Defeating Network Node Subversion on SCADA Systems Using Probabilistic Packet Observation

Thomas Richard McEvoy[2] and Stephen D. Wolthusen[1,2]
[1] Norwegian Information Security Laboratory,
Department of Computer Science,
Gjøvik University College, Norway
[2] Information Security Group,
Department of Mathematics,
Royal Holloway, University of London, UK
T.R.McEvoy@rhul.ac.uk stephen.wolthusen@rhul.ac.uk*

Abstract Supervisory control and data acquisition (SCADA) systems form a vital part of the critical infrastructure. Such systems have been subject to sophisticated and persistent attacks which are executed by processes under adversary supervision. Such attacks may be detected using inconsistencies in sensor readings or estimated behavior of the plant. However, to locate and eliminate malicious "agents" in networks, novel protocols are required to observe messaging behavior. In this paper, we propose a novel network protocol for SCADA systems which, for low computational cost, permits discovery and elimination of subverted nodes utilizing techniques related to probabilistic packet marking. We discuss its advantages over earlier work in this area, calculate message complexity requirements for detection and outline its resilience to various attack strategies .

1 Introduction

Supervisory control and data acquistion (SCADA) systems are ubiquitous to critical infrastructures. Such systems have been publicly targeted with relatively sophisticated and persistent attacks [1] where intrusion aims are not confined to creating short-lived, albeit disastorous, effects, but are based on medium to long term strategies. These so-called *advanced, persistent threats*(APT) make use of increasingly sophisticated concealment techniques which require advances in detection methods to uncover them.

The concept of utilising network protocols to aid intrusion detection on SCADA systems was first proposed in [2]. It is assumed an operator has a model of a control system and can detect inconsistencies in plant signals [3]. This method indicates a process is under attack, but is insufficient to detect which network nodes, or process control units have been subverted. By combining packet tracing methods [4,5] with this knowledge of control system responses, an operator can reason over which network nodes, or control units, are free of adversary control and initiate appropriate recovery strategies.

In this paper, we propose a simpler, but more robust approach. Setting aside the consideration of whether control units are subverted, we concentrate on observing (probabilistically) packet behavior along network routes. This enables us to detect efficiently the node at which packet contents are manipulated. If no such manipulation takes place and the control signal is still anomalous, we can assume that the control unit is subject to adversary subversion.

In response to anomalous conditions, network nodes probabilistically create copies of packets and send these to the operator. The operator can check these copies against other copies of the same packet, including the original, to determine not only the routes followed by packets through the network (which may also be subject to adversary manipulation), but also whether packet contents have been altered along a given route. We assume that any subverted nodes on the network can perfectly forge any message with regard to any shared secrets (and are hence immune to protocol based detection) and that any node may be subverted (with the exception of operator nodes). Hence we require a protocol which is relient to adversary interference and resists such forgery. We prove our protocol using both graph-theoretic and algebraic proofs [6] and demonstrate its efficiency in terms of node detection and network recovery.

In section 2, we provide related reading. In sections 3 and 4, we outline our problem and approach. We provide details of our probabilistic observation protocol in section 5, including proofs and a discussion of message complexity. We conclude with a discussion and an outline of future work in section 6.

2 Related Work

SCADA systems have been subject (publicly) to attacks which are not just intended to achieve a short-lived goal, but have a "stay resident" capability which incorporates the use of concealment techniques at both network and operating system level with the capability to manipulate process signals on control units [1]. Such attacks have been discussed at length in the literature [7,8]. Detective strategies based on protocol or signal analysis by themselves have been shown to have weaknesses in uncovering such attacks [9], particularly in the face of an *overwriting* adversary with the ability to subvert network nodes in the system [3]. Hence conjoint reasoning over both communication and control functionality is necessary [2], taking into account system context (such as related states in other control units, which may not have been subverted) [10]. This can be split into the analysis of the integrity of network nodes on the one hand and the analysis of the integrity of process control units on the other. Techniques related to IP traceback [11,12] have been proposed to help solve this kind of problem. In the original paper, this required demonstrating the existence of independent routes and then considering their consistency in terms of the model to determine their integrity. However, this approach relies on the knowledge of routes and system outputs in a fixed topology. The behavior of packets *en route* is not considered, nor is adversary ability to re-route messages, and it is not

clear that input signals are consistently delivered (except by means of complex inductive reasoning).

