A Case Study on Implementing False Data Injection Attacks Against Nonlinear State Estimation

Charalambos Konstantinou
Electrical and Computer Engineering
New York University
ckonstantinou@nyu.edu

Michail Maniatakos
Electrical and Computer Engineering
New York University Abu Dhabi
michail.maniatakos@nyu.edu

ABSTRACT

Smart grid aims to improve control and monitoring routines to ensure reliable and efficient supply of electricity. The rapid advancements in information and communication technologies of Supervisory Control And Data Acquisition (SCADA) networks, however, have resulted in complex cyber physical systems. This added complexity has broadened the attack surface of power-related applications, amplifying their susceptibility to cyber threats. A particular class of system integrity attacks against the smart grid is False Data Injection (FDI). In a successful FDI attack, an adversary compromises the readings of grid sensors in such a way that errors introduced into estimates of state variables remain undetected. This paper presents an end-to-end case study of how to instantiate real FDI attacks to the Alternating Current (AC) –nonlinear– State Estimation (SE) process. The attack is realized through firmware modifications of the microprocessor-based remote terminal systems, falsifying the data transmitted to the SE routine, and proceeds regardless of perfect or imperfect knowledge of the current system state. The case study concludes with an investigation of an attack on the IEEE 14 bus system using load data from the New York Independent System Operator (NYISO).

Keywords

Cyber security; smart grid; false data injection; flash memory; firmware; reverse engineering; state estimation.

1. INTRODUCTION

State Estimation (SE) is the procedure that estimates the power system operation state based on the available measurements and the real-time network model [24]. The system measurements are collected through Remote Terminal Units (RTUs) spread out over a geographically large area and send via the SCADA network to the master units at the utility control center. Specifically, every several seconds or minutes the data is supplied into the power system estimator. Supervisory decisions are then made to adjust controls. SE is a key function of the Energy Management System (EMS), which performs various control and planning tasks. One of these tasks, for example, is contingency analysis, which determines the ability of the grid to tolerate failures. The purpose of SE, as part of EMS, is to estimate unmeasured variables, detect faulty data, and make decisions in order to maintain the stability and safety of the system.

SE infrastructure is no longer immune to cyber threats as many of the products currently being deployed were not designed with security in mind [12]. Advanced control units, such as Programmable Logic Controllers (PLCs), RTUs, data concentrators, etc., are susceptible to the same vulnerabilities as widely used general-purpose computers, since they incorporate various Commercial-Off-The-Shelf (COTS) components. The problem is further exacerbated by the lifespan of industrial systems, which is typically 15 to 20 years or more [6]. Thus, products introduced several years ago are probably incapable of addressing contemporary security concerns.

An adversary capable of obtaining access to the SCADA network could alter the measurements sent to SE. While Bad Data Detection (BDD) mechanisms have been traditionally used to eliminate erroneous data due to topology errors and measurement abnormalities, it has been shown that judiciously falsified data can introduce errors in state variables without being detected by BDD [22]. This class of attacks could compromise signals in the electricity market or even mask the outage of lines [20]. A schematic of the SE routine under FDI attack is shown in Figure 1, where the measurements and the estimated state vector are denoted by $\boldsymbol{z}$ and $\hat{\boldsymbol{x}}$, respectively. Depending on the accuracy of the estimated state variables, FDI attacks can be classified into two categories: *perfect* and *imperfect* [29]. Perfect FDI attacks assume that the adversary constructs the attack vector $\boldsymbol{a}$ based on accurate SE values. On the other hand, imperfect attacks allow the adversary to obtain the state variables with error.

FDI delivery mechanisms typically assume modification of the data flowing across the SCADA network. For example, Man in the Middle (MitM) attacks require interception of the RTUs communication to alter their reported information. Alternatively, sensor spoofing requires access to the RTU or its periphery to meaningfully modify measurement data such as current and voltage values. Therefore, these strategies require direct access either to the network or the end device. A more appealing way to deliver the FDI payload is via infiltrating secondary channels of the smart grid. As discussed before, utilities typically integrate devices pro-

CPS-SPC'16, October 28 2016, Vienna, Austria
© 2016 ACM. ISBN 978-1-4503-4568-2/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2994487.2994491

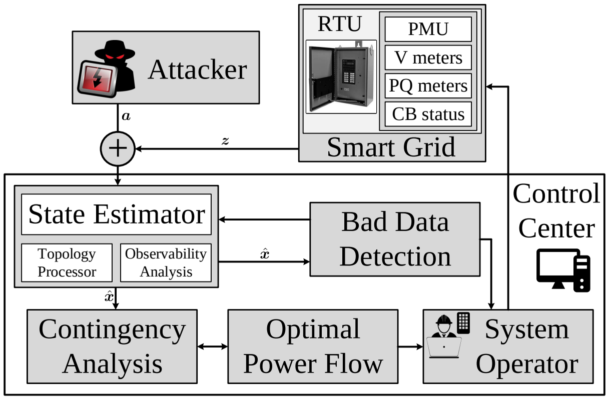

Figure 1: State estimation under FDI attack.

vided by external vendors; vendors, in turn, outsource manufacturing overseas in order to decrease fabrications costs. The attackers may opt to infiltrate any part of the supply chain in order to get access to the smart grid system.

In this paper, we implement a real FDI attack through firmware modifications. Since firmware controls the lower abstraction level of the system, i.e., the hardware, attacks at this level can evade advanced software security techniques. The attack incident against the Ukrainian power system in late 2015, where attackers locked out the control center operators through overwriting the legitimate firmware of substation serial-to-Ethernet converters, shows the importance of firmware attacks and how real the threat is [35]. Thus, we demonstrate how to reverse engineer RTU firmware and modify it in order to deliver FDI attacks. In the common case the firmware is not publicly available, we also present physical extraction of the firmware from the flash (non-volatile) memories of the RTU systems. Following the development of malicious firmware, its deployment into a grid device may follow different strategies depending on the access difficulty of each path. For example, attackers can hijack the vendor website, send malicious firmware updates by email, physically access the device, etc.

The structure of the paper is as follows: Related work is described in Section 2, while preliminaries on SE, FDI attacks, and flash memory chips are explained in Section 3. The attack methodology is presented in Section 4. Section 5 describes the reverse engineering findings, the test system, and the simulation results.

2. RELATED WORK

Several methodologies have been proposed on how to theoretically construct FDI attacks able to bypass BDD schemes [18]. Many of these methods assume that the attacker has knowledge of the system (e.g., electrical and topological data, SE and BDD algorithms, etc.) and the ability to falsify RTU measurements. Recently, studies have shown that an attacker can execute an undetected FDI attack without full knowledge of the system topology and parameter data: Liu *et al.* proposed a strategy to determine the optimal attacking region demonstrating that it is not necessary for an adversary to have knowledge of network information regarding the non-attacked region [19]. Additionally, it has been shown that FDI attacks can be implemented with incomplete information about the admittances of transmissions lines [28]. FDI attacks have also been developed for cases where the attacker can only access and modify a subset of meter measurements [3, 22].

Valid FDI attacks can also be constructed with the topology of the system being falsified. In this type of attacks, an adversary can manipulate both meter measurements and topological characteristics of the grid such as the status of switches and circuit breakers. For example, [14] demonstrates that topology attacks can mislead the control center when launched with only a small amount of resources (local information). Furthermore, falsified topological data can corrupt both the SE solution and the economic operation of smart grid [27].

All the aforementioned attacks have targeted Direct Current (DC) SE, traditionally modeled using a linear system. FDI attacks can also be developed in scenarios which the operator uses the Alternating Current (AC) SE routine [32]. Because of the nonlinear system complexity of such routines, the literature on constructing the AC model of the injected attack vector is limited: In [13], the authors introduced AC model-based FDI attacks and presented an algorithm which determines the signals an adversary needs to falsify in order to keep the attack hidden. Moreover, research has been conducted for the models of FDI attacks against AC SE in cases where the attacker has both perfect and imperfect knowledge of the system states [29, 36].

