Formal Modelling and Analysis of DNP3 Secure Authentication

Raphael Amoah, Seyit Camtepe, Ernest Foo
Queensland University of Technology, Brisbane, 4000, QLD, Australia

Abstract

Supervisory Control and Data Acquisition (SCADA) systems are one of the key foundations of smart grids. The Distributed Network Protocol version 3 (DNP3) is a standard SCADA protocol designed to facilitate communications in substations and smart grid nodes. The protocol is embedded with a security mechanism called Secure Authentication (DNP3-SA). This mechanism ensures that end-to-end communication security is provided in substations. This paper presents a formal model for the behavioural analysis of DNP3-SA using Coloured Petri Nets (CPN). Our DNP3-SA CPN model is capable of testing and verifying various attack scenarios: modification, replay and spoofing, combined complex attack and mitigation strategies. Using the model has revealed a previously unidentified flaw in the DNP3-SA protocol that can be exploited by an attacker that has access to the network interconnecting DNP3 devices. An attacker can launch a successful attack on an outstation without possessing the pre-shared keys by replaying a previously authenticated command with arbitrary parameters. We propose an update to the DNP3-SA protocol that removes the flaw and prevents such attacks. The update is validated and verified using our CPN model proving the effectiveness of the model and importance of the formal protocol analysis.

Keywords: Smart Grid, SCADA, DNP3, DNP3-SA, Formal Methods, CPN

1. Introduction

Supervisory Control and Data Acquisition (SCADA) systems are one of the key foundations of smart grids. Recent literature (Ancilotti et al., 2013; Lu et al., 2013; Yan et al., 2013; Gungor et al., 2013) shows that evolving
smart grids are revolutionising the energy industry and enabling the electricity network to be more reliable and economical. A disruption, either minor or major, deliberately or mistakenly caused to these infrastructures can lead to damaging highly sophisticated devices, inflicting substantial economic loses and posing as life-threatening situations. From the security perspective, legacy SCADA systems have long-lived under obscurity; as a result, they have proven to be insecure to recent cyber attacks\(^1\) (Nicholson et al., 2012; Miller and Rowe, 2012). For instance, had it not been for the stuxnet attack discovered in 2010 (Langner, 2011), which created awareness, SCADA security would have still lived in obscurity. Disrupting functionality in critical infrastructures is a very important issue to consider. Unfortunately, this situation has now become the target area for many malicious attackers. For example, in 2013 intruders managed to shut down a key tunnel road (Carmel Tunnels, in Haifa, Israel) for eight hours causing massive congestion\(^2\).

The Distributed Network Protocol version 3 (DNP3) (IEEE, 2012) is one of the standard SCADA protocols used to facilitate communications in smart grid automation. The protocol is designed such that it can allow smart grid nodes to collect, process, store and control data from DNP3-enabled IEDs (Intelligent Electronic Devices)\(^3\). DNP3 provides a security mechanism called *Secure Authentication* (DNP3-SA), which is used to secure end-to-end communication in substations (Gilchrist, 2008).

Integrating security controls in SCADA protocols, such as authentication and encryption are very important issues to consider in critical infrastructures since functionality, performance and behavioural correctness are crucial. This is to ensure that embedded security mechanisms fit well and do not contain errors that may weaken the security protection provided. The current state of the DNP3-SA protocol is informally described in its specifications. Informal approaches have been known to be very useful in designing


systems. However, they have also yielded inadequate methods which has led to ambiguities and incompleteness (Bolignano et al., 2001; Hall, 2007). Incompleteness among systems may introduce disturbing flaws because they are more focused on functionality than rather behavioural correctness. Provable security (Pass, 2011), which is a common method used to prove security properties of cryptographic primitives, may be an option to deal with ambiguities in systems. However, as attested in Bodei et al. (2005) and Pointcheval (2005), the method is more effective for proving the properties of cryptographic algorithms. Moreover, the provable approach lacks the support of computer-aided tools and as a result it becomes prone to error (Ngo et al., 2010). Formal methods (Woodcock et al., 2009), (which refers to the use of rigorous mathematical techniques and tools for specification, design and verification) provide the ability to construct precise and unambiguous models. These models can be analysed to reduce errors that are often introduced in systems (Tretmans, 1999). This approach effectively helps to reduce the efforts usually required by designers to manually investigate possible conditions that may lead to unexpected events.

This paper presents a Coloured Petri Nets (CPN) (Jensen et al., 2007) based approach that is used to create a parameterised model for DNP3-SA. CPN is a formal and discrete-event modelling language for system design, specification, simulation, validation and verification. Its graphical and programming interfaces provide the ability to express concurrency in communication protocols, data networks, and creating concepts at different levels of abstraction. Parameterisation in CPN is a technique used to create a single model in order to prevent the possibility of having separate models for different behaviours. Applications of CPN have been beneficial in modelling and analysing various industrial processes; ranging from protocols and networks to military systems (Tritilanunt et al., 2006; Floreani et al., 1996). DNP3-SA operates in two modes: Non-aggressive Challenge-Response (NACR) and Aggressive Mode (AGM). In our previous work, we provided a security analysis of the NACR, with a focus on packet inspection at the reception level of an outstation (Amoah et al., 2014). This paper extends our previous work by adding the AGM mode. Specifically, our contribution is three fold. Firstly, we use the concept of parameterisation to create a CPN model that covers the two communication modes of the DNP3-SA; NACR and AGM. The model is based on the specification and the experimental observations of real device behaviour (Substation Modernization Platform / Distribution Processor Gateway (SMP4/DP)). We used the CPN state space analysis tool (Jensen
et al., 2006) to validate the correctness of the model and check the authentication property. Secondly, we identified a violation of the authentication property in the aggressive mode through extensive state space analysis and simulation using the parameterised model. The violation is revealed by a previously unidentified security flaw in the NACR. The flaw allows an attack to manipulation certain sequence of messages to execute commands. Thirdly, by using the parameterised model, we present two different approaches that are used to counter the identified flaw. We analyse the two proposed approaches to show that the flaw has been resolved.