We argue that we need to simplify the reasoning process involved by observing packets behavior during communication. We also avoid the implicit assumption that the subverted nodes which manipulate output data are the same as those manipulating input data. If we can observe inconsistent data packets and the route they follow, the use of state estimation techniques [13], can be confined (except for initial detection) to consideration of the integrity of control systems.

We base our approach on IP Traceback methods which were originally devised to deal with denial of service attacks, with single and multiple origin points and we note immediately that , although the study of traceback mechanisms was motivated by denial-of-service attacks, it was recognised early on that the use of real -time traceback mechanisms could be used for other applications as well – for example, the analysis of legitimate traffic in a network, congestion control, robust routing algorithms, or dynamic network reconfiguration [14].

During DOS attacks, it was assumed that the following system characteristics held [5] –

1. Packets may be lost or reordered,
2. Attackers send numerous packets,
3. The route between attacker and victim is fairly stable,
4. Network nodes are both CPU and memory limited, and
5. Network nodes are not widely compromised.
6. An attacker may generate any packet,
7. Multiple attackers may conspire,
8. Attackers may be aware they are being traced,
9. A compromised router can overwrite any upstream information

Clearly, assumption 2 is not relevant with regard to packet manipulation in a SCADA environment, but other assumptions continue to hold and are augmented in the model we use – see section 3 and [3].

Various approaches to IP tracing were proposed with differing payoffs in the context of denial of service attacks. Examples included link testing (tracing the source of the attack recursively through the network) which could be slow under DOS conditions, router logging which clearly creates storage and processing overheads, behavioral monitoring looking for patterns suggest attack behavior (which requires stateful monitoring) , or packet-based traceback (marking packets with information about intermediate routers) or by creating information packets separate from data packets. The last two methods clearly creating additional overheads, e.g., due to variable length headers or having additional messages to process. Probabilistic packet marking was proposed both to minimise message overheads and router logging requirements, but at the expense of introducing uncertainly due to the probabilistic sampling of the flow path and the ability of the attacker to inject false packets [12]. Other hybrid

methods were also introduced to cut on message and storage overheads further, without solving the injection problem [4]. Our approach is based on the strategy proposed for the iTrace protocol and other techniques discussed in [4], a paper which also demonstrates relevant practical implementation techniques and which, in particular, allows us to solve the injection problem by utilising a dynamic hashing and key release system for both packet identity and contents.

Our algebraic specification is derived from the approach proposed in [2] as well as work by [6]. This offers a means of designing and testing protocol behavior abstracted from the details of construction, which, in SCADA environments in particular, may differ depending on the technology currently employed - see section 5. It is also an economical means of presenting certain types of graph-theoretic proofs relating to dynamic multigraphs which we use to model the SCADA network.

3 Problem

We retain the assumptions, with the single exception, noted in Section 2. In particular, we assume an adversary can subvert any network node, but not exercise wide control over the system, and that subverted nodes (or *agents*) are under adversary command and control meaning that their functionality can be arbitrarily rewritten by the adversary. Agents may therefore cooperate in deception under adversary control. Agents may also conditionally drop, delay or re-order any messages (in addition to such behavior being intrinsic to network operations) or communicate message information to the adversary. In addition, such agents are assumed to be able to forge any message which relies on shared secrets. Without adversary intervention, agents may selectively alter inputs (command parameters) to the system and outputs (process signals) for a limited range of control system behavior [3]. However, agents are not fully autonomous, adversary intervention is required to make changes to their functionality.

Agents, therefore, can either undermine network communications – in particular, they can manipulate process units by sending false instructions which mimicking "business as usual" to system operators, or else they can mimick negative conditions which cause operators to enter inappropriate commands. In this paper, we limit ourselves to considering an attack on network communications and excluding attacks on process units. Our problem is to detect and locate potential agents which are network nodes that interfere with communications and to bypass them in a way which allows the SCADA system to continue functioning, preferably without disrupting production.