Prior work on FDI attacks against SE follow a theoretical approach, assuming a payload delivery mechanism. In comparison, this work focuses on how to deliver a real attack through reverse engineering the firmware of RTUs and disrupting their SE reporting measurements. The concept of firmware modification attacks has been reported for various types of embedded systems. For instance, [10] demonstrates that malware can be injected into printers due to a vulnerability of the remote firmware update procedure. Costin *et al.* have presented a large scale analysis of firmware images discovering 38 vulnerabilities [8]. In the context of smart grid, an attacker able to compromise the central system of Advanced Metering Infrastructure (AMI) can distribute malicious firmware to smart meters [9]. Peck *et al.* have demonstrated a procedure to load malicious code into field device Ethernet cards due to the lack of firmware authentication [26]. Furthermore, firmware vulnerabilities in wireless access points and recloser controllers could cause serious erosion of the power system's stability margin [16, 33]. The continuous discovery of firmware vulnerabilities, combined with the fact that most microprocessor-based RTUs run firmware [30], render firmware attacks one of the most advanced threats for the smart grid.

3. PRELIMINARIES

In this section we provide a theoretical review of SE and FDI attacks. In addition, we describe how to physically extract and reverse engineer firmware from the non-volatile (flash) memories of the smart grid devices, in order to prepare a firmware modification attack.

3.1 State Estimation

The SE function as part of the EMS consists of various processes. The topology processor tracks the status of breakers to maintain a real-time network diagram. Observability analysis ensures that SE can be performed with the available set of measurements. BDD identifies and eliminates data that contain errors caused by communication failures or biased meters. SE estimates the system state based on the available measurements. The obtained state variables are used in other functions of the EMS, such as optimizing power flows, transmission stability analysis, load shedding, etc.

Based on the measurement function that models the relation between system and measurement states, there are two methods to perform SE: The AC estimation uses a nonlinear function to capture the relation of measurements and state variables. On the other hand, DC estimation corresponds to a simplified, linear version of the AC SE.

3.1.1 AC State Estimation

In the AC estimation, the SE routine utilizes a nonlinear relationship between the system and measurement elements:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{e} \tag{1}$$

$\boldsymbol{x} \in \mathbb{R}_{n\times 1}$ denotes the vector of state variables of a system with cardinality $\mathcal{S}$, where $n = 2S - 1$ (a state variable – typically the phase angle– is taken as reference at one of the system buses), and $\boldsymbol{z} \in \mathbb{R}_{m\times 1}$ is the vector of measured values, where m is the number of meter measurements, and $m \geq n$ in order for state variables $\boldsymbol{x}$ at each bus $i \in \mathcal{S}$ to be determined (the system becomes observable). The state variables include the voltage magnitudes and voltage angles at the buses of the system and the vector of measured values may include active and reactive power injections P_i and Q_i at each bus $i \in \mathcal{S}$, active and reactive power flows P_{ij} and Q_{ij} at each transmission line between two buses $i, j \in \mathcal{S}$, voltage magnitudes and angles. $\boldsymbol{e} \in \mathbb{R}_{m\times 1}$ is the vector of measurement errors and $\boldsymbol{h}()$ is the nonlinear function determined according to the physical structure of the system. Function $\boldsymbol{h}()$ provides the relationship between state variables $\boldsymbol{x}$ and measured values $\boldsymbol{z}$ calculated based on the power flow and power injection equations:

$$P_{ij} = V_i^2 g_{ij} - V_i V_j (g_{ij}\cos\delta_{ij} + b_{ij}\sin\delta_{ij}) \tag{2}$$

$$Q_{ij} = -V_i^2 (b_{ij} + b_{ij}^{sh}) + V_i V_j (b_{ij}\cos\delta_{ij} - b_{ij}\sin\delta_{ij}) \tag{3}$$

$$P_i = V_i \sum_{j\in\mathcal{S}_i} V_j (g_{ij}\cos\delta_{ij} + b_{ij}\sin\delta_{ij}) = \sum_{j\in\mathcal{S}_i} P_{ij} \tag{4}$$

$$Q_i = V_i \sum_{j\in\mathcal{S}_i} V_j (g_{ij}\sin\delta_{ij} - b_{ij}\cos\delta_{ij}) = \sum_{j\in\mathcal{S}_i} Q_{ij} \tag{5}$$

In (4) and (5), $\mathcal{S}_i \subseteq \mathcal{S}$ indicates the set of buses connected to bus i. The argument of the sinusoidal functions, $\delta_{ij} = \delta_i - \delta_j$, denotes the voltage phase angle difference between buses i and j. Moreover, g_{ij} and b_{ij} form the line series admittance y_{ij} and correspond to the conductance and susceptance of the line between buses i and j, respectively. Similarly, $y_{ij}^{sh} = g_{ij}^{sh} + b_{ij}^{sh}$ represents the shunt admittance between line and ground of the transmission line from bus i to bus j.

The relation between state variables and measurements is given by $\boldsymbol{h}(\boldsymbol{x})$ in (1). The Jacobian matrix $\boldsymbol{J}_h$ of $\boldsymbol{h}(\boldsymbol{x})$ provides the information with regards to which measurement is dependent on which state variable. Each entry in row i and column j of the matrix corresponds to the derivative of ith measurement with respect to the jth state variable according to (2) – (5).

Algorithm 1 Honest Gauss Newton Method

```
1: procedure HONEST GAUSS NEWTON
2:     $\boldsymbol{x}_0 = \big[\delta_{2_0},\ \delta_{3_0},\ \ldots,\ \delta_{S_0},\ V_{1_0},\ V_{2_0},\ \ldots,\ V_{S_0}\big]^T$
           $= \big[0,\ 0,\ \ldots,\ 0,\ 1,\ 1,\ \ldots,\ 1\big]^T$
3:     for each measurement $\boldsymbol{z}$ and the estimate $\boldsymbol{x}$ do
4:         $\Delta\boldsymbol{x}_k = \left(\boldsymbol{J}_h{}^T \cdot W \cdot \boldsymbol{J}_h\right)^{-1} \cdot \boldsymbol{J}_h{}^T \cdot W \cdot (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}_k))$
5:         $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x}_k$
6:         Update nonlinear function vector $\boldsymbol{h}(\boldsymbol{x}_{\boldsymbol{k+1}})$
7:         Update Jacobian matrix $\boldsymbol{J}_h$ based on $\boldsymbol{x}_{k+1}$
8:     end for
9: end procedure
```

The procedure for solving the AC SE problem is via iterative techniques such as the Honest Gauss Newton method, the Dishonest Gauss Newton method, and the Fast Decoupled State Estimator [24]. The Honest Gauss Newton method (also known as the basic Weighted Least Square (WLS) estimation) is shown in Algorithm 1, where W is a diagonal weight matrix (typically inverses of measurement noise variance, R^{-1}). In Dishonest Gauss Newton, step 7 is excluded. The accuracy of the estimated variables $\hat{\boldsymbol{x}}$ is calculated via the Euclidean norm of the residual:

$$\|\boldsymbol{r}\| = \|\boldsymbol{z} - \boldsymbol{h}(\hat{\boldsymbol{x}})\| \tag{6}$$

For the Dishonest Gauss Netwon and the scenario which the algorithm converges to the true system state $\boldsymbol{x}$, then:

$$\Delta\boldsymbol{x}_k = 0 \implies \boldsymbol{F}(\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{x}_k)) = 0 \tag{7}$$

where:

$$\boldsymbol{F} = \left(\boldsymbol{J}_h{}^T \cdot W \cdot \boldsymbol{J}_h\right)^{-1} \cdot \boldsymbol{J}_h{}^T \cdot W \tag{8}$$

3.1.2 DC State Estimation

Contrary to AC SE, DC SE state variables include only the voltage angles. DC SE assumes that bus voltage magnitudes to be constant and equal to one, branch resistances and shunt elements are negligible, and angle differences between buses are small.

The relationship between measurements and state variables in DC SE is based on the following linear function:

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{e} \tag{9}$$

where the linear matrix $\boldsymbol{H} \in \mathbb{R}_{m\times n}$ is the Jacobian matrix $\boldsymbol{J}_h$ in AC SE, corresponding to the derivatives of the DC-equivalent version of power flow and power injection equations with respect to the voltage phase angles.