The paper is organised as follows. Section 2 gives an introduction to DNP3. Section 3 and 4 respectively describe the approach used in modelling, model description as well as colour set declarations. Section 5 presents the validation and verification analysis of our DNP3-SA CPN model. Our proposed solution is presented in Section 6. Finally, section 7 presents our discussion, conclusion and future work of the paper.

2. Overview of the Distributed Network Protocol Version 3 (DNP3)

DNP3 is the defacto communication protocol for master stations and outstations in power grids. Exchange of messages in the protocol is in the form of requests and responses. Each of these messages (requests and responses) contain an application control field (AC), function code (FC) and object header (OH). AC is used to determine whether a given fragment has been received in the correct order. FC specifies the action of the request or response sent. OH is supplementary information, usually associated with DNP3 objects that may be required to create a complete DNP3 message. DNP3 objects are index points within the protocol database software that store data such as binary input/output, analog data, and counters. OHs may sometimes be required to accompany function codes in messages to specify what format, type or group of data a station must process and return as response. For example, a master station may use the FC 0x01 and OH g12v1 to read the current analog input type value from the outstation. Furthermore, a response fragment contains an additional field called the Internal Indicator (IIN). IINs are found in responses from outstations. They indicate certain states and error conditions within outstations (IEEE, 2012, p. 13-23).

DNP3-SA is the security mechanism, which provides authentication in the application layer of the DNP3 protocol. The mechanism ensures that stations are protected against “rogue applications” that may want to manipulate
the protocol. The authentication mechanism is unilateral but it operates in two ways; one-pass and two-pass authentication through a Keyed-Hash Message Authentication Code (HMAC). Two-pass authentication is known as the challenge-response or Non-aggressive challenge response (NACR) while the one-pass is the aggressive mode (AGM). It is to be noted that before the AGM (one-pass) operation can be carried out, there should be at least one or more occurrences of the NACR operations. This enables the AGM to make use of certain crucial components from NACR operation (this behaviour is elaborated and illustrated in Figure 1). In terms of operation, DNP3-SA strictly ensures that certain requests, particularly ‘critical requests’, invoked by either master stations or outstations, are challenged and authenticated correctly for every session before they are further processed. A request or unsolicited response is considered critical, if the message contains a mandatory code. A mandatory code is any code that can potentially control a given station, by performing set-point adjustments or setting certain parameters. Any station that makes use of mandatory codes in a given message shall be challenged by the receiving device’s security mechanism to prove its identity.

Table 1 depicts a message sequence chart (MSC) that presents the behaviour of DNP3-SA. For simplicity, we have omitted the AC fields in all DNP3 packets because they do not contribute to the result of this paper. Master station and Outstation represent the communicating entities. \( Csk_{mo} \) is a controlling session key obtained from a long-term secret key, \( L_k \), which is manually distributed among the entities. The session key is used to authenticate data transmitted in the control direction by the master station. \( FC \), \( OH \) and \( IIN \) respectively represent the function code, object header and internal indicator data that may be contained in a request or response (standard and error). Standard responses are expected responses for a particular request sent, while an error response could be a failure in authentication. \( Chlg \) represents a challenge message. It contains a Challenge Sequence Number (CSQ), \( Sn \) that increases by \( i \) (where \( i \leftarrow 1 \)) each time a challenge is issued, a Message Authentication Code (MAC) algorithm, \( H \) and a nonce \( N \). \( HMAC_1 \) and \( HMAC_2 \) represent HMAC tags generated by the master station and outstation respectively. Finally, \( U_{ID} \) represents a user identification number, which is associated with the communicating parties.

In Table 1, NACR presents the non-aggressive challenge-response operation. The master station sends a request that requires a critical service to the outstation. On receipt of the request, the outstation issues \( Chlg \) accordingly. The master station uses \( H \) from \( Chlg \) and \( Csk_{mo} \) to hash \( Chlg \) received, the
request issued \((FC,OH)\), and \(U_{ID}\) to compute its HMAC tag \((HMAC_1)\). The HMAC tag is then sent to the outstation. On receipt of the tag, the outstation uses the same computational approach and elements to generate its tag \((HMAC_2)\) and checks if the tags match. If the tags match, then authentication has been successful. The outstation processes the request and replies to the master station with a standard response. Otherwise, an error message is sent, which signifies that authentication has failed.