4 Approach

We assume an initial detection event will initiate the use of the proposed *probabilistic packet observation protocol*. On detection, network nodes participating in the packet observation protocol will probabilistically take a copy of certain

packets and hash them using a time released keys (to avoid adversary forgery). The resulting packet is sent to the operator and may be compared with the original packet and any subsequent copies. The protocol applies to both input and output messages. This enables the discovery of agent nodes along communication routes (assuming the control function itself is not subverted and participates in the protocol). Subsequently, dynamic routing techniques using redundant routes (whose existence we assume) may enable the operator to bypass the subverted nodes. We leave the problem of determining which control units are subverted to another paper.

5 Probabilistic Observation Protocol

We describe a SCADA network which we model as a directed multigraph. This enables us to uniquely identify routes which are non-cyclical paths between the operator and the network's control units. Subsequently, we define operations over this multigraph using an algebraic representation based on the π-calculus. We give a simple example of the nature of an adversary attack using this approach. We formally define the observation protocol and, assuming an initial detection event, show how an operator may discover a subverted node and also recover from attack using dynamic routing techniques making use of redundant channels. The complexity requirements are modelled and the hashing and key release system used to protect the protocol from subverstion is outlined.

5.1 SCADA Systems

A SCADA (supervisory control and data acquisition) system consists of an operations center, manned by one or more operators who use an HMI (human-machine interface) to view the state of the system which is determined by signals from a set of control units and associated sensors, which may be redundant. The operators in response to changes in the system state or as the result of management decisions on system output send signals to control units which alter the operational parameters of those units. In fact, large parts of this process may be automated or under the control of expert systems. Such control is "supervisory" because the control units take responsibility for enacting changes and subsequent real-time local adjustments to actuators (which operate the physical system under control). All signals are communicated by network or other (e.g., satellite, point to point, PSTN, broadband, wireless) communications routes. The history of the system is recorded in a database ("the historian") and may also be communicated to management information systems on corporate networks in near real-time.

5.2 Initial Network Model of a SCADA System

We define the initial network model of a SCADA system and outline some basic concepts which we subsequently use.

Definition 1. ***Initial Network Model*** *Let G be a SCADA network. We represent G by a directed multigraph $\overrightarrow{G}$. Let $\overrightarrow{G} = V(\overrightarrow{E})$ be a digraph such that each $v \in V$ represents a network node which we also label a* process *and each $\overrightarrow{e} \in \overrightarrow{E}$ is a directed edge which represents* potential communication *means between network nodes (i.e., processes) $u, v \in V$ which we also label a* channel*. We call $\overrightarrow{G}$ the* initial network model *of G. We assume that the topology of $\overrightarrow{G}$ is fixed w.r.t channels and processes.*

We note, at this point, that we have not yet defined operations $\overrightarrow{G}$. For example, while all channels are potential means of communication, we have not specified which ones are used, or in what order, or under what conditions. We define potential *routes* and show that routes are uniquely identiable.

Definition 2. ***Potential Route*** *Let $\overrightarrow{G}$ be an initial network model as before. Let R be a non-empty, non-cyclical directed path, consisting of directed edges which are channels between two vertices which are processes, say U and W, not necessarily adjacent (i.e., there may be intervening vertices $V_1, V_2, \ldots, V_n$ between U and W which are joined by the edges in R), then we call R a* potential route *between U and V.*

Lemma 1. *Let $\overrightarrow{G}$ be an initial network model. Let R be a potential route in $\overrightarrow{G}$, then R is uniquely identifiable.*

Proof. Let R be a route between U and W. The route R is not an empty path since, by definition, it consists of, at least, one directed edge or channel. If we label the channels of R uniquely, then $L(R)$ the set of labels formed from R is unique. We can see this immediately because if we select another route S between U and W and consider whether the edges have been labelled, then if they are already labelled, they are the labels belonging to R and $S = R$, else if edge is not labelled, the $S \neq R$.