3.1.3 Bad Data Detection

The errors contained into the acquired data from RTUs are identified by BDD algorithms. Most of these schemes are based on the largest normalized residual method, i.e., on the residual between the obtained measurements and the values for these measurements as a function of the estimated system

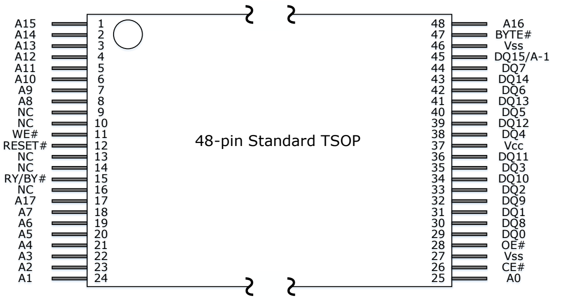

Figure 2: Typical electrical interface of a 16-bit NOR chip (Spansion S29AL004D).

states. In the scenario which the residual in (6), violates the condition $\|\boldsymbol{r}\| < \boldsymbol{\epsilon}$ (where $\boldsymbol{\epsilon}$ a predefined threshold), then there is at least one faulty measurement.

3.2 Flash Memory Chips

During the last decades, flash memory has been the most dominant electronic non-volatile storage medium. As such, it has widespread use in embedded systems including portable drives, digital cameras, and Solid-State Disks (SSDs). Flash memory, as a computer storage medium, can be electrically erased and rewritten and its non-volatility allows to maintain and retrieve stored data even after having been power cycled. Coming from the advancement of EPROM (Erasable Programmable Read-Only Memory), there are primarily two types of non-volatile flash memories: NAND and NOR [31].

NAND memory is best suited for file or sequential-data applications (e.g., USB disks, digital cameras) while NOR chips are best suited for random access (memory mapped). This characteristic gives NOR its execute-in-place (XiP) functionality, which is often required in embedded systems. NOR chips have also slightly higher read performance. On the other hand, NAND memories (typically accessed in bursts of 512 bytes) write and erase much faster. In terms of cost and capacity, NOR technology is typically available from $1MB$ to $16MB$ and is mostly cost effective in low capacities ($1MB$-$4MB$). On the contrary, NAND chips are overall better priced and mainly available from $8MB$ to $128MB$. However, the random access of NOR flash (XiP) makes it an attractive solution for code storage and execution, i.e., to hold and execute firmware.

In terms of structural differences, NAND memory has several benefits over NOR. For instance, $32MB$ NAND is about half the size of $16MB$ NOR. The hardware design of NOR requires approximately 48 I/O pins for a 16-bit device while NAND may require half the pins for a similar interface. An example of the NOR electrical interface is shown in Figure 2 with the pin names described in Table 1. In order to read array data from the chip outputs, the system must drive the $CE\#$ and $OE\#$ pins to V_{IL} while $WE\#$ and $RESET\#$ should remain at V_{IH}. The $BYTE\#$ pin controls whether the device outputs array data in words or bytes. Under these requirements, the data stored at the memory location determined by the address pins is asserted to the outputs.

3.2.1 Memory Data Acquisition

In this part, we present data acquisition techniques to extract the data from flash memories. We focus on cases where non-volatile chips store firmware code and data. There are three main acquisition approaches to extract firmware from flash chips [4], presented in the following paragraphs: *a)* using flasher tools, *b)* through JTAG[1] test access port, and *c)* via physical extraction of the chip.

Table 1: Pin Configuration of a 16-bit NOR flash chip.

Pin	**Description**
$A0 - A17$	18 addresses
$DQ0 - DQ14$	15 data inputs/outputs
$DQ15/A-1$	DQ15 (data input/output, word mode) A-1 (LSB address input, byte mode)
$BYTE\#$	Selects 8-bit or 16-bit mode
$CE\#$	Chip enable
$OE\#$	Output enable
$WE\#$	Write enable
$RESET\#$	Hardware reset pin, active low
$RY/BY\#$	Ready/Busy# output
V_{cc}	3.0 volt-only single power supply
V_{ss}	Device ground
NC	Pin not connected internally

Flasher tools: Flasher tools are commercial hardware and software devices that allow to flash the contents of memory chips. Flasher tools, if available for the specific chip model, provide the most easy and non-invasive process to read memory information. However, it cannot be guaranteed that data is not written in flash memory. Furthermore, the tool may skip parts of the memory and therefore extract only a partial image of the flash contents [23].

JTAG: External interfaces such as JTAG and serial access ports can be used beyond their specification purposes. The JTAG protocol is a serial protocol with an extra mode selection pin TMS (Test Mode Select). The JTAG test access port is used at the end of the production phase to check the integrated circuit for defects. Moreover, JTAG can also be used to debug embedded systems and access flash memory. JTAG access ports are often difficult to locate on the embedded system board, and, even if a JTAG port exists (not all embedded devices are JTAG enabled), vendors commonly lock (reversible) or burn (irreversible) the JTAG interface for security purposes.

Physical extraction: Another way to obtain a memory image is to physically remove the chip from the Printed Circuit Board (PCB) and extract its contents via a chip reader using a three-step process: The first step is to de-solder the chip. The preferred method for removing either a TSOP[2] or a micro BGA[3] chip is with hot air using a rework station. For TSOP-based memories, hot air is blown on the edges of the chip evenly until the solder alloy melts and the chip can be removed from the board (using tweezers, vacuum air gripper, or any other tool). The second step includes preparing the chip for reading, by ensuring that the pins are cleared from the old solder and are aligned correctly. The last step is setting the equipment for reading, as shown in Figure 3. Based on the chip model, the code for reading the memory is transferred from the PC to a microcontroller or an FPGA

[1] JTAG: Joint Test Action Group - IEEE Std 1149.1 Standard Test Access Port and Boundary-Scan Architecture [2].

[2] TSOP: Thin Small-Outline Package.

[3] BGA: Ball Grid Array.

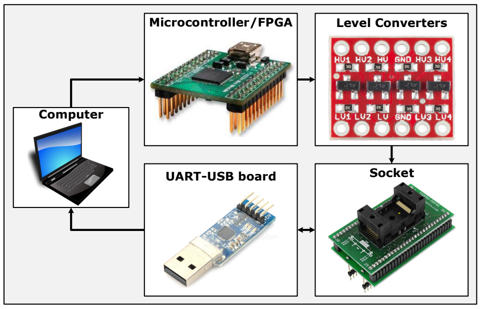

Figure 3: Flash chip reader setup.

by selecting the appropriate protocol. The board is used for protocol conversion and communication with the chip. Level converters transform electrical signals to the voltage levels required by the memory. The TSOP or BGA socket is used for connection to the flash chip. Finally, the memory data is sent to the PC through a UART-USB board to store the image file. After data acquisition, the chip can be re-soldered to the PCB.

3.2.2 *Firmware Reverse Engineering*

Reverse engineering is the process of extracting data about the operation of a system in order to determine its original design. The procedure is achieved via disassembling and analyzing the system components. In the firmware scenario, the goal is to locate and extract functional blocks (e.g., binary code, scripts, configuration files, web interfaces, etc.) from the firmware package [34]. Following this process, an adversary can reveal information of the system features, leverage code vulnerabilities, and even modify parts of the image in order to maliciously alter the system functionality.

The procedure of firmware modification attacks typically begins through reconnaissance of the image to gain insights about the firmware structural contents (boot loader, kernel and file system details, etc.). The reconnaissance phase is followed by data extraction. This task allows to identify standardized machine code formats (e.g., PE and ELF), groups of files (e.g., TAR and LZMA), and other resources [7]. The examination of these files can lead to the discovery of vulnerabilities and also allow an attacker to modify and exploit the unpacked file system. Finally, the attacker is required to repack and upload the malicious image back to the device.

