self.fig.canvas.flush_events()

if __name__ == '__main__':
    pmu_client1 = Client(remote_ip='10.0.0.1',remote_port=4712, idcode=1, mode='TCP')
    pmu_client2 = Client(remote_ip='10.0.0.2',remote_port=4713, local_port=4713, idcode=2, mode='UDP')

    fm = FreqMeter()
    pdc = PDC(clients=[pmu_client1,pmu_client2],history=300)
    pdc.callback = fm.update_plot

    pdc.run()
```

:

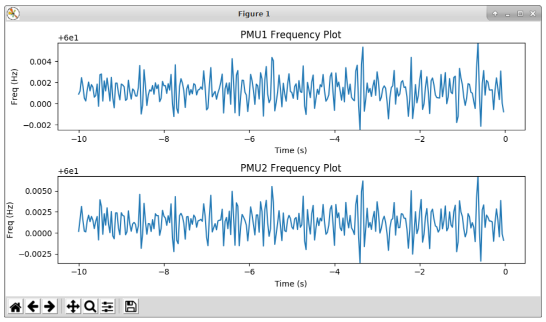


Figure 17: A screen shot of the real-time frequency meter example presented in Section B.3.6

B.4 Performance Considerations

To develop this proposed concept, the authors chose Python, which is well known for its short development time, rich package library, and most commonly used languages for research development. Despite the deficiencies in performance, the short development time compensates for the computing time, which was not a problem for general research. Given that most synchrophasor applications are usually time critical, three essential techniques were embedded in the proposed package to improve performance: single-threaded asynchronous programming, a customized binary parser, and a simplified configuration frame.

B.4.1 Single-Threaded Asynchronous Programming

Asynchronous execution refers to executing multiple tasks that are independent of each other. One task can be started before another task is ended. Synchronous execution refers to executing multiple tasks that have dependencies. Tasks need to be executed in a specific order and not overlap each other. For a program that can communicate with multiple remote nodes concurrently entails using asynchronous programming because the network packets are not guaranteed to be delivered in order.

The traditional method for implementing asynchronous programming is using either multi-threaded or multi-process programming on a multiple core computer. Multi-process programming creates multiple processes, each with a different virtual memory space that can undertake simultaneous computing on multiple CPU cores. Such multi-threaded programming creates multiple threads sharing the same virtual memory. Here, the operating system schedules the execution of each thread or process with any switches between thread or processes causing overhead in the form of context switches. Both methods work well for a computation-heavy program because the simultaneous computation using multiple CPU cores can reduce the completion time if the program is written correctly.

However, when using a physical interface for gathering data from multiple synchrophasor devices, the computational load is light. Although the system input/output(I/O), such as the write/read to disk, or wait for a network packet to receive, is quite time-consuming, it requires a minimum of computation. Indeed the time required for such calculations is much less than that required for packages to receive and switch between threads or processes when multi-threading or multi-processing is used.

The proposed package uses a single thread to implement the asynchronous programming pattern. Rather than halting the CPU and waiting for the I/O task to finish, single-threaded asynchronous programming releases the CPU to work on the next scheduled task while waiting for the physical interface to finish the I/O task, and return to the unfinished I/O task at a later time. This concept is similar to the creation of multiple threads, each of which is assigned a small and easy task without the context switch overhead while using multi-threading or multi-processing.

However, single-threaded synchronous programming is not as straightforward as multi-threaded programming. The switch between tasks needs to be manually assigned. In Python, the functions that are executed asynchronously are known as a 'coroutine', which is scheduled using an 'event loop.' The proposed package provides both wrapper API that acts like standard protocols for easier programming and coroutine API for advanced implementation.

In the early development stage of the proposed package, the switch from multi-threading to single-threaded asynchronous programming reduced the average latency of processing one synchrophasor message by 90%.

B.4.2 Kaitai Struct

Kaitai Struct is a declarative language used for describing various binary data structures, and the proposed package's Parser module was developed with this language [?]. Although other parsers are available that can parse the synchrophasor messages, most have a less than ideal performance, hence our rationale for developing this proposed package. Indeed, the substitution of the binary parsing methods within Python to the Kaitai Struct defined-parser reduced the parsing time by 90%.

B.4.3 Minimized Configure Frame

Minimized Configure Frame is a customized data structure that uses the minimum required information in a configuration frame for later parsing of a data frame. Three options are available for managing the dynamically defined data frame packet structure: a) an 'on the fly' program recompile; b) an attachment of the configuration frame binary in front of every data frame; c) and the use of the variable in the parsed configuration frame to parse the data frames. Although Option A provides the best performance, it is characterized by poor code readability and reusability. Option

B, however, while providing the best code readability exhibits the poorest performance in that the same configuration frame is parsed every time a data frame is received. Finally, Option C, while striking a good balance between both Options A and B, is hindered by the latency of the inquiring values from the massive structure of a configuration frame. Thus, in the proposed software package, a minimized data structure with only the necessary information and minimum depth is created and used as an input to parse data frames. This change from the original configuration frame to a minimized configuration frame reduced the average parsing time reduced by 30%.

B.4.4 Performance Test

A performance test program is implemented using the proposed software package. The test was performed in the Real-Time Power and Intelligent Systems (RTPIS) Lab [92] at Clemson University. The subject was a computer with Intel(R) Xeon(R) E5-1620 v2 @ 3.70GHz CPU, 8GB Ram, 500GB HDD and 1 Gigabit Ethernet port.