Let $\mathcal{R}_{U,W}$ be the set of all routes between U and W. Where $\mathcal{R} = \emptyset$ we say that U and W are *disjoint* and no direct communication potential communication exists between them. Where $|\mathcal{R}| > 1$ we say that potential communication between U and W is *redundant*. Where two routes, say S and R, share nodes or edges such that $L(S) \cap L(R) \neq \emptyset$, we say that the routes S and R are *dependent*. Where $L(S) \cap L(R) = \emptyset$, we say that the routes S and R are *independent*. When we include the vertices of $\mathcal{R}$ together with its edges, we see that it forms a subgraph $\overrightarrow{G}_i \subseteq \overrightarrow{G}$ which consists of all potential communication channels (hence routes) between U and W. We note that distinct subgraphs, say $\overrightarrow{G}_i$ and $\overrightarrow{G}_j$, are not necessarily independent. But we also note that distinct dependent partitions of a subgraph $\mathcal{R}$ may be independent of each other.

We assume the existence of a network operator [1] which we represent by a vertex U and one or more control units which we represent by vertices W_i belonging to a set $\mathcal{W}$. We assume, during normal operations, that U sends and

[1] We simplify the notion of an operational center by referring to a single operator rather than multiple operators working as a group to control the system

receives message packets from each W_i and that all other vertices $\mathcal{V}$ act as forwarding links for these messages, hence only U and each $W_i \in \mathcal{W}$ have bi-directional channels with their adjacent vertices. We assume that the set $\mathcal{W}$ contains redundant sensors or control units which the operator can use for intrusion detection purposes by modelling contexts[3]. More importantly, we assume that potential communication between U and $\mathcal{W}$ is redundant for each $W_i \in \mathcal{W}$ [15] and that redundant channels are not necessarily used during normal network operations, hence they are available for other purposes such as contingent actions by the operator.

Defining Operations over the Initial Network Model We subsequently model operations, i.e., the *network system*, over the initial network model using an algebraic representation based on the π-calculus [6]. For example, let U, V be a process and C be a control system, we define C by –

$$\begin{aligned} V &:= \nu\ z\ y(u_{\langle \bar{c} \rangle}).f(u) \rightarrow \bar{x}z_{\langle \bar{c} \rangle} \oplus \lambda \\ U &:= \nu\ z[TRUE]\bar{y}a_{\langle \bar{c} \rangle} + x(z_{\langle \bar{c} \rangle}) \\ C &:= !U|!V \end{aligned} \tag{1}$$

where each phrase is called a capability. Period . indicates an ordered sequence, $+$ an arbitrary sequence, $\oplus$ represents a choice, $f() \rightarrow \nu z, z, \bar{x}z$ is a function which creates/updates a name and may send it, $\bar{x}z$ indicates send the name z by channel x, $x(z)$ means receive the name z by channel x and (purely for convenience) the tuple $\langle \bar{c} \rangle$ is a set of characteristics associated with a name (which are defined on use). $[TRUE]$ represents some condition which must be fulfilled before a capability is exercised. Processes may replicate $!P$ on exhausting their functional capabilities. We freely use $\coprod$ and $\sum$ to deal with multiple concurrent processes $P|Q|R|\ldots$ and sums of capabilities $\bar{x}_1 a + \bar{x}_2 a + \ldots$. Labels (e.g. λ) are used to indicate some inaction, which may or may not be observable (e.g., decision making, message loss). See [2] for a fuller explanation of this approach.

5.3 Attack Model

An attack is carried out by an adversary node, which does not form part of the initial network model, but is assumed to be able to address the network during operations using network nodes which are open to external network communications. The adversary sends a malicious message to a network node which, if it succeeds, results in the capabilities of the node being arbitrarily overwritten by the adversary and the node acting as an agent of adversary [3]. Overwriting the model introduces functions or alters the conditions under which a process works. For example, the processes in equation 2 might be re-written -

$$
\begin{aligned}
V &:= \nu\ z\ y(u_{\langle \bar{c} \rangle}).f(u) \rightarrow \bar{x}z_{\langle \bar{c} \rangle} \oplus \lambda \\
U &:= \nu\ z([z < 3]\bar{y}a_{\langle \bar{c} \rangle} \oplus \lambda) + x(z_{\langle \bar{c} \rangle}) \\
S &:= !U|!V
\end{aligned} \tag{2}
$$

Re-writing could also include introducing fucntions making use of normally redundant channels to re-route messages and conditional message delay, drop or manipulation as well as communicating information to the adversary.

5.4 Algebraic Protocol Definition

We define a probabilistic packet observation protocol which enables us to locate and, using dynamic network re-routing, elide an agent node from the SCADA system, pending investigation and recovery.