Firmware reverse engineering can be a long and tedious procedure. Often, vendors develop their own file formats to describe flash images. In other cases, the files are compressed with non-standard algorithms. Also, the firmware may include data integrity checks: For instance, checksums added in various parts of the firmware add an extra difficulty layer towards delivering a firmware modification attack, as the adversary needs to identify the algorithms and generate proper responses for the modified firmware. In the event of checksum mismatch, the device could be permanently disabled ("bricked"). In addition, vendors may further obfuscate, encrypt, or sign the memory-packed firmware data to hinder the reverse engineering process (e.g., keyed-hash message authentication code - HMAC). Our experience, however, with security assessment of smart grid equipment shows that most systems are not sufficiently supplemented with adequate security mechanisms.

4. ATTACK METHODOLOGY

The attack is split in two distinct phases: the *payload generation* phase, where the FDI attack model requirements are examined, and the *payload delivery* phase, where the appropriate steps are followed to carry the attack.

4.1 Payload Generation

In the context of FDI attacks, adversaries are classified into different types according to their skills and knowledge [21]: *Innocent-curious attackers* are motivated to learn system details but do not possess the capabilities to extract information of the network model or the RTUs. *Skilled attackers* can acquire data about the electrical and topological structure of smart grid. In addition, they can obtain information regarding the utilized protocols and RTUs. Although they have skills to create a large-scale disruption, they cannot gain access to critical equipment or areas of the system. *Powerful-motivated attackers* can acquire detailed information about the grid and its components as well as infiltrate the system in order to cause undesirable state changes. We focus on SE data-driven FDI attacks performed by both *skilled* and *powerful-motivated* attackers with full or partial knowledge of the network topology and parameters.

4.1.1 *FDI Attack Model against AC SE*

The goal of an adversary is to corrupt the measurement data from a subset of RTUs $D_{\mathcal{S}}$. As a result, the measured data will become $\boldsymbol{z_a} = \boldsymbol{z} + \boldsymbol{a}$. Given the false measurement vector $\boldsymbol{z_a}$, the SE solution becomes $\hat{\boldsymbol{x}}_a \neq \hat{\boldsymbol{x}}$, where $\boldsymbol{a} \in \mathcal{A}$ is the attack vector and $\mathcal{A}$ is the set of feasible attack vectors:

$$\mathcal{A} \triangleq \{\boldsymbol{a} \in \mathbb{R}_m : a_i = 0, \forall i \notin D_{\mathcal{S}}\} \tag{10}$$

In the simple scenario where the DC (linear) SE model is utilized, the malicious vector $\boldsymbol{a}$ can pass the BDD scheme iff $\boldsymbol{a}$ can be expressed as a linear combination of $\boldsymbol{H}$, that is $\boldsymbol{a} = \boldsymbol{Hc}$ for an arbitrary vector $\boldsymbol{c}$. In such scenario, the residue-based BDD cannot detect the FDI as the residual under attack is not affected:

$$\begin{aligned} \|\boldsymbol{r}_a\| &= \|\boldsymbol{z}_a - \boldsymbol{H}\hat{\boldsymbol{x}}_a\| = \|\boldsymbol{z} + \boldsymbol{a} - \boldsymbol{H}(\hat{\boldsymbol{x}} + \boldsymbol{c})\| \\ &= \|\boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}}\| = \|\boldsymbol{r}\| \end{aligned} \tag{11}$$

In AC SE, however, the power flows between buses and power injections at each bus are nonlinearly dependent on voltage magnitudes and voltage angles. If an attacker is able to get exactly the same system state $\hat{\boldsymbol{x}}$ as the control center, then a perfect FDI attack vector can be developed. In such scenario, the residue under attack is:

$$\begin{aligned} \|\boldsymbol{r}_a\| &= \|\boldsymbol{z}_a - \boldsymbol{h}(\hat{\boldsymbol{x}}_a)\| \\ &= \|\boldsymbol{z}_a - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\hat{\boldsymbol{x}}) - \boldsymbol{h}(\hat{\boldsymbol{x}})\| \\ &= \|\boldsymbol{z} + \boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\hat{\boldsymbol{x}}) - \boldsymbol{h}(\hat{\boldsymbol{x}})\| \\ &= \|\boldsymbol{r} + \boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\hat{\boldsymbol{x}})\| \end{aligned} \tag{12}$$

where $\hat{\boldsymbol{x}}$ is the estimated state vector (which for a perfect FDI attack is equal to the true estimated state vector $\boldsymbol{x}$), $\hat{\boldsymbol{x}}_a = \hat{\boldsymbol{x}} + c$ is the estimated state under attack, r is the measurement residual vector, and a and c are the attack vector and the changes in the attacked state variables, respectively.

For the FDI to be hidden from the BDD algorithm, the upper bound of $\|\boldsymbol{r}_a\|$ must be equal to $\|\boldsymbol{r}\|$. Therefore, the attack vector $\boldsymbol{a}$ must be chosen according to:

$$\|\boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\boldsymbol{x})\| = 0 \tag{13}$$

In this case, based on (7) and the assumption that Dishonest Gauss Newton algorithm converges to a fixed vector $\widetilde{\boldsymbol{x}}$, the following must be fulfilled:

$$\begin{aligned}\Delta\boldsymbol{x} = 0 &\implies \boldsymbol{F}(\boldsymbol{z}_a - \boldsymbol{h}(\widetilde{\boldsymbol{x}})) = 0\\ &\implies \boldsymbol{F}(\boldsymbol{z} - \boldsymbol{h}(\hat{\boldsymbol{x}})) + \boldsymbol{F}(\boldsymbol{h}(\hat{\boldsymbol{x}}_a) - \boldsymbol{h}(\widetilde{\boldsymbol{x}})) = 0\\ &\implies \boldsymbol{F}(\boldsymbol{h}(\hat{\boldsymbol{x}}_a) - \boldsymbol{h}(\widetilde{\boldsymbol{x}})) = 0\end{aligned} \tag{14}$$

According to (14), if $[\boldsymbol{h}(\hat{\boldsymbol{x}}_a) - \boldsymbol{h}(\widetilde{\boldsymbol{x}})]$ is not in the null space of matrix $\boldsymbol{F}$ then $\boldsymbol{h}(\hat{\boldsymbol{x}}_a) = \boldsymbol{h}(\widetilde{\boldsymbol{x}})$. The structure of the nonlinear function vector $\boldsymbol{h}(\cdot)$ may indicate that in practical cases $\hat{\boldsymbol{x}}_a = \widetilde{\boldsymbol{x}}$. Therefore, the attack vector $\boldsymbol{a}$ satisfying (13) can cause an undetectable attack able to bypass BDD.

Besides FDI attacks in which state variables can be accurately estimated, imperfect attacks allow adversaries to obtain the estimated system states with error [36]. The inaccuracy in the attacker's obtained SE solutions relax the constraint of the attack vector in (13). As a result, the imperfect FDI model against AC SE introduces a relaxing error $\boldsymbol{\tau}$ and the measurement residual vector under attack (bypassing the BDD scheme $\implies \|\boldsymbol{r}\| \leq \boldsymbol{\epsilon}$) becomes:

$$\begin{aligned}\|\boldsymbol{r}_a\| &= \|\boldsymbol{r} + \boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\hat{\boldsymbol{x}})\|\\ &\leq \|\boldsymbol{r}\| + \|\boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\hat{\boldsymbol{x}})\|\\ &= \|\boldsymbol{r}\| + \|\boldsymbol{\tau}\| \leq \boldsymbol{\epsilon}\end{aligned} \tag{15}$$

Thus, the constraint of (13) is modified and satisfies:

$$\|\boldsymbol{a} - \boldsymbol{h}(\hat{\boldsymbol{x}}_a) + \boldsymbol{h}(\boldsymbol{x})\| \leq \boldsymbol{\tau} = \boldsymbol{\epsilon} - \|\boldsymbol{r}\| \tag{16}$$

Consequently, the requirements to ensure that the FDI based on imperfect information remains undetected are: *1)* the attack vector $\|\boldsymbol{a}\|$ satisfies (16), and *2)* the difference between the received and estimated measurements (as a function of the estimated state) – the true residue $\|\boldsymbol{r}\|$ – satisfies the residue test $\|\boldsymbol{r}\| < \boldsymbol{\epsilon}$.