In Table 1, AGM illustrates the aggressive mode operation. In this operation, we assume that the master has “the most recent challenge message”. As previously stated, the AGM operation heavily depends on NACR. That is, every last challenge message received by the master station during a successful NACR operation, is considered to be the “most recently received challenge message”. If a master station wishes to use the AGM operation to communicate with an outstation, it is expected that the master station must use all elements of the most recent challenge message, and only update the CSQ to the next counter, before computation of the HMAC tag is done. In Table 1, AGM shows that before the master station can send an AGM request for a critical service, it first needs to compute an HMAC tag \((HMAC_1)\) on the request about to be sent. The master use the most recent challenge message, \(Chlg\), and increased \(Sn\) by 1 to compute a valid HMAC tag. In computing the HMAC tag, the methodology employed in NACR is used. That is, the master station uses \(H\) from \(Chlg\) together with \(Csk_{mo}\) to hash the updated \(Chlg\), request (about to sent), and \(U_{ID}\) to compute a valid HMAC tag. After the tag has been obtained, the master station finally sends the request together with \(HMAC_1\) to the outstation. On the receipt of the message, the outstation stores the received tag, extract the \(FC\) and \(OH\) from the request, and uses the same methodology to compute its own HMAC tag \((HMAC_2\) based on request received and an updated \(Sn\)). After obtaining \(HMAC_3\), the outstation compares the tags. If the tags match, then the request is further processed and a standard response is generated and sent to the master station. Otherwise, an error message is generated. In our next section, we will present a CPN model of the DNP3-SA protocol.

### 3. The CPN Model of DNP3-SA

As depicted in Figure 2, a CPN model consists of places (circular), transitions (rectangular) and directed arcs (arrows between places and transitions). Places represent various states in a given model. They are assigned
with colour sets (colsets). Colour sets are data types such as integer, string and record that enable places to store data (token(s) or multisets - more than one token). A place with a token(s) is known as a Marking. Transitions signify events and they occur (i.e. moving a token from one place to another), while directed arcs connects places to transitions. CPN also uses other elements such as variables and functions. Variables represent colour sets. They are typed and can be assigned values. An assigned variable on an arc is referred to as a binding element. Moreover, whenever a transition is said to have occurred, it represents a state change. For example, in Figure 1 we present a MSC depicting a simple protocol where SenderA transmits messages to ReceiverA to introduce its users. The MSC is translated into a CPN model, which is depicted in Figure 2. In this example, the process in creating the model involve 4 steps; 1) identifying the communicating entities, 2) identifying the messages to be exchanged between the entities, 3) creating the channel for communication, and 4) identifying the data type for the message to be exchange. The translation is as follows: In the MSC, SenderA and ReceiverA are represented by places in the CPN model (SenderA and ReceiverA, see Figure 2). The events of sending and receiving messages from SenderA to ReceiverA are represented by 2 transitions: Sending Messages and Receiving Messages. Messages on SenderA in the MSC are represented as various tokens (multiset) on place SenderA of the CPN model. The network part (which is the arc that connects SenderA to ReceiverA in the MSC) is represented by place Network in the CPN model. Finally, the transmission of each messages from SenderA to ReceiverA in the MSC is represented with the CPN variable msg (see msg on arcs from SenderA to ReceiverA).
Table 1: A MSC depicting the generic behaviour of DNP3-SA.
3.1. CPN Modelling Approach

Two modelling approaches, which are specific to DNP3-SA were used: Protocol Abstraction and Model Parameterisation with Adversaries.

**Protocol Abstraction:** In translating and obtaining a complete model for DNP3-SA, we made use of DNP3 specification, two Substation Modernization Platform / Distribution Processor Gateway (SMP4/DP) devices from Cooper Power Systems\(^4\), and a Linux-based computer and a switch. The idea is to set up a small SCADA network mimicking the behaviour of typical substation in a power distribution network. SMP4 devices are used to convert almost any protocol to another using various protocol libraries. For testing purposes, they can also be configured to act as a master station or slave by using preloaded protocol templates. In our case, we configured one of the devices as a DNP3 master and the other device to behave as a DNP3 slave. With preloaded templates, the slave could conveniently generate DNP3 data objects internally for testing. Then, we set up our Linux-based computer to continuously monitor traffic from the master station and outstation. The motivation of this test-bed is to obtain precise protocol behaviour such the relationship between NACR and AGM.

**Model Parameterisation with Adversaries:** This approach is used to prevent the possibility of having separate models in order to obtain different behaviours for analysis. It provides the ability to combine multiple instances of certain behaviours seamlessly. They are usually enforced on either arc inscriptions, transition guards, or even code regions. Our model uses parameterisation to combine our attack models as well as other behaviours that are not stated in the DNP3-SA’s specification. By doing so, it gives us the ability to control when to turn them ON←true or OFF←false, depending on the environment for testing. This paper only considers the generic adversaries stated in the protocol specification; modification, spoofing and replay. These attacks are considered because they have been stated in the protocol specification to be addressed by DNP3-SA. Therefore, parameterisation gives us flexibility to re-use our existing model. The same model can behave differently depending on the parameters provided for analysis.

3.2. Assumptions

Without loss of generality, we make the following assumptions:

1) The master station can use both security modes (NACR and AGM) in communicating with an outstation. 2) Master station will be the sole entity that initiates communication; sending of request. 3) Both entities have their long-term secret keys as well as session keys pre-established. 4) Every request transmitted from the master station is considered as a critical request. 5) The underlying layers of the protocol via the network are reliable.

4. Model Description

This section presents the CPN colour set declarations and the hierarchical CPN model of DNP3-SA. The hierarchical model consists of three modules: A top-level CPN page, a second-level CPN page and 14 third-level CPN pages. The top-level and second-level pages are used to simplify the complexities of the model. The third-level pages present details of the model.