Let V be any network node which is not an operator. We can define the action of V algebraically by –

$$
\begin{aligned}
V :=& \nu\ \tilde{akt}\ \sum x_i(u_{\langle r,f,d \rangle}).Observe(f,u) \rightarrow (\bar{x}w_{\langle r,0,d,a \rangle}.\mathbf{0} \oplus \lambda) \\
& .NewKey(t, \tilde{k}) \rightarrow (\bar{x}{k^{t-\delta}}_{\langle r \rangle}.\mathbf{0} \oplus \lambda) \\
& + \sum \bar{x}_j(u_{\langle r,f \rangle})|!V
\end{aligned} \tag{3}
$$

– where a is the node address, f is the packet flag, d is the packet identity, r is the address to which the packet is routed, k is a key from the vector of keys $\tilde{k}$. x_i and x_j are sets of channels over which we sum the send and receive capabilities.

Initially, for all packets, we set a flag $f = 0$ and observation is a null event. As the result of a potential detection event, the operator initiates the protocol by setting $f = 1$. On receiving flagged packets, whether inputs or outputs, routers determine based on some probability q to "observe" identified packets. The $Observe()$ function copies the packet and hashes its data contents, the packet identity and its own IP address which it stores as a characteristic. Then it sends the copied packet to the operator. We note the potential for packets to be dropped by λ.

Finally, after a set period of time t, routers publish the current key to the operator and generate a new key in such a way that subsequent (and previous, unrevealed) keys cannot be predicted – see section 5.6. This action is controlled by the function $NewK()$ and it should be noted that the names used by this function for keys and time are restricted to the scope of the function (i.e., they are local variables).

On receiving packets whose identities match, the operator can compare the data contents of copied packets with each other and the original, noting its origin and potential routing (based on the initial network model $\overrightarrow{G}$). Hence where

packet contents differ due to adversary manipulation, it can be determined from several packets which routes may have been subverted by the adversary, simply by observing the hashed address of observed packets and whether or not they have been manipulated. A process of elimination over each route leads to a determination of which nodes have been subverted on which routes.

Once sufficient subverted routes are identified (i.e., such that they can effectively be bypassed using redundant channels), the operator can (other than for further detective purposes) re-direct traffic away from affected routes using a dynamic re-routing protocol [2]. Here the existence of redundant channels which we have assumed enables the operator to economically restore control of communication channels without disrupting production on the SCADA system. Full investigation and recovery procedures can follow. One of the implications of this research is that SCADA systems should incorporate such redundancy as a matter of policy.

5.5 Complexity Requirements

We consider a subgraph $\mathcal{R} \subseteq \overrightarrow{G}$ of routes between an operator node U and a control unit W_i. Assuming that we can observe mismatched packets originating from W_i via $\mathcal{R}$, how many observation packets on average need to be generated for the protocol to enable the detection of the complete set of subverted nodes?

We initially consider the case of a single route R_i for detecting the manipulation of signals originating from the control unit[3]. We designate the nodes directly adjacent to the operator S and the node which is the control unit T and we assume that $T \neq S$ (that there is, at least, one node V between U and W). Let $<$ be the relation "follows" and $\leq$ the relation "follows, or is equal to" in order of communication, then, for outputs, $S < T$ and we call the set of nodes from S to T the *observation range* of R_i. Let the number of nodes in the initial observation range (inclusive of S and T) be n. We assume the probability p of observing a packet k as it traverses each node from T to S is uniform.

Since the node S will produce a manipulated packet (because there is a mismatch), we overload the node label S to designate the set of nodes which communicate invalid packets, likewise T [4] is used to designate the set of nodes producing valid packets. We require to find out how many nodes are in S and how many are in T since the agent node is the first node S in order of communication. To do this, we have to gather packet information for all nodes along R_i. This problem is related to the card collector's problem which is encountered in IP traceback for detecting DOS attacks [12], but with strong efficiencies in that, if we observe a node $V_i < T$ and V_i is a member $t \in T$ of valid nodes then all nodes $V_i < V_{i-1} < V_{i-2} < \ldots < T$ are likewise valid; similarly, if $V_i = s \in S$

[2] This is achieved using another protocol which we do not define here. But, for example, TCP/IP and similar protocols have fields for setting required routes.