The latency to synchronize all synchrophasors is measured, which is considered as the time difference between the last data frame received until the user-defined callback function begins, as shown in Fig. 18.

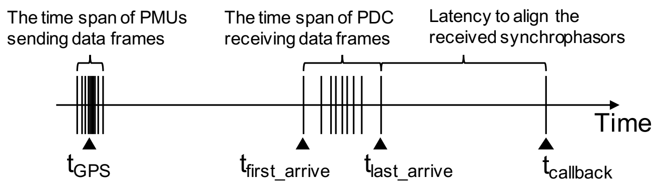

The time span of PMUs sending data frames
The time span of PDC receiving data frames
Latency to align the received synchrophasors
Time
t_{GPS}
t_{first_arrive}
t_{last_arrive}
$t_{callback}$

Figure 18: The timeline of events related to one synchronized synchrophasor measurements. The 'Latency to synchronize the received synchrophasors ' is considered as the time difference between the last data frame received until the user-defined callback function is started.

Table 5: The latency test results

Communication Method	Mean (ms)	Standard Deviation(ms)	Min (ms)	Max (ms)
TCP/UDP	0.3208	0.0804	0.1411	0.9198
UDP Spontaneous	0.3078	0.0649	0.1345	0.7950
TCP	0.3277	0.1324	0.1415	25.2840

The latency time series of the tests are shown in Figs. 19, 20, and 21 respectively. The

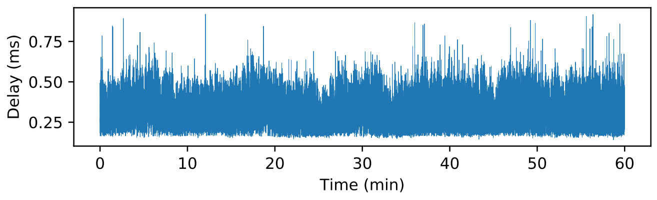


Figure 19: The time series of latencies caused by the proposed package. In this test, 15 PMUs are connected, one hour of data is collected. All PMUs are using TCP/UDP communication method.

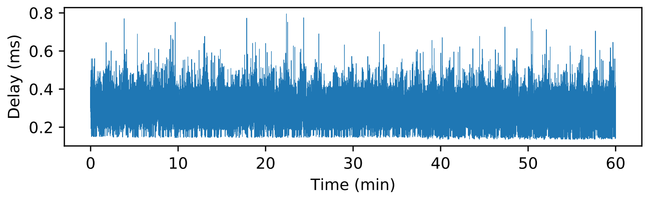


Figure 20: The time series of latencies caused by the proposed package. In this test, 15 PMUs are connected, one hour of data is collected. All PMUs are using UDP Spontaneous Data Transmission communication method.

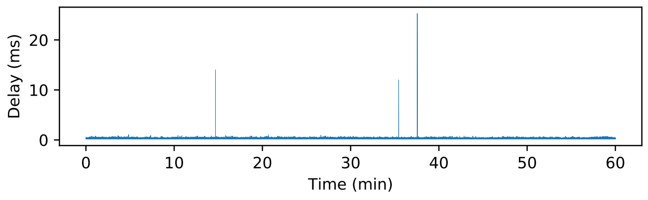


Figure 21: The time series of latencies caused by the proposed package. In this test, 15 PMUs are connected, one hour of data is collected. All PMUs are using TCP communication method.

mean, standard deviation, minimum delay, and maximum delay over the one hour test are calculated and shown in Table 5.

Three tests are conducted. Each test uses a different communication method. In each test, 15 PMUs are connected using the Client module. Data are collected and synchronized using the PDC module. One hour of data is collected.

Although the average of the delay of the proposed Python package meets the IEEE Std C37.118 requirement after applying our techniques, this package is intended for prototyping and not for critical applications in production. As shown in Figs. 19, 20, and 21, the delays are changing over time, and there are delays over 2ms occasionally. Thus, user needs to pay extra attention when using the proposed package for timing sensitive application implementation. Also, due to the nature of the Python language, there are many behaviors could surprise a less experienced programmer, such as underflow problems.

B.5 Conclusion

The PhasorToolBox simplifies all required efforts to connect to remote synchrophasor devices, acquire data, parse data and align the data frames in minimum delay and complexity. More importantly, the PhasorToolBox provides a very easy-to-use interface for development, and overcomes the performance issue of the Python programming language so that developers may quickly prototype new ideas that utilize Python's full potential. For the source code of the proposed package and more examples, please refer to [125].