In the FDI scenario where the relaxing error inequality $\|\boldsymbol{\tau}\| \neq 0$ cannot hold, the FDI follows the perfect FDI attack model, i.e., the perfect FDI is a special case of the imperfect FDI with the relaxing error set to zero.

4.1.2 Case Study Requirements for FDI attacks

The FDI attack model either requires knowledge of the system topology and parameters or imposes constraints which allow only limited system information. In the latter scenario, a small fraction of measurements (sufficient to make the system observable) may provide enough data upon which SE attacks can be constructed [15]. In the following, we present our case study requirements for synthesizing stealthy attacks targeting AC SE via secondary payload delivery channels.

Data acquisition in smart grid begins at the RTU (or PLC) level. We focus our attention to these field devices hardware and firmware as often adversaries target these layers to remain undetected and effective as long as possible. In the security context of smart grid, the role of hardware (e.g., flash memories) and firmware as the root of trust is essential, as all other layers rely and build upon them. Figure 4 shows the conceptual diagram of how an attacker would construct the attack on the measurement data. The adversary reconstructs the memory contents of RTUs to send falsified data to the SCADA system and SE routines. In other words, the attacker injects malicious firmware in those RTUs that contribute to the computation of a particular system state.

The first prerequisite for the presented FDI attack model is to bypass integrity functions $q_k(\hat{\boldsymbol{x}}, \boldsymbol{z})$, $k = 1, 2, ..., m$ within the firmware code, used as built-in protections against modifications. The binary code and data residing in flash memories often include protection mechanisms (e.g., data integrity checksum functions) that will either halt the device or even deny and rollback any unapproved changes. The error in the acquired system states constitutes also an important functional barrier for the practical implementation of FDI attacks. As shown in Section 4.1.1, imperfect attacks allow a percentage of inaccuracy in the attacker's acquired SE solution. However, in order to guarantee that the attack is hidden from BDD, the residue under attack should follow (15), i.e., the relaxing error $\boldsymbol{\tau}$ should lie within the range of certain error bounds: $[0, \boldsymbol{\epsilon} - \|\boldsymbol{r}\|]$. As the relaxing error $\boldsymbol{\tau}$ approaches zero, the FDI attack follows the perfect FDI attack model in which the adversary obtains precisely and without error the required state variables.

Therefore, the proposed payload delivery method models imperfect FDI attack as an optimization procedure which considers the aforementioned practical problems in corrupting the RTU measurement data. The attacker, aware of the RTU model, extracts the firmware from the device hardware and designs the modification attack such as firmware checks are validated correctly while the error within the attack vector is minimized, i.e., the function $q_k(\hat{\boldsymbol{x}}, \boldsymbol{z})$ is equal to the pre-modified value β_k for every altered measurement signal $k = 1, 2, ..., m$, and the attack vector is designed in such a way aiming to optimize the relaxing error $\boldsymbol{\tau}$ (minimize it towards zero). Apparently, since we are effectively trying to identify checksum "collisions", different data integrity schemes will provide different opportunities for minimizing the error. Parity byte/word and checksum should offer many possible solutions, while Cyclic Redundancy Checks (CRCs) would greatly limit the number of possible sets of the error $\boldsymbol{\tau}$. Following this process, the adversary finds the optimal attack values satisfying both requirements in order to construct the firmware and inject it to the desired RTUs.

In this context, we apply the method of Lagrange multipliers for identifying the minimum values of the relaxing error $\boldsymbol{\tau}$ subject to the firmware check constraints:

$$\begin{aligned}\boldsymbol{\mathcal{L}}(\hat{\boldsymbol{x}}, \boldsymbol{z}, \boldsymbol{\lambda}) &= \boldsymbol{p}(\hat{\boldsymbol{x}}, \boldsymbol{z}) - \boldsymbol{\lambda} \cdot \boldsymbol{q}(\hat{\boldsymbol{x}}, \boldsymbol{z})\\ &= \boldsymbol{p}(\hat{\boldsymbol{x}}, \boldsymbol{z}) - \sum_{k=1}^{m} \lambda_k \cdot (\beta_k - q_k(\hat{\boldsymbol{x}}, \boldsymbol{z}))\end{aligned} \tag{17}$$

where $\boldsymbol{p}(\hat{\boldsymbol{x}}, \boldsymbol{z})$ is the objective function that minimizes $\boldsymbol{\tau}$ based on (16), and $\boldsymbol{q}(\hat{\boldsymbol{x}}, \boldsymbol{z})$ is the firmware validation equality constraint. If the measurements $\boldsymbol{z}^* = \begin{bmatrix} z_1^*, & z_2^*, & \dots, & z_m^* \end{bmatrix}^T$ minimize $\boldsymbol{p}(\hat{\boldsymbol{x}}, \boldsymbol{z})$ subject to the constraint $q_k(\hat{\boldsymbol{x}}, \boldsymbol{z}) = \beta_k$, for $k = 1, 2, ..., m$, then either the vectors $\nabla q_1(\hat{\boldsymbol{x}}, \boldsymbol{z}^*), ..., \nabla q_m(\hat{\boldsymbol{x}}, \boldsymbol{z}^*)$ are linearly dependent, or there exist vector $\boldsymbol{\lambda}^*$ of size m, named the *Lagrange multiplier*, such that:

$$\nabla \boldsymbol{p}(\hat{\boldsymbol{x}}, \boldsymbol{z}^*) - \sum_{k=1}^{m} \lambda_k^* \nabla q_k(\hat{\boldsymbol{x}}, \boldsymbol{z}^*) = 0 \tag{18}$$

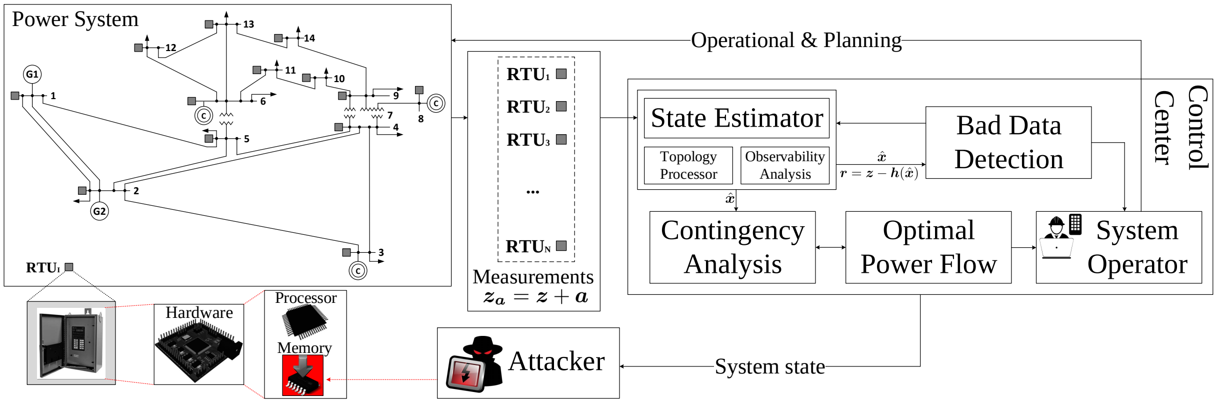

Figure 4: Overall architecture of the memory-based reverse engineering FDI attack against state estimation.

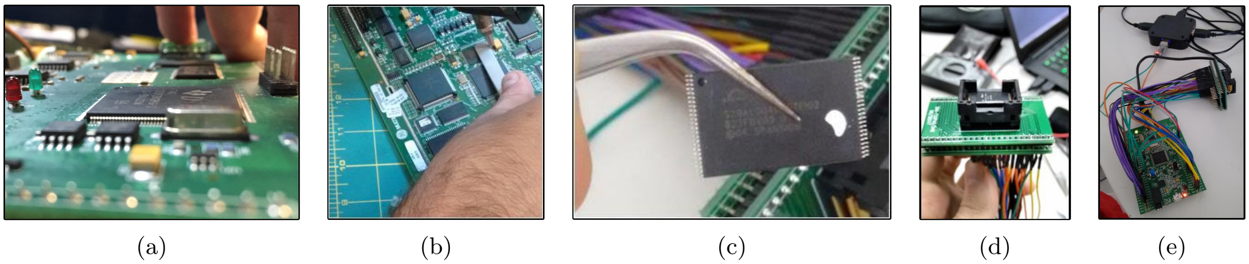
(a)
(b)
(c)
(d)
(e)

Figure 5: Flash memory data acquisition process: (a) Printed Circuit Board (PCB), (b) de-soldering, (c) flash chip, (d) memory chip in the TSOP socket, (e) chip reader connections with STM32F4DISCOVERY Discovery board.