4.1. Colour set Declaration

Table 2 presents a sample declaration (DNP3-SA messages to CPN mapping) for NACR and AGM request messages in DNP3-SA CPN. The complete list of messages can be found in the Appendix A (Table A.7). Table 2 consists of three columns: Remarks, Components and CPN declarations. Remarks in the table present the various types of messages used in the model. Components show the elements that make up the various messages. CPN declarations present the translation of DNP3 messages into the CPN language. For example, in the Requests row of the table (Table 2), we show that a request fragment is made up of FC and OH components. These components have their corresponding CPN declarations set to be colset fcode & oheader. The colour sets (fcode & oheader) have a timed string as their data types. In addition, both colour sets have also been assigned with timed token values: 1'(“0x01”)@1++ 1'(“0x02”)@2 1'(“0x03”)@3, & 1'(“g20v1”)@1++ 2'(“g20v7”)@2 respectively. The tokens values; 1'(“0x01”), 1'(“0x02”) and 1'(“0x03”) represent DNP3 messages Read, Write and Initiate Application respectively. In summary, a complete NACR CPN request from Table 2 is modelled as the product of Fcode and oheader; (“0x01”,“g20v1”). While AGM request is modelled as the product of colour set Request and hmac (refer to Table A.7 for details).
### Table 2: CPN Declaration for DNP3-SA Model

<table>
<thead>
<tr>
<th>Remarks</th>
<th>Components</th>
<th>CPN Declarations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>FC</td>
<td>Val allrqfc = 1'(&quot;0x01&quot;)@1++ 1'(&quot;0x02&quot;)@2++ 1'(&quot;0x03&quot;)@3; Colset fcode = string timed; Var f:fcode</td>
</tr>
<tr>
<td></td>
<td>OH</td>
<td>Val alloh = 1'(&quot;g20v1&quot;)@1++ 2'(&quot;g20v7&quot;)@2; Colset oheader = string timed; Var oh:oheader</td>
</tr>
<tr>
<td></td>
<td>CPN Request</td>
<td>Colset Request = product fcode&quot;oheader</td>
</tr>
<tr>
<td></td>
<td>CPN AGM Request</td>
<td>Colset AGrequest = product Request&quot;hmac</td>
</tr>
</tbody>
</table>

![Diagram of DNP3-SA CPN model - Top-level page.]

**Figure 3: DNP3-SA CPN model - Top-level page.**

### 4.2. DNP-SA CPN Model Specification

**Top-level Page:** Figure 3 shows the main page for the DNP3-SA CPN model. It presents an abstract view of the model on a single page. It constitutes of three substitution transitions (T\textsubscript{sub}); Master station, Network\_MITM and Outstation. A substitution transition is a page (submodule) with multiple layers of details. The directed arcs between the substitution transitions indicate that the destination page is also a submodule of the source page.

**Second-level Page:** Figure 4 shows the multi-stage page of the DNP3-SA CPN model. It consists of eight different substitution transitions. In the figure, all substitution transitions are marked with a number to indicate the flow of messages in the model. On the far-left of the figure (Figure 4), T\textsubscript{sub}; SendRequests, MasterRecv & MasSecure model the behaviour of the master station. The middle part of the figure depicts T\textsubscript{sub}; NETWORK\_MITM (the network and other activities such as a man-in-the-middle attack (MITM)). Finally, the far-right side of the figure presents T\textsubscript{sub}; OutRecv\_Send, CHGenerator, OutSecure & OProcess, model the outstation’s behaviour. Each module presented in this page has details of its behaviour presented on a third-level page. For example, details of T\textsubscript{sub} MasterRecv (8) & Network\_MITM (2) are depicted in Figure 5 & 6 respectively.
tively. To improve readability and understanding, third-level pages that are relevant and contribute to this paper are presented in Appendix B).

Figure 5 depicts details of the final step in the model; \( T_{\text{sub}} \text{MasterRecv} \). In this figure, the master station receives and stores responses from the outstation. Furthermore, as the master station receives responses, it can also enable itself and execute subsequent requests through \( T_{\text{sub}} \text{SendRequests} \).

Figure 6 presents \( T_{\text{sub}} \text{Network/MITM} \), a third-level page from Figure 4. It presents details of the network as well as parameterised attack behaviours carried out by a Man-In-The-Middle attack (MITM). For clarity, this figure marks the second process after \( T_{\text{sub}} \text{SendRequests} \) from Figure 4 has occurred. It is divided into three parts; top, middle and bottom to emphasise the activities involved. The top part of the figure (starting from the left transition \text{ConnectA}) depicts the transmission NACR requests from the master station through \( \text{SendRq} \) and forward it to the outstation via \( \text{sendRq} \). Similarly on the far right, transition \text{Connect AGM} signifies the transmission of the AGM request from the master through place \( \text{SendAGMRq} \) to the outstation via \( \text{sendAGRq} \). If any of the MITM parameters (guard functions on transition \text{LReplayA} & \text{Connect AGM}) are set to \text{true}, it may enable the attacker to launch malicious activities such as modifying, replaying requests or spoofing identity (particularly the master) in an attempt to gain access or even a command on the outstation.