[3] Applying the equivalent argument to input signals originating with the operator is left as an exercise for the reader.

[4] We initially assume that the control unit will produce valid packets.

all subsequent nodes are assumed to be in S. Hence we can designate the node V_i to be the new endpoint for the search either as S' or T'. This means that the next valid observation operation will take place along the nodes inside the new *observation range*.

Let X_i be a random variable which indicates the number of packets which need to be sent to obtain a valid observation (inside the observation range) where $i = 1, 2, 3, \ldots$ then –

$$\sum_i E(X_i) = \frac{n}{n - d_0} + \frac{n}{n - (d_0 + d_1)} + \frac{n}{n - (d_0 + d_1 + d_2)} + \ldots + \frac{n}{n - (\sum_j d_j)} \tag{4}$$

– since each $E(X_i)$ is a geometrically distributed random variable which represents the average number of packets required to make a successful observation in the observation range which shrinks randomly (by a distance of d_i hops for which a determination has been made) on each observation and $\sum_j d_j = n - 1$ and $d_0 = 2$ (since we already know that the nearest node S produces malicious packets and the furthest node T produces a valid packet).

The sum $2 + \sum_j d_j = n - 1$ where is $j > 1$ implies that we have positive integer solutions $d_1 + d_2 + \ldots + d_r = n - 3$ for each $r \in \{1..n - 3\}$. Hence there are $\sum_{r=1}^{n-3} \binom{n-4}{r-1}$ possible sums each of which is equally likely to occur.

Let Y be a random variable which indicates the average total number of packets required to take complete observation of the set of nodes R, then we have –

$$E(Y) = \sum_{k=1}^{\sum_{r=1}^{n-3} \binom{n-4}{r-1}} \frac{1}{\sum_{r=1}^{n-3} \binom{n-4}{r-1}} \left[\sum_i E(X_i) \right]_k \tag{5}$$

But, it should be noted – considering detection along all routes belonging to a subgraph $\mathcal{R}$ – that further efficiencies are gained , where the route R_i has several other routes $R_j, R_k, \ldots$ with which it shares nodes (i.e., the routes are dependent) since the validity or otherwise of a single node may determine node validity (or otherwise) along several routes. It follows that for a subgraph $\mathcal{R} \subseteq \overrightarrow{G}$ with m routes, the largest average number of packets $\left[\sum_{l=1}^{m} E(Y)_l \right]_{max}$ needed for observation occur when each route is independent.

These findings have interesting implications for the design of SCADA systems since route independence allows the operator to bypass subverted routes, but route dependence enables quicker discovery of subverted nodes. This suggests a strategy of subdividing each subgraph $\mathcal{R}$ into dependent subpartitions $R_1, R_2, \ldots$ which are independent of each other, balancing efficiency of discovery against independence of potentially redundant channels. This strategy may also make it more difficult for the adversary to efficiently subvert sufficient nodes in each subpartition. The analysis of suitable topologies based on aiding intrusion detection and response using network protocols widens this area as a new field of study.

5.6 Time-Sequenced Key Value Release

A network node V generates an initial sequence of keys $\tilde{k}$ and subsequently generates a fresh key based on some nonce using a randomly selected key from its key chain, using a suitable one-way function. The fresh key becomes part of its key chain, randomly replacing another key, and is used for packet marking from that point. Depending on some time period, or set of discrete events (e.g., observing n packets), the network node generates another fresh key by hashing a randomly selected key from its key chain with the current key and replacing one of the keys, again at random. It subsequently publishes the previously used key to the operator. Subsequent keys are released in the order in which they are used. This scheme is similar to the one proposed in [4]. This approach is resilient to adversary attack because insufficient time exists to guess keys and both the packet identity and its contents are hidden from the attacker, hence packets cannot be directly manipulated and only arbitrarily delayed, dropped, re-routed or vandalized. However, these forms of attack only delay rather than disrupt discovery. In some cases, they may accelerate it by providing further packet-based anomalies.