Appendix C Game Theory Analysis Payoff Matrices

Table 6: Use Zero for AGC

	Attack/Countermeasure PMUs	Mean	Variance	Confidence Interval
No Attack	All PMUs	105.91	355238.25	120.11
Drop PMUs	PMU_8	143818.63	28654.14	34.11
Drop PMUs	Primary PMUs	1581714.13	2661952.09	328.79
Drop PMUs	Primary and Secondary PMUs	3595413.76	8932540.53	602.28
Drop PMUs	Primary Secondary and Tertiary PMUs	5609022.12	21081720.96	925.27
Drop PMUs	All PMUs	8341974.82	43433784.32	1328.09
BGP Hijacking	PMU_8	143681.50	56065.04	47.72
BGP Hijacking	Primary PMUs	1566895.99	259496534.78	3246.23
BGP Hijacking	Primary and Secondary PMUs	3578841.06	345333562.32	3744.84
BGP Hijacking	Primary Secondary and Tertiary PMUs	5590011.64	463871542.63	4340.23
BGP Hijacking	All PMUs	8319940.70	643699682.04	5112.76
Random Delay	PMU_8	107604.23	1688223.92	261.84
Random Delay	Primary PMUs	1172045.66	433445273.68	4195.48
Random Delay	Primary and Secondary PMUs	2683735.88	1154281958.74	6846.52
Random Delay	Primary Secondary and Tertiary PMUs	4195095.35	2349700630.99	9768.33
Random Delay	All PMUs	6237060.23	6393221673.95	16112.91

Table 7: Use Sixty for AGC

	Attack/Countermeasure PMUs	Mean	Variance	Confidence Interval
No Attack	All PMUs	0.00	0.00	0.01
Drop PMUs	PMU_8	14.81	11.22	0.68
Drop PMUs	Primary PMUs	165.66	1451.41	7.68
Drop PMUs	Primary and Secondary PMUs	379.88	7580.86	17.55
Drop PMUs	Primary Secondary and Tertiary PMUs	587.77	17863.49	26.93
Drop PMUs	All PMUs	910.14	44347.35	42.44
BGP Hijacking	PMU_8	14.80	11.22	0.67
BGP Hijacking	Primary PMUs	164.32	1487.23	7.77
BGP Hijacking	Primary and Secondary PMUs	378.36	7673.94	17.65
BGP Hijacking	Primary Secondary and Tertiary PMUs	586.01	18001.55	27.04
BGP Hijacking	All PMUs	908.09	44664.44	42.59
Random Delay	PMU_8	11.09	6.22	0.50
Random Delay	Primary PMUs	123.10	837.15	5.83
Random Delay	Primary and Secondary PMUs	284.08	4419.51	13.40
Random Delay	Primary Secondary and Tertiary PMUs	439.02	10176.69	20.33
Random Delay	All PMUs	680.63	24795.84	31.73

Table 8: Use The Last Available Data for AGC

	Attack/Countermeasure PMUs	Mean	Variance	Confidence Interval
No Attack	All PMUs	0.01	0.00	0.01
Drop PMUs	PMU_8	12.00	107.73	2.09
Drop PMUs	Primary PMUs	134.12	12455.54	22.49
Drop PMUs	Primary and Secondary PMUs	313.00	68904.99	52.90
Drop PMUs	Primary Secondary and Tertiary PMUs	484.22	165054.78	81.87
Drop PMUs	All PMUs	788.55	399174.67	127.32
BGP Hijacking	PMU_8	3.93	10.87	0.66
BGP Hijacking	Primary PMUs	46.00	1409.27	7.57
BGP Hijacking	Primary and Secondary PMUs	120.23	9815.27	19.96
BGP Hijacking	Primary Secondary and Tertiary PMUs	169.77	18505.40	27.41
BGP Hijacking	All PMUs	326.97	56712.49	47.99
Random Delay	PMU_8	3.04	6.58	0.52
Random Delay	Primary PMUs	35.11	835.90	5.83
Random Delay	Primary and Secondary PMUs	92.02	5841.47	15.40
Random Delay	Primary Secondary and Tertiary PMUs	130.41	11194.89	21.32
Random Delay	All PMUs	249.33	33344.08	36.80

Table 9: Use SCADA for AGC

	Attack/Countermeasure PMUs	Mean	Variance	Confidence Interval
No Attack	All PMUs	0.02	0.02	0.02
Drop PMUs	PMU_8	7.09	31.45	1.13
Drop PMUs	Primary PMUs	75.79	3675.06	12.22
Drop PMUs	Primary and Secondary PMUs	189.45	22708.65	30.37
Drop PMUs	Primary Secondary and Tertiary PMUs	288.55	50991.74	45.51
Drop PMUs	All PMUs	503.88	137241.01	74.65
BGP Hijacking	PMU_8	7.08	31.44	1.13
BGP Hijacking	Primary PMUs	75.72	3672.25	12.21
BGP Hijacking	Primary and Secondary PMUs	189.34	22698.51	30.36
BGP Hijacking	Primary Secondary and Tertiary PMUs	288.39	50967.60	45.49
BGP Hijacking	All PMUs	503.66	137189.76	74.64
Random Delay	PMU_8	5.31	17.71	0.85
Random Delay	Primary PMUs	56.83	2080.11	9.19
Random Delay	Primary and Secondary PMUs	142.50	12832.20	22.83
Random Delay	Primary Secondary and Tertiary PMUs	216.54	28769.45	34.18
Random Delay	All PMUs	377.68	77328.36	56.04

Table 10: Use VSN for AGC

	Attack/Countermeasure PMUs	Mean	Variance	Confidence Interval
No Attack	All PMUs	0.03	0.03	0.04
Drop PMUs	PMU_8	3.66	9.12	0.61
Drop PMUs	Primary PMUs	44.15	1166.56	6.88
Drop PMUs	Primary and Secondary PMUs	140.92	9996.10	20.15
Drop PMUs	Primary Secondary and Tertiary PMUs	310.91	31419.84	35.72
Drop PMUs	All PMUs	1721.53	86590.37	59.30
BGP Hijacking	PMU_8	3.65	9.12	0.61
BGP Hijacking	Primary PMUs	44.10	1167.40	6.89
BGP Hijacking	Primary and Secondary PMUs	140.81	10000.40	20.15
BGP Hijacking	Primary Secondary and Tertiary PMUs	310.63	31433.89	35.73