In this paper, we examine the latter case in order to find candidate solutions for the attack vector $\boldsymbol{a}$ in the falsified measurement vector $\boldsymbol{z}_a^* = \boldsymbol{z}^* + \boldsymbol{a}$.

4.2 Payload Delivery

The attacker's knowledge of the system allows the development of either perfect or imperfect FDI attacks. The compromise of just one RTU allows the adversary to manipulate multiple measurements transmitted to the SE process [13,19]. This attack scenario can be achieved by altering the residing firmware information of flash memories in order to execute malicious code or circumvent firmware critical functions in the code flow execution.

The attack methodology to disrupt the operation of RTU devices is the following:

1. *Identify SCADA equipment*: Models of RTU controllers, communication protocols, and data acquisition products deployed in the system should be identified. Public resources regarding the operation of the grid can be utilized by adversaries, e.g., newsletters and success stories can reveal these technologies [17].

2. *Locate control and monitoring devices*: Find the location of control and monitoring equipment in the power system. Similarly to the previous step, web accessible data can be utilized [17]. In addition, an adversary can locate devices through physical inspection or internet connectivity.

3. *Firmware acquisition*: Since the attacker is aware of the RTU model, the firmware of the device can be obtained from the product vendor or distributor (e.g., website, email, etc.). In case the firmware of the unit is not accessible, the product can be purchased and its firmware extracted physically. The procedure described in Section 3.2 can reveal all the required firmware information for a successful attack: *a)* how to bypass firmware protections, e.g., hard-coded credentials, encryption and checksum algorithms, and *b)* construct malicious firmware by exploiting code vulnerabilities. This step will also provide guidance to the adversary of how the device operates and interacts with other systems.

4. *Firmware injection*: In order to upload the developed malicious firmware into the RTU device, attackers may follow different strategies depending on the access difficulty of each penetration path. For example, adversaries can hijack a vendor or distributor website[4] by creating a rogue copy of it. In addition, RTU devices may be compromised via social engineering. For instance, infected USB drives dropped in a parking lot could be a way of getting the malicious image into the utility network. This method was used to propagate Stuxnet in otherwise considered air-gapped networks [11]. The malformed firmware can be also uploaded to the RTU by a malicious insider.

5. CASE STUDY

In this section, we describe the findings of analyzing memory data from three commercial RTUs. We also explain how

[4] The hijacked website typically shows contents similar to the original, but instead redirects to a malicious server for downloading a malicious firmware.

Table 2: Memory characteristics of RTU designs.

RTU device	Model	Type	Size	Package
A	Spansion S29AL004D	NOR	4 Mbit	48-pin TSOP
	AMD Am29F400B	NOR	4 Mbit	44-pin SOP
B	Atmel AT29C040A	NOR	4 Mbit	32-pin TSOP
	Atmel AT29C040A	NOR	4 Mbit	32-pin TSOP
C	Spansion S29AL008J	NOR	8 Mbit	48-pin TSOP

to prepare a case system and we present simulation results demonstrating the performance of the developed attacks.

5.1 Flash Data Acquisition and Analysis

In order to evaluate the proposed methodology, we first follow the chip-off data acquisition process described in Section 3.2, i.e., physically remove the chips from the RTU PCB and extract its contents via a chip reader as shown in Figure 5. Specifically, we de-solder the flash memories from three commercial RTUs (A,B,C)[5]. The memory findings are shown in Table 2. All of the RTUs support NOR technology of 4 and 8 *Mbit* size. The acquired flash chips contents reveal that the Spansion model of RTU device A is used for data storage and the AMD model contains the firmware code. RTU B includes two identical flash chips which both contain firmware code and data. Finally, RTU C has only one non-volatile memory holding all the necessary file system details of the device.

The next part of the process includes reading the bare metal image from the chip (steps (d) and (e) in Figure 5). We build the flash reader circuit of Figure 5 (e) based on the outline of equipment setup shown in Figure 3. The code for reading the flash memories is transferred from the PC to an ARM Cortex-M4 based STM32F4DISCOVERY Discovery board. The microcontroller platform reads the flash memory using a technique called bit-banging. The method is mainly used for serial communications and allows to directly interact with chips through software instead of dedicated hardware. For all the addresses, we bit-bang the address into the shift registers of the memory, set the control pins to the appropriate low or high voltage levels, read the byte from the NOR flash and send it over the serial UART line.

To ensure that the acquired binary image from the desoldered flash chip is correct, we dump the data multiples times to check that the obtained files are identical. Statistically, the read errors are not at the same offset each time. Therefore, for each chip, we read the files by chunks to prevent memory exhaustion problems and we compare every byte in order to select the most frequent one for the output.

Once we have the binary ROM data for each RTU, we extract standardized machine code formats and groups of files. Then, we start the image analysis; we identify the Instruction Set Architecture (ISA) of each image and examine its data via interactive disassemblers and debuggers. The ISA identification can be achieved by inspecting the binary data for common patterns. For instance, for an embedded PowerPC processor common patterns include [4E 80 00 20] (blr - return), [48 xx xx xx] (branches), and [7C 08 xx A6] (mtlr %rx - save link register). Our analysis reveals that the firmware image of device A is designed for a Motorola/Freescale ColdFire ISA. Among others, the default credentials are hard-coded into the firmware code. The data extracted from the flash chip of the second RTU is identified to be a 32-bit Executable and Linking Format (ELF) file designed for a PowerPC instruction set. The firmware analysis uncovered 127,682 lines of assembly code including the `SHA-1` hash value of the default access credentials and the structure of functioning critical modules (e.g., DNP3 protocol configuration routines, control checks for time-current curve data calibration, etc.). Finally, the extracted data from device C is encrypted; the boot ROM included in the TI TMS320F2812 Digital Signal Processor (DSP) of the RTU locates and loads the firmware which is then being decrypted on the fly after power-on or reset.

To demonstrate the implementation of FDI attacks through flash memory data modifications, we focus on the firmware reverse engineering of the RTU device B. The implemented attack modifies the setup profiles specifying the metering procedure ($\boldsymbol{z}$) of the RTU. This is accomplished by maliciously altering the calibration control mechanisms encompassed in the firmware initialization process. Particularly, we modify the calibration registers within the booting process of the firmware in order for the RTU firmware measurement subroutines to report erroneous data. For the firmware attack to be successful however, it is required to also change the validation fields of the uncompressed firmware. The resulting assembly code revealed that the ELF file is checked using a 16-bit Cyclic Redundancy Check (CRC) function:

$$q(\boldsymbol{y}) = y^{\beta} \cdot M(y) \bmod G(y) \tag{19}$$

where $\beta = 16$ is the degree of the generator polynomial $G(y) = y^{16} + y^{15} + y^2 + 1$, and $M(y)$ is the original message polynomial to be examined.

The CRC algorithm is used as a self-checking integrity mechanism to detect changes to the firmware after bootstrapping and inventory of the system resources. It recalculates the checksum value $q(\boldsymbol{y})$ when the firmware metering subroutines are modified. Hence, each modification to the RTU measurement vector $\boldsymbol{z}$ via the firmware calibration subroutine is performed in accordance to (17), i.e., minimize $\boldsymbol{\tau}$ while the residue test holds and the CRC remains the same.

5.2 Test System

In this part, we describe how to prepare the data of the test system in order to examine our case study methodology. The developed data is exported to MATLAB and power system analysis and estimation is applied on the IEEE 14 benchmark system (Figure 6) using MATPOWER package.