6 Conclusion and Future Work

Advanced, persistent threats constitute a "clear and present danger" to critical infrastructure systems. In this paper, we propose a novel protocol for use on SCADA systems which enables the operator to observe packet behavior with regard to data and routing which will allow us to locate and counter such subverted nodes. The protocol is based on previous work in IP Traceback, orginally used for detection and prevention of DOS attacks.

The protocol represents a low cost computational solution to determining the integrity of channels on SCADA systems. If combined with the ability to route packets dynamically, this approach represents a cost effective approach to defeating node based attacks from an engineering point of view by making use of system redundancy.

Future work will consist of determining, for various SCADA protocols, practical implementations of this approach and testing them under simulation with regards to performance.Further work remains to be done on other related protocols or variants and extending the work to other kinds of network. Furthermore, the study of network topology in relation to resilience and intrusion detection itself represents a potentially fruitful topic for research. Finally, this paper does not consider the problem of determining, based on reliable or, at least, partially reliable channels, where control units on the system may have been directly attack either by sabotage, or remotely. This work, based on developing state estimation techniques, for contextual intrusion detection remains to be explored.

References

1. Chen, T.M., Abu-Nimeh, S.: Lessons from Stuxnet. Computer **44**(4) (April 2011) 91 –93
2. McEvoy, T.R., Wolthusen, S.: A Plant-Wide Industrial Process Control Security Problem. In Butts, J., Shenoi, S., eds.: Critical Infrastructure Protection: Proceedings of the First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection. International Federation for Information Processing Advances in Information and Communication Technology, Hanover, NH, USA, Springer-Verlag (March 2011) (in press).
3. McEvoy, T.R., Wolthusen, S.: A Formal Adversary Capability Model for SCADA Environments. Proceedings of the 5th International Workshop on Critical Information Infrastructures Security, CRITIS 2010 **()** (Sep. 2010) ()
4. Song, D.X., Perrig, A.: Advanced and Authenticated Marking Schemes for IP Traceback. In: INFOCOM 2001: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 2. (2001) 878 –886
5. Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Network Support for IP Traceback. IEEE/ACM Transactions on Networking **9**(3) (June 2001) 226 –237
6. Sangiorgi, D., Walker, D.: pi-Calculus: A Theory of Mobile Processes. Cambridge University Press, New York, NY, USA (2001)
7. Verba, J.; Milvich, M.: Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SCADA IDS). In: Technologies for Homeland Security, IEEE Conference on. (2008) 469 – 473
8. Gamez, D., Nadjm-tehrani, S., Bigham, J., Balducelli, C., Burbeck, K., Chyssler, T.: Safeguarding Critical Infrastructures. In: Dependable Computing Systems: Paradigms, Performance Issues, and Applications],, Wiley[Imprint], Inc. (2000)
9. Svendsen, N., Wolthusen, S.: Using Physical Models for Anomaly Detection in Control Systems. In Palmer, C., Shenoi, S., eds.: Critical Infrastructure Protection III. Volume 311 of IFIP Advances in Information and Communication Technology. Springer Boston (2009) 139–149
10. Sheng, S., Chan, W., Li, K., Xianzhong, D., Xiangjun, Z.: Context Information-based Cyber Security Defense of Protection System. IEEE Transactions on Power Delivery **22**(3) (Jul 2007) 1477 –1481
11. Al-Duwairi, B., Govindarasu, M.: Novel Hybrid Schemes Employing Packet Marking and Logging for IP Traceback. IEEE Transactions on Parallel and Distributed Systems **17**(5) (May 2006) 403 – 418
12. Park, K., Lee, H.: On the Effectiveness of Probabilistic Packet Marking for IP Traceback Under Denial of Service Attack. In: INFOCOM 2001: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 1. (2001) 338 –347
13. Simon, D.: Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. 1. auflage edn. Wiley & Sons (August 2006)
14. Dean, D., Franklin, M., Stubblefield, A.: An Algebraic Approach to IP Traceback. ACM Transactions on Information System Security **5** (May 2002) 119–137
15. Cardenas, A.A., Roosta, T., Sastry, S.: Rethinking Security Properties, Threat Models, and the Design Space in Sensor Networks: A Case Study in SCADA Systems. Ad Hoc Networks **7**(8) (2009) 1434 – 1447 Privacy and Security in Wireless Sensor and Ad Hoc Networks.