BGP Hijacking	All PMUs	1717.11	85216.61	58.83
Random Delay	PMU_8	2.74	5.12	0.46
Random Delay	Primary PMUs	33.06	652.41	5.15
Random Delay	Primary and Secondary PMUs	105.61	5607.97	15.09
Random Delay	Primary Secondary and Tertiary PMUs	233.22	17664.14	26.78
Random Delay	All PMUs	1287.54	48676.64	44.46

Table 11: Drop All PMUs

Countermeasures	Mean	Variance	Confidence Interval
Use Zero for AGC	8341974.82	43433784.32	1328.09
Use Sixty for AGC	910.14	44347.35	42.44
Use the Last Received Value for AGC	788.55	399174.67	127.32
Use SCADA for AGC	503.88	137241.01	74.65
Use VSN for AGC	1721.53	86590.37	59.30

Table 12: Drop Primary and Secondary PMUs

Countermeasures	Mean	Variance	Confidence Interval
Use Zero for AGC	3595413.76	8932540.53	602.28
Use Sixty for AGC	379.88	7580.86	17.55
Use the Last Received Value for AGC	313.00	68904.99	52.90
Use SCADA for AGC	189.45	22708.65	30.37
Use VSN for AGC	140.92	9996.10	20.15

Bibliography

[1] Bothunter: A network-based botnet diagnosis system. `http://www.bothunter.net/`. [Last visited: 06-Aug-2015].

[2] IEC TC57 WG15:IEC 62351 Security Standards for the Power System Information Infrastructure. Technical report, International Electrotechnical Commission.

[3] Tor. `https://www.torproject.org/`. [Last visited: 06-Aug-2015].

[4] Factors affecting pmu installation costs. Technical report, Department of Energy, October 2014.

[5] IEEE standard for synchrophasor measurements for power systems – amendment 1: Modification of selected performance requirements. *IEEE Std C37.118.1a-2014 (Amendment to IEEE Std C37.118.1-2011)*, pages 1–25, April 2014.

[6] Synchrophasor technology and renewables integration: Naspi technical workshop. Technical report, Department of Energy, June 2014.

[7] G. P. Alliance. openpdc. `http://openpdc.codeplex.com`. [Last visited: 06-Aug-2015].

[8] M. Almas, L. Vanfretti, and M. Baudette. Babelfish—tools for ieee c37.118.2-compliant real-time synchrophasor data mediation. *SoftwareX*, 6:209 – 216, 2017.

[9] A. Apvrille. Cryptography for mobile malware obfuscation. In *RSA Conference Europe*, 2011.

[10] A. Arvani and V. S. Rao. Cyber security of smart grid systems using intrusion detection methods. In *The International Conference on Computer Security and Digital Investigation (ComSec2014)*, pages 21–28. The Society of Digital Information and Wireless Communication, 2014.

[11] T. Baumeister. Literature review on smart grid cyber security. 2010.

[12] T. Baumeister and Y. Dong. Towards secure identity management for the smart grid. *Security and Communication Networks*, 9(9):808–822, 2016.

[13] C. Beasley, G. Venayagamoorthy, and R. Brooks. Cyber Security Evaluation of Synchrophasors in a Power System. In *Power Systems Conference (PSC), 2014 Clemson University*, pages 1–5, March 2014.

[14] C. Beasley, X. Zhong, J. Deng, R. Brooks, and G. Kumar Venayagamoorthy. A Survey of Electric Power Synchrophasor Network Cyber Security. In *Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), 2014 IEEE PES*, pages 1–5, Oct 2014.

[15] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *Proceedings of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, pages 43–48, 2006.

[16] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 31–38. IEEE, 2010.

[17] P. Biondi. Scapy is a powerful python-based interactive packet manipulation program and library. `https://scapy.readthedocs.io/en/latest/introduction.html`. [Last visited: 27-Jul-2018].

[18] B. Blevins. Financial malware focuses on hiding malicious traffic, localization. `http://searchsecurity.techtarget.com/news/2240212531/Financial-malware-focuses-on-hiding-malicious-traffic-localization`, 2014. [Last visited: 06-Aug-2015].

[19] E. Bou-Harb, C. Fachkha, M. Pourzandi, M. Debbabi, and C. Assi. Communication security for smart grid distribution networks. *IEEE Communications Magazine*, 51(1):42–49, January 2013.

[20] W. F. Boyer and S. A. McBride. Study of security attributes of smart grid systems–current cyber security issues. *Idaho National Laboratory, USDOE, Under Contract DE-AC07-05ID14517*, 2009.

[21] R. R. Brooks. *Introduction to Computer and Network Security: Navigating Shades of Gray.* Chapman & Hall/CRC, 2013.

[22] R. R. Brooks, J. M. Schwier, and C. Griffin. Behavior detection using confidence intervals of hidden markov models. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(6):1484–1492, 2009.