[5] The paper does not include the name of the target RTUs in order to avoid exposing vulnerabilities.

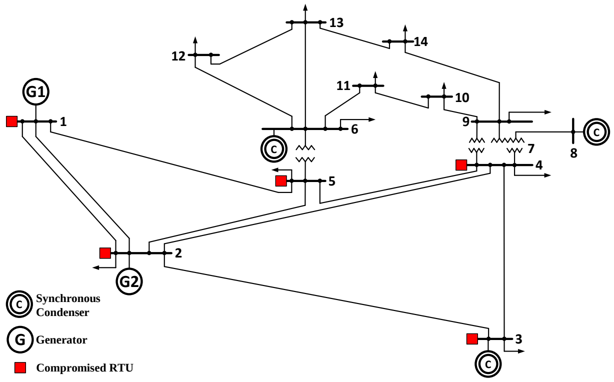

Figure 6: IEEE 14 bus system.

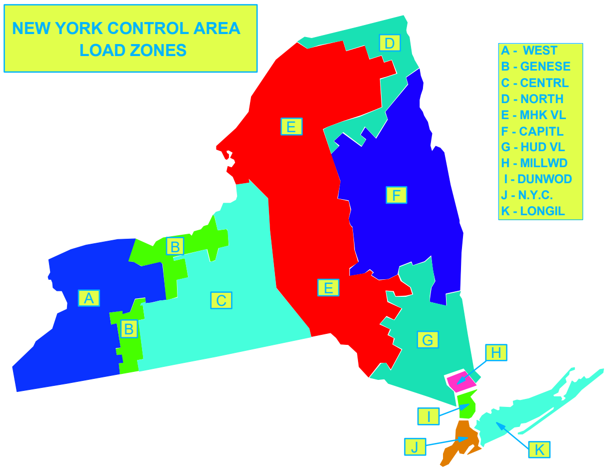

Figure 7: NYISO map of the 11 control area load zones.

Due to the lack of real-time system measurements and states, we utilize the load data of New York (NY) state provided by the NYISO [1, 5]. Specifically, we use the five minute load data of 13 months, year 2015 and January 2016, of NYISO to generate the system states. First we connect, in an ascending order, each region of NYISO as shown in Figure 7 with every load bus of IEEE 14 system (e.g., region A-WEST to bus 2, region B-GENESE to bus 3, etc.). Then, we normalize the NYISO active load to the initial active and reactive IEEE 14 load bus data assuming a constant power factor. Thus, for each five minute interval, there exists active and reactive load information for the IEEE 14 bus system. The ratio of the derived total active/reactive load to the IEEE 14 bus initial total active/reactive load is used as the rate which the generators increase their output power. In a real system, the latter can be adjusted based on the operators knowledge of forecasting and scheduling generation algorithms [25]. After this step, we use the data to calculate the system state $\boldsymbol{x}$ via the AC power flow Newton-Raphson function. Finally, based on the Jacobian matrix of the IEEE 14 bus system $\boldsymbol{J}_h$, we calculate for each interval the measurement vectors $\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x})$ utilized in our study.

5.3 Results

The proposed procedures are evaluated with simulations performed on the developed IEEE 14 test system. The system has 27 state variables which are the voltage angles and voltage magnitudes of the buses, with the first bus angle chosen as the reference one. Each bus and each transmission line are assumed to be equipped with RTUs which measure the power injection and power flow.

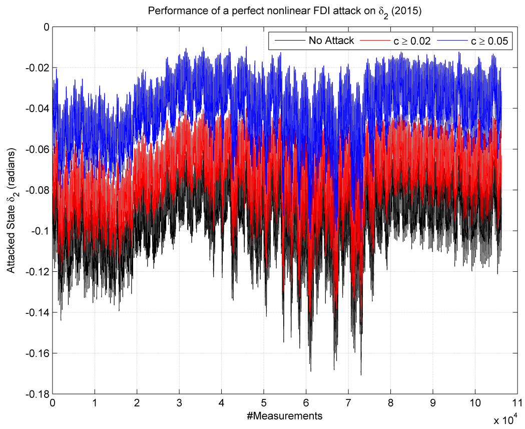

Figure 8: Performance of perfect AC FDI attack on δ_2 of IEEE 14 system for 2015 NYISO data.

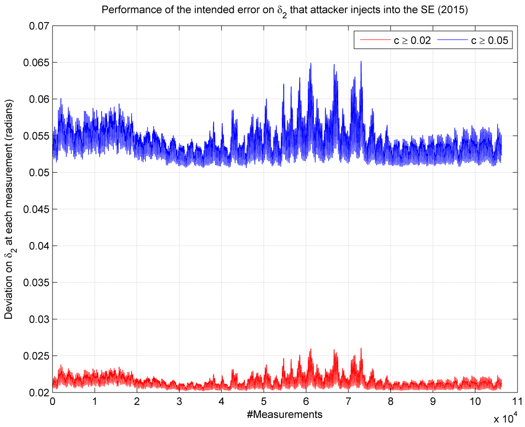

Figure 9: Performance of the indented error c on δ_2 of IEEE 14 system for 2015 NYISO data (perfect AC FDI attack).

For the IEEE 14 bus system we assume that the attacker aims to compromise SE for the voltage phase angle on bus 2, δ_2. To inject the FDI attack error c_2 into the true state value δ_2, the attacker requires (based in (2) – (5)) to modify the measurements $\boldsymbol{z}^\star \in \mathcal{S}_2$, where $\mathcal{S}_2 = \{1, 3, 4, 5\}$. The IEEE 14 benchmark system with the compromised RTU sensors is shown in Figure 6.

The effect of the developed FDI attack is first examined in the scenario which the adversary accurately estimates the system states $\boldsymbol{x}^\star \in \mathcal{S}_2$ (perfect attack) and aims to deviate the solution δ_2 from the true value by $c_2 \geq 0.02$ and $c_2 \geq 0.05$. The AC SE is performed via the Dishonest Gauss Newton method and the simulation results for 2015 NYISO data are illustrated in Figure 8. The graph indicates the state variable δ_2 before and after the attacks. Figure 9 presents the performance of the indented error on δ_2 that the attacker aims to inject to the SE via the falsified measure-

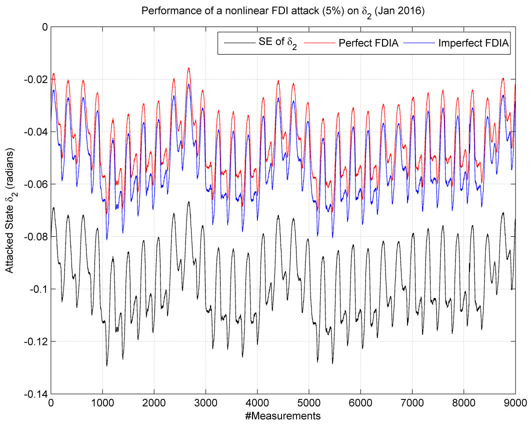

Figure 10: Performance of imperfect AC FDI attack on δ_2 of IEEE 14 system for January 2016 NYISO data.

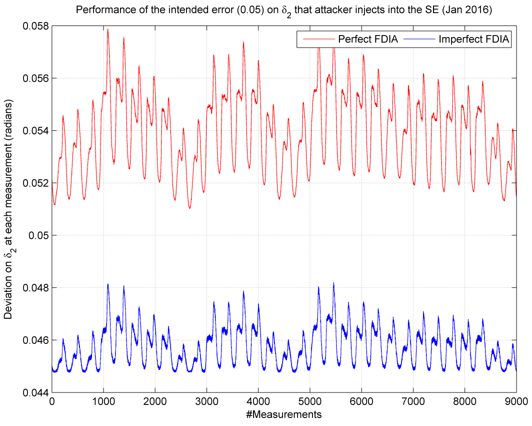

Figure 11: Performance of the indented error c on δ_2 of IEEE 14 system for January 2016 NYISO data (imperfect AC FDI attack).

ments $\boldsymbol{z}^\star \in \mathcal{S}_2$. We can see that the SE solution successfully deviates at the intended amount of c_2.