The middle part of the figure (refer to transition \text{ConnectC}, starting from the bottom) depicts the transmission of challenge messages from the outstation to the master station through place \( \text{SendChlg1} \) to \( \text{sendChlg1} \). Similarly on the far right, it is depicted that transition \text{Sol.ConnectC} sends challenge messages from the outstation to the master via \( \text{SendChlg2} \) to \( \text{sendChlg2} \). Here, if the attack parameter (arc inscriptions on the places) is set to \text{true}, it is expected that the attacker will be able modify values in every passing challenge message from the outstation to the master. It is to be noted that transition \text{Sol.ConnectC} parameter is set to \text{false}. The transition is set to false because it is not considered as a behaviour of the DNP3-SA protocol.

At the bottom of the figure, transition \text{ConnectB}, place \( \text{SendRsp} \) and \( \text{sendRsp} \) are depicted. They model the transmission of response from the outstation to the master station.
Figure 4: DNP3-SA CPN model - Generic view of the Second-level page.
5. Validation and Verification of DNP3-SA CPN Model

We perform the validation of DNP3-SA using CPN’s state space analysis tool (Jensen et al., 2006). In validating the DNP3-SA CPN model, it is important to consider the following: 1) formally defining the authentication property using the Computational Tree Logic (CTL), which is supported by CPN tools and 2) ensuring that the MITM attacks in the model as well as other behaviours, which are not part of the protocol specification for now.
are set to \texttt{false}. This is to help determine whether the model reflects the protocol behaviour under normal circumstances.

Analysis using the state space presents a report with a number of behavioural properties. These properties include home markings, dead markings, live and dead transitions among others (discussed later during the validation stage). In this paper, we are interested in dead markings and dead transitions. Dead markings are states that do not have binding elements to make them active (Jensen et al., 2007). They represent termination points in the model. Dead transitions on the other hand, are transitions that cannot be executed because there exists no path from a reachable marking to enable them. These properties are of interest due to their ability in effectively simplifying the analyses of large and complex models without loosing the generality of the concepts. For instance, throughout these analyses we expect a single dead marking representing the outstation processing all requests after authentication for each request has been successful. We also expect 14 dead transitions during the initial validation process; where out of the 14 transitions, 2 of them represent the absence of attack models. The remaining transitions depict other parameters that are set to \texttt{false}; consisting of attacks parameters as well as other behaviours that are not part the DNP3-SA's specification. It is to be noted dead transitions are subjective to change during all analyses. This is as a result of some dead transitions such as attack models that might later be set to \texttt{true}.

5.1. Formalising the Authentication Property of DNP3-SA

For all validations, verifications, and evaluation purposes, we define and later formally represent the authentication property using CTL. 

\textbf{Authentication condition:} The outstation is able to verify the identity of the master station, if the master station is able to produce a valid HMAC tag; where the tag is the same as the tag computed by the outstation. Thus, if the tags are the same, then authentication is successful, otherwise, authentication fails. A pass during authentication leads to a single state where IIN values; ’00\_IIN\_1’ or ’00\_IIN\_2’ are obtained on place \texttt{M\_Internal} of the \texttt{T\_sub\_MasterRecv} (8) (refer to Figure 5 for details). If for any reasons, values other than the expected values are obtained on \texttt{M\_Internal}, then authentication has failed. This requirement applies regardless of whether the attack parameters are set to \texttt{true} or \texttt{false}, or the mode of operation employed is NACR or AGM.
Definition of the authentication property corresponds to a subset of places in $T_{\text{OutSecure}}$ (6) and $T_{\text{MasterRecv}}$ (8) in Figure 4. Interested readers can also make reference to Figure 5, and Figure B.19 of the Appendix B for details. In $T_{\text{OutSecure}}$ (6) (Figure B.19 of Appendix B for details) three (3) places were used. That is, a place called $\text{StoreHMAC}$ models the storage area for HMAC tags from the master station. The second place is called $\text{OHMAC}$. It models the outstation’s storage area for its own HMAC tags. While the third place is called $\text{AuthStatus}$, it models the authentication status (i.e. an authentication pass or fail). In $T_{\text{MasterRecv}}$ (8), a single place called $\text{MInternal}$ models the storage area for IIN values on the master station (refer to Figure 5).

The following are CTL statements that are used in formalising the property.

- $\text{Ev}(A)$: $\forall A \ldots$ refers to a truth value. This operator returns a true value if all paths, from a given state, lead to a state where $A$ is true. $\text{Pos}$: $\exists A \ldots$ refers to a truth value. This operator also returns a true value if there exists at least one path, from a given state, that leads to a state where $A$ is true. This $\text{IM}^{\text{CPN}}$ denotes to our initial CPN model without attacks. $\text{MITM}^{\text{CPN}}$ denotes our attack model in $\text{IM}^{\text{CPN}}$. $\text{MD}^A$, $\text{RP}^A$, $\text{SP}^A$ denotes modification, replay and spoofing attack models carried out by a MITM respectively. $\text{MRSA}^A$ denotes all attacks set to true in $\text{MITM}^{\text{CPN}}$. $\text{M}_0$ denotes the initial marking or state in $\text{IM}^{\text{CPN}}$. $\text{[M}_0\text{]}$ denotes the set of reachable markings from $\text{M}_0$ and $(\text{SAM}(M_i, P^{\text{PlaceName}}))$ denotes a place name at a marking $M_i$ ($M_i \in \text{[M}_0\text{]}$). Furthermore, $\text{M}^{\text{tag}}$ and $\text{O}^{\text{tag}}$ represent the master station and the outstation calculated HMAC tags. $\text{MO}^{\text{true}}$ and $\text{MO}^{\text{false}}$ represent the match and mismatch of HMAC tags. $\text{Tg}^{\text{true}}$ and $\text{Tg}^{\text{false}}$ represent authentication pass or failure. Finally, $\text{I}^t$ and $\text{I}^f$ denote the IIN values to be obtained on the master station ($\text{MInternal}$ on Figure 5).