[23] R. R. Brooks, L. Yu, Y. Fu, G. Cordone, J. Oakley, and X. Zhong. Using markov models and statistics to learn, extract, fuse, and detect patterns in raw data. In N. S. Rao, R. R. Brooks, and C. Q. Wu, editors, *Proceedings of International Symposium on Sensor Networks, Systems and Security*, pages 265–283, Cham, 2018. Springer International Publishing.

[24] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security Privacy*, 7(1):78–81, Jan 2009.

[25] G. Carl, R. R. Brooks, and S. Rai. Wavelet based denial-of-service detection. *Comput. Secur.*, 25(8):600–615, Nov. 2006.

[26] L. Chen, J. M. Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin. A normalized statistical metric space for hidden markov models. 43(3):806 – 819, 2013.

[27] D.-E. Cho, S.-S. Yeo, and S.-J. Kim. Authentication method for privacy protection in smart grid environment. *Journal of Applied Mathematics*, 2014, 2014.

[28] G. Combs. Wireshark. `https://www.wireshark.org`. [Last visited: 06-Aug-2015].

[29] A. H. Copeland et al. John von neumann and oskar morgenstern, theory of games and economic behavior. *Bulletin of the American Mathematical Society*, 51(7):498–504, 1945.

[30] R. Craven. Traffic analysis of anonymity systems. Master's thesis, Clemson University, 2010.

[31] S. D'Antonio, L. Coppolino, I. A. Elia, and V. Formicola. Security issues of a phasor data concentrator for smart grid infrastructure. In *Proceedings of the 13th European Workshop on Dependable Computing*, EWDC '11, pages 3–8, New York, NY, USA, 2011. ACM.

[32] M. Dietz, A. Czeskis, D. Balfanz, and D. S. Wallach. Origin-bound certificates: A fresh approach to strong client authentication for the web. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 317–331, Bellevue, WA, 2012. USENIX.

[33] K. P. Dyer. fteproxy. `https://fteproxy.org`, 2013. [Last visited: 06-Aug-2015].

[34] K. P. Dyer. LibFTE 0.1.0. `https://github.com/kpdyer/libfte`, 2015. [Last visited: 06-Aug-2015].

[35] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 61–72, New York, NY, USA, 2013. ACM.

[36] D. Fifield, N. Hardison, J. Ellithorpe, E. Stark, D. Boneh, R. Dingledine, and P. Porras. Evading censorship with browser-based proxies. In *Privacy Enhancing Technologies*, pages 239–258. Springer, 2012.

[37] C. Foundation. Application layer packet classifier for linux: L7-filter. `http://l7-filter.sourceforge.net/`, 2009. [Last visited: 06-Aug-2015].

[38] Y. Fu, Z. Jia, L. Yu, X. Zhong, and R. Brooks. A covert data transport protocol. In *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 1–8, Oct 2016.

[39] S. Ghosh and M. H. Ali. Minimization of adverse effects of time delay in smart power grid. In *ISGT 2014*, pages 1–5, Feb 2014.

[40] M. Goncharov. Trend micro incorporated research paper 2012 - russian underground 101. `http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-russian-underground-101.pdf`, 2011. [Last visited: 06-Aug-2015].

[41] S. Gong, Z. Zhang, M. Trinkle, A. D. Dimitrovski, and H. Li. Gps spoofing based time stamp attack on real time wide area monitoring in smart grid. In *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 300–305, Nov 2012.

[42] D. Goodin. Buggy smart meters open door to power-grid botnet. `http://www.theregister.co.uk/2009/06/12/smart_grid_security_risks/`, June 2009. [Last visited: 06-Aug-2015].

[43] D. Green. Python wrapper for tshark, allowing python packet parsing using wireshark dissectors. `https://github.com/KimiNewt/pyshark`. [Last visited: 27-Jul-2018].

[44] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, volume 5, pages 139–154, 2008.

[45] W. G. J. Halfond and A. Orso. Amnesia: Analysis and monitoring for neutralizing sql-injection attacks. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, ASE '05, pages 174–183, New York, NY, USA, 2005. ACM.

[46] B. Harakrishnan, S. Jason, C. Ryan, B. Richard R, H. Kathryn, G. Daniele, and G. Christopher. Side-Channel Analysis for Detecting Protocol Tunneling. *Advances in Internet of Things*, 1(2):13–26, 2011.

[47] D. He, S. Chan, Y. Zhang, M. Guizani, C. Chen, and J. Bu. An enhanced public key infrastructure to secure smart grid wireless communication networks. *IEEE Network*, 28(1):10–16, January 2014.

[48] R. Hewett, S. Rudrapattana, and P. Kijsanayothin. Cyber-security analysis of smart grid scada systems with game models. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, CISR '14, pages 109–112, New York, NY, USA, 2014. ACM.

[49] R. Housley. Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP). RFC 3686, RFC Editor, January 2004.

[50] W. Hu, J. Hiser, D. Williams, A. Filipi, J. W. Davidson, D. Evans, J. C. Knight, A. Nguyen-Tuong, and J. Rowanhill. Secure and practical defense against code-injection attacks using software dynamic translation. In *Proceedings of the 2Nd International Conference on Virtual Execution Environments*, VEE '06, pages 2–12, New York, NY, USA, 2006. ACM.