For the imperfect FDI attack scenario, the firmware modifications of the flash memory data of RTU B are applied to the embedded device. The reported RTU measurement information $\boldsymbol{z_a}$ is transferred to the SE simulation module. Due to the implementation and noise variation errors of this process, a percentage of inaccuracy is permitted in the attacker's acquired SE solution. The results shown in Figure 10 and Figure 11 for January 2016 demonstrate that the adversary can no longer achieve a perfect attack as the SE solution cannot reach the intended deviation of 5%. However, the adversary is able to change the estimated state variables while the attack remains hidden and not detected by the BDD function. The proposed imperfect FDI attack model, i.e., the problem of identifying the values of the relaxing error $\boldsymbol{\tau}$ subject to the firmware data integrity equality constraint (CRC-16), is presented in Figure 12. The active power injection results indicate that for changing the SE state δ_2 by 5%, the attack vector $\boldsymbol{z}_a^* = \boldsymbol{z}^* + \boldsymbol{a}$ satisfies (18) when $\bar{\boldsymbol{\lambda}} = 0.30567$ for the NYISO data of January 2016.

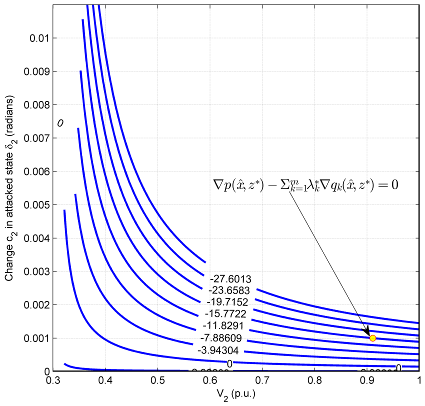

Figure 12: State variables at bus 2 subject to the constrained optimization problem of Equation 18 for falsified active power injection measurement vector (January 2016).

6. CONCLUDING REMARKS

This research work demonstrates the payload delivery process of FDI attacks by modifying the firmware of RTUs. The grid impact of such attacks is examined through simulations that demonstrate the violation of correct SE operation. Future work includes delivering attacks that can also change breakers status such that the operator validates a fictitious topology or does not consider a real topology change. Also, we plan to develop a hardware-in-the-loop testbed in order to properly model the complex behavior of the grid.

Acknowledgment

This work was supported by the NYU Abu Dhabi Global PhD Fellowship program.

7. REFERENCES

[1] New York Independent System Operator (NYISO). [Online]: http://www.nyiso.com/.

[2] IEEE Standard Test Access Port and Boundary Scan Architecture. *IEEE Std 1149.1-2001*, 2001.

[3] R. Bobba, K. Rogers, et al. Detecting false data injection attacks on dc state estimation. In *Preprints of the First Workshop on Secure Control Systems, CPSWEEK*, 2010.

[4] M. Breeuwsma, M. De Jongh, et al. Forensic data recovery from flash memory. *Small Scale Digital Device Forensics Journal*, 1(1):1–17, 2007.

[5] G. Chaojun, P. Jirutitijaroen, and M. Motani. Detecting false data injection attacks in ac state estimation. *IEEE Transactions on Smart Grid*, 6(5):2476–2483, 2015.

[6] S. Clements and H. Kirkham. Cyber-security considerations for the smart grid. In *IEEE PES General Meeting*, pages 1–5, July 2010.

[7] A. Costin. *Large Scale Security Analysis of Embedded Devices' Firmware*. PhD thesis, EURECOM/TelecomParisTech, 2015.

[8] A. Costin, J. Zaddach, et al. A large-scale analysis of the security of embedded firmwares. In *23rd USENIX*

Security Symposium (USENIX Security 14), pages 95–110, 2014.

[9] CRitical Infrastructure Security AnaLysIS. *Crisalis Project EU, Deliverable D2.2 Final Requirement Definition.*

[10] A. Cui, M. Costello, and S. Stolfo. When firmware modifications attack: A case study of embedded exploitation. In *NDSS*, 2013.

[11] N. Falliere, L. Murchu, and E. Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5:6, 2011.

[12] M. Govindarasu, A. Hann, and P. Sauer. Cyber-physical systems security for smart grid. *Future Grid Initiative White Paper, PSERC, Feb*, 2012.

[13] G. Hug and J. Giampapa. Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks. *Smart Grid, IEEE Transactions on*, 3:1362–1370, 2012.

[14] J. Kim and L. Tong. On topology attack of a smart grid: Undetectable attacks and countermeasures. *IEEE Journal on Selected Areas in Communications*, 31(7):1294–1305, 2013.

[15] J. Kim, L. Tong, and R. Thomas. Subspace methods for data attack on state estimation: A data driven approach. *Signal Processing, IEEE Transactions on*, 63(5):1102–1114, 2015.

[16] C. Konstantinou and M. Maniatakos. Impact of firmware modification attacks on power systems field devices. In *2015 IEEE International Conference on Smart Grid Communications*, pages 283–288, 2015.

[17] C. Konstantinou, M. Sazos, and M. Maniatakos. Attacking the smart grid using public information. In *2016 17th Latin-American Test Symposium (LATS)*, pages 105–110.

[18] G. Liang, J. Zhao, et al. A review of false data injection attacks against modern power systems. *IEEE Transactions on Smart Grid*, PP(99):1–1, 2016.

[19] X. Liu, Z. Bao, et al. Modeling of local false data injection attacks with reduced network information. *Smart Grid, IEEE Transactions on*, 6(4):1686–1696, 2015.

[20] X. Liu, Z. Li, and Z. Li. Impacts of bad data on the pmu based line outage detection. *arXiv preprint arXiv:1502.04236*, 2015.

[21] X. Liu, P. Zhu, et al. A collaborative intrusion detection mechanism against false data injection attack in advanced metering infrastructure. *Smart Grid, IEEE Transactions on*, 6(5):2435–2443, 2015.

[22] Y. Liu, P. Ning, and M. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 14(1), 2011.

[23] J. Michel. From NAND chip to files. [Online]: http://blog.j-michel.org/post/86992432269/from-nand-chip-to-files.

[24] A. Monticelli. *State Estimation in Electric Power Systems: A Generalized Approach.* Springer US, 1999.

[25] M. Nejati, N. Amjady, and H. Zareipour. A new stochastic search technique combined with scenario approach for dynamic state estimation of power systems. *IEEE Transactions on Power Systems*, 27(4):2093–2105, 2012.

[26] D. Peck and D. Peterson. Leveraging ethernet card vulnerabilities in field devices. In *SCADA Security Scientific Symposium*, pages 1–19, 2009.

[27] M. Rahman, E. Al-Shaer, and R. Kavasseri. Impact analysis of topology poisoning attacks on economic operation of the smart power grid. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 649–659. IEEE, 2014.

[28] M. Rahman and H. Mohsenian-Rad. False data injection attacks with incomplete information against smart power grids. In *Global Communications Conference, IEEE*, pages 3153–3158, 2012.

[29] M. Rahman, H. Mohsenian-Rad, et al. False data injection attacks against nonlinear state estimation in smart power grids. *IEEE PES General Metting*, 2013.

[30] L. Sollecito. Smart grid, the road ahead. *GE Digital Energy, Protection and Control Journal*, 8(8):15–19, 2009.

[31] A. Tal. Two flash technologies compared: Nor vs nand. *White Paper of M-SYstems*, 2002.

[32] A. Teixeira, S. Amin, et al. Cyber-security analysis of state estimators in electric power systems. In *IEEE Conference on Decision and Control*, 2010.

[33] X. Wang, C. Konstantinou, et al. Malicious firmware detection with hardware performance counters. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2016.

[34] J. Zaddach and A. Costin. Embedded devices security and firmware reverse engineering. *Black-Hat USA*, 2013.

[35] K. Zetter. *Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid*. Wired, 2016.

[36] J. Zhao, G. Zhang, et al. Forecasting-aided imperfect false data injection attacks against power system nonlinear state estimation. *IEEE Transactions on Smart Grid*, 7(1):6–8, 2016.