**Property - Authentication Property**

Predicates for $\text{IM}^{\text{CPN}}$:

- $\text{M}^{\text{tag}} M_i = (\text{SAM}(M_i, P^{\text{StoreMAC}})) \neq \emptyset$
- $\text{O}^{\text{tag}} M_i = (\text{SAM}(M_i, P^{\text{OHMAC}})) \neq \emptyset$
- $\text{MO}^{\text{true}} M_i = (\text{SAM}(M_i, P^{\text{OHMAC}})) = (\text{SAM}(P^{\text{StoreMAC}}))$
- $\text{MO}^{\text{false}} M_i = (\text{SAM}(M_i, P^{\text{OHMAC}})) \neq (\text{SAM}(P^{\text{StoreMAC}}))$
- $\text{Tg}^{\text{true}} M_i = (\text{SAM}(M_i, P^{\text{AuthStatus}})) = \text{"Authpass"}$
- $Tg^{\text{False}} M_i = (\text{SAM}(M_i, P^\text{AuthStatus})) = 1^\prime \text{"Authfail"}$

- $\text{If } M_i = (\text{SAM}(M_i, P^\text{Internal})) = (1^\prime \text{"00_IIN_1"}) \lor (1^\prime \text{"00_IIN_2"})$

- $\text{If } M_i = (\text{SAM}(M_i, P^\text{Internal})) = (1^\prime \text{"sysfail"})$

DNP3-SA is agreed to have an uncompromising authentication property, iff $\text{IM}^{\text{CPN}} \lor \text{MITM}^{\text{CPN}}$ have the following behaviour:

- If all attacks carried out in $\text{MITM}^{\text{CPN}}$ are false, then
  \[ \forall M_i \in [M_0] \, \text{Ev}(M^\text{True} M_i \land Tg^\text{True} M_i \land I^\text{t} M_i) \land \neg \text{Pos}(Tg^\text{False} M_i \land I^\text{f} M_i) \]

- Otherwise, if $\text{MD}^\text{A} \lor \text{RP}^\text{A} \lor \text{SP}^\text{A} \lor \text{MRS}^\text{A}$ in $\text{MITM}^{\text{CPN}}$ are true, then
  \[ \exists M_i \in [M_0] \, \text{Ev}(M^\text{False} M_i \land Tg^\text{False} M_i \land I^\text{f} M_i) \land \neg \text{Pos}(Tg^\text{True} M_i \land I^\text{t} M_i) \]

<table>
<thead>
<tr>
<th>Table 3: State Space Reports - DNP3-SA CPN Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State Space Report for DNP3-SA CPN Model</strong></td>
</tr>
<tr>
<td><strong>Initial CPN Model (IM^{\text{CPN}})</strong></td>
</tr>
<tr>
<td><strong>State Space Nodes</strong></td>
</tr>
<tr>
<td><strong>State Space Arcs</strong></td>
</tr>
<tr>
<td><strong>SCC Graph Nodes</strong></td>
</tr>
<tr>
<td><strong>SCC Graph Arcs</strong></td>
</tr>
<tr>
<td><strong>Dead Markings</strong></td>
</tr>
<tr>
<td><strong>Dead Transitions</strong></td>
</tr>
</tbody>
</table>

5.2. Validation of DNP3-SA CPN model

We present the full state space reports of DNP3-SA CPN model with all attacks stated in the specification. The table consists of two groups; Initial CPN model ($\text{IM}^{\text{CPN}}$) and MITM Attacks ($\text{MITM}^{\text{CPN}}$). $\text{IM}^{\text{CPN}}$ implies that all attack parameters are set to false. $\text{MITM}^{\text{CPN}}$ implies that one or more attacks have can be set to true, henceforth switched on.

$\text{IM}^{\text{CPN}}$ in Table 3 depicts that the State Space Nodes and Arcs have matching values with the Strongly Connected Components (SCC) Graph Nodes
and Arcs. SCC is a property used to determine the reachability of nodes and iterations in models. Obtaining the same values for state space and SCC indicates that every node in the model is reachable from any other node. Furthermore, there exists no loops in the model that may render the model to have infinite occurrences.

The report also shows that there exists one dead marking; instance (299) and 14 dead transitions. These imply that the obtained values are consistent with our expectations defined in Section 5. The reason is that the single dead marking obtained represents a single state. This single state is where each request transmitted by the master station is successfully authenticated by the outstation. Thus, leading to the occurrences of all transitions in $T_{sub \text{ Process}}$ except $T_{sub \text{ AuthErr}}$, which is a submodule embedded in $O\text{Process} - 7$ (refer to Figure 4 and Figure B.20 in Appendix B). $T_{sub \text{ AuthErr}}$ marks authentication error in module $O\text{Process} - 7$ (refer to Figure B.20 in Appendix B for details). Moreover, a closer inspection on the dead transitions revealed that all 14 dead transitions are consistent as well. That is, two (2) out of the 14 dead transitions were found to be inactive, because no attack was turned ON. They reflected the generation points for authentication error messages ($T_{sub \text{ AuthErr}}$ of Figure B.20). The 12 remaining transitions reflected other activities (not part of DNP3-SA’s specification) and all attack behaviours that are set to false. Achieving this behaviour concludes that the DNP3-SA CPN model behaves as expected, as the model is able to verify the authenticity of the master station for every request executed.