[51] IEEE. IEEE standard for synchrophasors for power systems. *IEEE Std C37.118-2005 (Revision of IEEE Std 1344-1995)*, pages 0_1–57, 2006.

[52] IEEE. C37.118.2-2011 - IEEE standard for synchrophasor data transfer for power systems. pages 1–53, Dec 2011.

[53] I. Jayawardene and G. K. Venayagamoorthy. Cellular computational extreme learning machine network based frequency predictions in a power system. In *2017 International Joint Conference on Neural Networks (IJCNN)*.

[54] I. Jayawardene and G. K. Venayagamoorthy. Reservoir based learning network for control of two-area power system with variable renewable generation. *Neurocomputing*, 170:428 – 438, 2015.

[55] J. Johnston and J. DiNardo. *Econometric methods*, volume 2. New York, 1972.

[56] A. Keenan, R. Schweller, M. Sherman, and X. Zhong. Fast arithmetic in algorithmic self-assembly. *Natural Computing*, 15(1):115–128, Mar 2016.

[57] A. Keenan, R. Schweller, and X. Zhong. Exponential replication of patterns in the signal tile assembly model. In D. Soloveichik and B. Yurke, editors, *DNA Computing and Molecular Programming*, pages 118–132, Cham, 2013. Springer International Publishing.

[58] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, RFC Editor, November 1998.

[59] P. Kocher, J. Jaffe, and B. Jun. *Differential Power Analysis*, pages 388–397. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[60] K. F. Krommydas and A. T. Alexandridis. Modular control design and stability analysis of isolated pv-source/battery-storage distributed generation systems. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(3):372–382, Sept 2015.

[61] F. Lau, S. H. Rubin, M. H. Smith, and L. Trajkovic. Distributed denial of service attacks. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 2275–2280 vol.3, 2000.

[62] W. B. Leea, T. H. Chen, W. R. Sun, and K. I. J. Ho. An s/key-like one-time password authentication scheme using smart cards for smart meter. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 281–286, May 2014.

[63] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili. A study of communication and power system infrastructure interdependence on pmu-based wide area monitoring and protection. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–7, July 2012.

[64] S. Liu, B. Chen, T. Zourntos, D. Kundur, and K. Butler-Purry. A coordinated multi-switch attack for cascading failures in smart grid. *IEEE Transactions on Smart Grid*, 5(3):1183–1195, May 2014.

[65] C. Lu. Network Traffic Analysis Using Stochastic Grammars. PhD Dissertation, Dept. of Electrical and Computer Engineering, Clemson University, 2012.

[66] C. Lu and R. Brooks. Botnet traffic detection using hidden markov models. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '11, pages 31:1–31:1, New York, NY, USA, 2011. ACM.

[67] L. Y. Lu and C. C. Chu. Consensus-based secondary frequency and voltage droop control of virtual synchronous generators for isolated ac micro-grids. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(3):443–455, Sept 2015.

[68] B. Luitel and G. Venayagamoorthy. Decentralized asynchronous learning in cellular neural networks. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(11):1755–1766, Nov 2012.

[69] B. Luitel and G. K. Venayagamoorthy. Cellular computational networks – a scalable architecture for learning the dynamics of large networked systems. *Neural Networks*, 50:120–123, 2014.

[70] K. Mandal and S. Banerjee. Synchronization phenomena in microgrids with capacitive coupling. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(3):364–371, Sept 2015.

[71] D. Manky. Zeus: God of diy botnets. `http://www.fortiguard.com/legacy/analysis/zeusanalysis.html`, 2009. [Last visited: 06-Aug-2015].

[72] K. E. Martin, D. Hamai, M. G. Adamiak, S. Anderson, M. Begovic, G. Benmouyal, G. Brunello, J. Burger, J. Y. Cai, B. Dickerson, V. Gharpure, B. Kennedy, D. Karlsson, A. G. Phadke, J. Salj, V. Skendzic, J. Sperr, Y. Song, C. Huntley, B. Kasztenny, and E. Price. Exploring the ieee standard c37.118-2005 synchrophasors for power systems. *IEEE Transactions on Power Delivery*, 23(4):1805–1811, Oct 2008.

[73] mathworks. Math. graphics. programming. `https://www.mathworks.com/products/matlab.html`. [Last visited: 27-Jul-2018].

[74] M. Mike. China tries to block encrypted traffic. `https://www.techdirt.com/articles/20121217/10222821404/china-tries-to-block-encrypted-traffic.shtml`, 2011. [Last visited: 06-Aug-2015].

[75] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg. Skypemorph: Protocol obfuscation for tor bridges. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 97–108. ACM, 2012.

[76] T. B. N. S. Monitor. Bro. `https://www.bro.org/`. [Last visited: 06-Aug-2015].

[77] K. Morison, L. Wang, and P. Kundur. Power system security assessment. *IEEE Power and Energy Magazine*, 2(5):30–39, Sept 2004.

[78] T. Morris, S. Pan, J. Lewis, J. Moorhead, N. Younan, R. King, M. Freund, and V. Madani. Cybersecurity risk testing of substation phasor measurement units and phasor data concentrators. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, CSIIRW '11, pages 24:1–24:1, New York, NY, USA, 2011. ACM.

[79] M. Naglic, M. Popov, M. A. M. M. van der Meijden, and V. Terzija. Synchro-measurement application development framework: An IEEE standard c37.118.2-2011 supported matlab library. *IEEE Transactions on Instrumentation and Measurement*, pages 1–11, 2018.

[80] R. Oppliger, R. Hauser, and D. Basin. Ssl/tls session-aware user authentication. *Computer*, 41(3):59–65, March 2008.

[81] İ. Özçelik. DoS Attack Detection and Mitigation. PhD Dissertation, Dept. of Electrical and Computer Engineering, Clemson University, 2015.

[82] J. E. Padilla, M. J. Patitz, R. Pena, R. T. Schweller, N. C. Seeman, R. Sheline, S. M. Summers, and X. Zhong. Asynchronous signal passing for tile self-assembly: Fuel efficient computation and efficient assembly of shapes. In G. Mauri, A. Dennunzio, L. Manzoni, and A. E. Porreca, editors, *Unconventional Computation and Natural Computation*, pages 174–185, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[83] PEERING. Peering the bgp testbed. `https://peering.usc.edu`. [Last visited 27-Jul-2017].

[84] R. Perdisci, I. Corona, and G. Giacinto. Early detection of malicious flux networks via large-scale passive dns traffic analysis. *Dependable and Secure Computing, IEEE Transactions on*, 9(5):714–726, 2012.

[85] B. Prince. How pushdo malware hides c&c traffic. `https://www.securityweek.com/how-pushdo-malware-hides-cc-traffic`, 2013. [Last visited: 06-Aug-2015].

[86] T. T. Project. obfsproxy. `https://www.torproject.org/projects/obfsproxy.html`, 2015. [Last visited: 06-Aug-2015].

[87] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[88] F. Y. Rashid. Zeus source code leak means even more banking malware to hit the web. `http://www.eweek.com/c/a/Security/Zeus-Source-Code-Leak-Means-Even-More-Banking-Malware-to-Hit-the-Web-253343`, 2011. [Last visited: 06-Aug-2015].

[89] S. Ray and G. Venayagamoorthy. Wide-Area Signal-Based Optimal Neurocontroller for a UPFC. *Power Delivery, IEEE Transactions on*, 23(3):1597–1605, July 2008.

[90] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482–494, 1998.

[91] RTDS Technologies Inc. Real-Time Digital Power System Simulation. `https://www.rtds.com`. [Last visited: 17-Feb-2017].

[92] RTPIS Lab. Real-Time Power and Intelligence Systems (RTPIS) Laboratory. `http://rtpis.org`. [Last visited: 06-Aug-2015].

[93] H. Ruiwen, D. Jianhua, and L. L. Lai. Reliability evaluation of communication-constrained protection systems using stochastic-flow network models. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2017.

[94] D. Sancho. Steganography and malware: Concealing code and c&c traffic. `http://blog.trendmicro.com/trendlabs-security-intelligence/steganography-and-malware-concealing-code-and-cc-traffic/`, 2015. [Last visited: 06-Aug-2015].

[95] J. M. Schwier, R. R. Brooks, C. Griffin, and S. Bukkapatnam. Zero Knowledge Hidden Markov Model Inference. *Pattern Recognition Letters*, 30(14):1273–1280, 2009.

[96] D. P. Shepard, T. E. Humphreys, and A. A. Fansler. Evaluation of the vulnerability of phasor measurement units to gps spoofing attacks. *International Journal of Critical Infrastructure Protection*, 5(3):146 – 153, 2012.

[97] V. Skeloru. Zeus 2.0.8.9 source code. `https://github.com/Visgean/Zeus`, 2011. [Last visited: 06-Aug-2015].

[98] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, Berkeley, CA, USA, 2001. USENIX Association.

[99] Sourcefire. Snort. `https://www.snort.org/`. [Last visited: 06-Aug-2015].

[100] S. Sridhar, A. Hahn, and M. Govindarasu. Cyber-physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, Jan 2012.

[101] W. Stallings. *Network and Internetwork Security: Principles and Practice*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[102] A. Stevenson. Hackers turning to tor network to hide evolved malware, warns kaspersky lab. `http://www.v3.co.uk/v3-uk/news/2335401/hackers-turning-to-tor-network-to-hide-evolved-malware-warns-kaspersky-lab`, 2014. [Last visited: 06-Aug-2015].

[103] J. Stewart, T. Maufer, R. Smith, C. Anderson, and E. Ersonmez. Synchrophasor security practices.

[104] J. Stewart, T. Maufer, R. Smith, C. Anderson, and E. Ersonmez. Synchrophasor security practices. *Schweitzer Engineering Laboratories, Pullman, Washington (¡ www. selinc. com/-WorkArea/DownloadAsset. aspx*, 2010.

[105] A. Tartakovsky. Asymptotic properties of cusum and shiryaev's procedures for detecting a change in a nonhomogeneous gaussian process. *Mathematical Methods of Statistics*, 4:389–404, 1995.

[106] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and H. Kim. A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *IEEE Transactions on Signal Processing*, 54(9):3372–3382, Sept 2006.

[107] A. Technica. Russian-controlled telecom hijacks financial services' internet traffic. `https://arstechnica.com/information-technology/2017/04/russian-controlled-telecom-hijacks-financial-services-internet-traffic/`, 2017. [Last visited 27-July-2017].