5.3. Verifying DNP3-SA against attacks stated in the specification

The following attacks are carried out by a MITM, which have been parameterised in the DNP3-SA CPN model (refer to Figure 6)

- $MD^A$ denotes modification attack, where the attacker is able to modify packets such as requests and contents of challenge messages.
- $RP^A$ denotes replay attack, where the attacker is able to replay any messages from the master station.
- $SPA^A$ denotes spoofing attack, where the attacker is allowed to masquerade as the master station to the outstation.
- $MRS^A$ denotes all attacks are set to true. This allows the attacker to launch all attacks.
MITM in Table 3 presents the reports for all attacks that have been set to true in DNP3-SA CPN model. From the table, MD and RP present a single dead marking is obtained for each. A close inspection revealed that the dead marking obtained for each attack represents the termination of all executed requests with all their authentication failing; leading to the execution of $T_{\text{AuthErr}}$. As a result, 17 dead transitions were obtained for each attack. A close inspection on the obtained dead transitions revealed that, out of 17, five (5) of them marked various activities such as reading, writing or initiating an application in $T_{\text{OProcess}}$ that did not occur. This is as a result of the attacks. The remaining 12 dead transitions marked paremeterised transitions that are set to false. These results are consistent with our expectations and implies that the authentication property hold.

MITM, SP in Table 3 presents 6 dead markings and 28 dead transitions obtained. This attack behaviour rendered the model to become inaccessible. This behaviour is expected, as our attack model is under the assumption that session keys between the stations are pre-shared before the attack. This behaviour leads DNP3-SA CPN model to hang and not to complete successfully. This, does not affect the behaviour of the protocol nor does it renders the protocol to be inaccessible in real life. In practice, a timer allows the protocol to drop a given transaction after a threshold. We, however, carried out this analysis to further investigate impacts and benefits it has on the DNP3-SA (i.e. used in conjunction with other attack models).

5.4. Evaluation of DNP3-SA CPN model

In this section, we set all attack parameters to true. This combines all attacks behaviours specified above. Then, we set place Tagcount of Figure 6 to the value 1 and also set the modification attack parameter to manipulate the challenge message from the outstation (refer to the middle part of Figure 6). These settings are very crucial because: 1) it enables the attacker to gather enough data after two (2) initial requests have been successful executed on the outstation during NACR operation, and 2) the master station can consider the last challenge data from the outstation as its most recent challenge data for AGM operations.

MITM, MRS in Table 3 presents 72 instances of dead markings and only 12 dead transitions. Results obtained in this report imply that the model has certain unexpected behaviours. This is as a result of the high number of dead markings obtained. This behaviour, therefore calls for further investigations. We define three (3) SML queries and use the interactive simulation
provided by the CPN tools to conduct the investigations. The SML query functions include: \textit{ListDeadMarkings()}, \textit{SearchNodes} and \textit{reachability}. \textit{ListDeadMarkings()} is used to list all dead markings identified in the model. \textit{SearchNodes} is used to search for all possible paths of a given marking. For example, identifying a marking that represents the state of the attacker in the model. \textit{Reachability} is a property focused in determining whether there exists an occurrence sequence starting from the marking of node \textit{A} to the marking of node \textit{B}. In the model, reaching at least a \textit{single state} from where the attacker can transmit request and the authentication for that request fails; implies that the authentication property holds in DNP3-SA. On the contrary (from the attacker’s state), if one can still reach a \textit{state} where the authentication is successful then, the authentication property is said to be violated.

In the analysis, we used \textit{ListDeadMarkings()} to obtained all dead markings stated in the report and further simulate them individually. Simulating all the obtained markings enabled us to classify the dead markings into two (2) main groups: \textit{expected} and \textit{unexpected}. According to the configuration of the model, an expected marking is a state, where it is expected that only two (2) of the initial requests are being successfully executed on the outstation, but having subsequent requests failing. However, for a state that has \textit{more than two} (2) of its initial requests successfully executed on the outstation implies that, it is an unexpected dead marking. From the simulation, 36 instances (which ranged from 11698 – 11733) of unexpected dead markings were listed (not listed due of space limitations). Obtaining these instances require a close inspection. The inspection leads to the usage of the \textit{SearchNodes} query. We used the \textit{SearchNodes} query within a function to search for all possible paths that issued requests from the attacker state. This is to search and return at least one node value.