[108] S. Uludag, K. S. Lui, W. Ren, and K. Nahrstedt. Secure and scalable data collection with time minimization in the smart grid. *IEEE Transactions on Smart Grid*, 7(1):43–54, Jan 2016.

[109] L. Vanfretti, V. H. Aarstrand, M. S. Almas, V. S. Perić, and J. O. Gjerde. A software development toolkit for real-time synchrophasor applications. In *2013 IEEE Grenoble Conference*, pages 1–6, June 2013.

[110] G. Venayagamoorthy. Dynamic, stochastic, computational, and scalable technologies for smart grids. *Computational Intelligence Magazine, IEEE*, 6(3):22–35, Aug 2011.

[111] G. K. Venayagamoorthy. Computational approaches for bad data handling in power system synchrophasor networks. *IFAC Proceedings Volumes*, 47(3):11269 – 11274, 2014.

[112] G. K. Venayagamoorthy. Situational awareness / situational intelligence system and method for analyzing, monitoring, predicting and controlling electric power systems, Oct. 3 2017. US Patent 9,778,629 B2.

[113] T. Verge. A network error routed traffic for the uk's nuclear weapons agency through russian telecom. `https://www.theverge.com/2015/3/13/8208413/uk-nuclear-weapons-russia-traffic-redirect`, 2015. [Last visited 27-July-2017].

[114] T. Verge. Iran's porn censorship broke browsers as far away as hong kong. `https://www.theverge.com/2017/1/7/14195118/iran-porn-block-censorship-overflow-bgp-hijack`, 2017. [Last visited 27-July-2017].

[115] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh. Stegotorus: a camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 109–120. ACM, 2012.

[116] P. Winter and S. Lindskog. How china is blocking tor. *arXiv preprint arXiv:1204.0447*, 2012.

[117] P. Winter, T. Pulls, and J. Fuss. Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 213–224. ACM, 2013.

[118] M. Xiang, Q. Bai, and W. Liu. Trust-based adaptive routing for smart grid systems. *Journal of Information Processing*, 22(2):210–218, 2014.

[119] S. Xiao, W. Gong, and D. Towsley. *Dynamic Key Management in a Smart Grid*, pages 55–68. Springer New York, New York, NY, 2014.

[120] Y. Yan, Y. Qian, H. Sharif, and D. Tipper. A survey on cyber security for smart grid communications. *Communications Surveys Tutorials, IEEE*, 14(4):998–1010, Fourth 2012.

[121] S.-S. Yeo, S.-J. Kim, and D.-E. Cho. Dynamic access control model for security client services in smart grid. *International Journal of Distributed Sensor Networks*, 10(6):181760, 2014.

[122] L. Yu, J. Schwier, R. Craven, R. Brooks, and C. Griffin. Inferring Statistically Significant Hidden Markov Models. *Knowledge and Data Engineering, IEEE Transactions on*, 25(7):1548–1558, July 2013.

[123] L. Yuan, W. Xing, H. Chen, and B. Zang. Security breaches as pmu deviation: Detecting and identifying security attacks using performance counters. In *Proceedings of the Second Asia-Pacific Workshop on Systems*, APSys '11, pages 6:1–6:5, New York, NY, USA, 2011. ACM.

[124] S. T. Zargar, J. Joshi, and D. Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials*, 15(4):2046–2069, Fourth 2013.

[125] X. Zhong. PhasorToolBox. `http://rtpis.org/projects/phasortoolbox/`. [Last visited: 27-Jul-2018].

[126] X. Zhong, A. Ahmadi, R. Brooks, G. K. Venayagamoorthy, L. Yu, and Y. Fu. Side channel analysis of multiple pmu data in electric power systems. In *2015 Clemson University Power Systems Conference (PSC)*, pages 1–6, March 2015.

[127] X. Zhong, P. Arunagirinathan, A. Ahmadi, R. R. Brooks, and G. K. Venayagamoorthy. Side-channels in electric power synchrophasor network data traffic. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, CISR '15, pages 3:1–3:8, Oak Ridge, Tennessee, USA, 2015. ACM.

[128] X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. A virtual synchrophasor network for power system resiliency under concurrent dos attacks. *IEEE Transactions on Power Systems*. Submitted for peer review.

[129] X. Zhong, P. Arunagirinathan, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. Phasortoolbox – a python package for synchrophasor application prototyping. In *2018 Clemson University Power Systems Conference (PSC)*, 2018.

[130] X. Zhong, Y. Fu, L. Yu, R. Brooks, and G. K. Venayagamoorthy. Stealthy malware traffic - not as innocent as it looks. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 110–116, Oct 2015.

[131] X. Zhong, I. Jayawardene, G. K. Venayagamoorthy, and R. Brooks. Denial of service attack on tie-line bias control in a power system with pv plant. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(5):375–390, Oct 2017.

[132] X. Zhong, G. K. Venayagamoorthy, and R. Brooks. Bgp hijacking detection in synchrophasor network. *IEEE Transactions on Power Systems Letter*. Submitted for peer review.

[133] X. Zhong, L. Yu, R. Brooks, and G. K. Venayagamoorthy. Cyber security in smart dc microgrid operations. In *2015 IEEE First International Conference on DC Microgrids (ICDCM)*, pages 86–91, June 2015.