Figure 7 depicts our results. The top-side of the figure presents the \textit{SearchNodes} queries used within a function and its outcome. The node value obtained from the query (96) is used against all the unexpected dead markings (11698 – 11733) to determine if it is possible to reach them. At the bottom of Figure 7, the reachability statements used are depicted. The statements returned \textit{true} values for all the unexpected dead markings instances; implying that all instances are reachable from the attacker state. Relating this outcome to the authentication property defined in Section 5.1 implies that the property has been violated in the presence of this attack model. Even though, the attacker does not possess the key to compute valid tags for
authentication; a write command has been successfully executed on an IED in the model (transition $T_{writeB}$). **Observations and Results:** Inspecting the results through simulation revealed that there exists instances where the attacker, without the knowledge of any key, is capable of replaying valid AGM messages to execute commands on the outstation; writing or reading values or even initiating an application on an IED during the AGM operation. Since the protocol messages are transmitted in ‘cleartext’, requests and challenge messages can be manipulated, which can prevent relevant protocol sequences from completing during the NACR mode. For instance, randomly manipulating CSQs or nonces in challenge messages can cause loss of synchronisation and unexpected authentication failures. In the model the attacker instead, can sequentially manipulate the CSQ in the challenge message to the next counter value after observing that the CSQ value increases by one for every session. This is because the CSQ value can be predicted easily; allowing the CSQ value to be set higher. In doing so, both stations are ‘fooled’ into producing HMAC tags, which result in an authentication failure during the NACR verification process. Although, the tag verification fails in the NACR mode, the same (old) tag becomes valid if the tag is coupled with a particular request and used during AGM mode of operation. Thus,
when an authentication fails during NACR, the outstation still continues to update its CSQ for its next challenge. Understanding this flaw enables the attacker to replay the old tag (previously captured from the master station during NACR mode) to the outstation as an aggressive message. In doing so, a successful execution of a command can be carried out via transition $T_{\text{writeB}}$ (refer to Figure B.20 in Appendix B). In our model, this behaviour has led to identifying 12 out 17 expected dead transitions. Therefore, this behaviour violates the authentication property defined in Section 5.1.

6. Proposed Solutions for the Identified Flaw

Analyses using the DNP3-SA CPN model have proven the protocol can withstand individual attacks stated in the DNP3 specification. However, the model has also revealed a flaw when all attacks are enabled. In the protocol specification, it is stated that the aggressive mode operation is “somewhat more secure against replay attacks”. Analysis from our model has revealed a flaw in the protocol, which is inherited from the NACR mode. The flaw allows an attacker, without possessing any key to replay a valid HMAC tag (old message) from one mode of operation to another, and execute commands on the outstation. Impacts of such attacks on real-life critical systems such as the smart grid can be extremely detrimental. As a result, we present and model two (2) approaches that can be used to fix the flaw identified. The proposed approaches are further validated and verified using the model.

6.1. Solution 1: HMAC for challenges data

In Table 4, we depict an approach that suggests the computation of an HMAC tag on the challenge message from either the master station or outstation; depending on who the challenger is. Computation of the tag is based on user association ID and hash functions (such as SHA-HMAC and AES-GMAC) already used in the DNP3-SA protocol. Table 4 illustrates the master station sending a critical request to the outstation. On receipt of the request, the outstation generates $Chalg\left(Sn, H, N\right)$. $Chalg$ is then used together with $U_{ID}$ by the outstation as input elements to compute $CHMAC_0$. $CHMAC_0$ is computed by using $H$ from $Chalg$, the session key, $Csk_{mo}$ and the input elements ($Chalg$, $U_{ID}$). The outstation, then sends a message comprising of $CHMAC_0$ and $Chalg$ to the master station. On receipt of the message, the master station uses its $U_{ID}$ and $Chalg$ from the outstation as its inputs to compute $CHMAC_1$. Then, the master station compares the
tags for equality. If the tags match, the master station is triggered through 
\(ETHMAC\) to compute an HMAC tag on the initial request. Otherwise, the 
protocol logs the failure and aborts the operation \((logError&Abort)\). This 
behaviour will prevent the master station (responder) from being ‘tricked’ 
into computing a valid HMAC tag, which will later become beneficial to the 
attacker for a successful attack. This approach also helps to prevent the out-
station from automatically updating its CSQ when a reply (expected HMAC 
tag on the ‘waiting to process request’) is not yet received.

<table>
<thead>
<tr>
<th>Master Station</th>
<th>Outstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Cs_k_{mo})</td>
<td>(Cs_k_{mo})</td>
</tr>
<tr>
<td>(\leftarrow L_k \Rightarrow)</td>
<td>(\leftarrow L_k \Rightarrow)</td>
</tr>
</tbody>
</table>

**NACR**

\[Request \leftarrow (FC, OH)\]

\[
CHMAC_1 \leftarrow H_{Cs_k}(Chalg, U_ID)
\]

\[CHMAC_0 \leftarrow H_{Cs_k}(Chalg, U_ID)\]

\[CHMAC_1 = CHMAC_0\] 

\[ETHMAC \leftarrow (FC, OH, IIN)\]

\[logError&Abort\]

| Table 4: Solution 1 implementation of HMAC for challenge messages. |

6.2. Solution 2: Randomisation for CSQ / Limiting CSQ to AGM

According to the DNP3-SA standards, CSQ is a mechanism used to se-
quentially maintain a match of replies (responses) between stations. That is,
CSQs in stations are set to zero (0) at startup. However, the CSQs increase 
by 1 each time a station challenges or requires authentication. Our experi-
ments on real devices (SMP4) has revealed that the mechanism (CSQ) is not 
only used for synchronisation but it also serves as a key element for imple-
menting the AGM mode of operation (i.e. transitioning from one mode of
operation to another (NACR to AGM) for less bandwidth usage (see IEEE, 2012, p. 175)). Analysis from DNP3-SA CPN model has revealed a flaw associated with the CSQ, because the attacker could easily guess what the next CSQ will be. As a result, a successful replay attack was carried out. Randomising CSQs during the NACR operation will make it almost impractical for the attack to be successful. This is because, each time the CSQ is issued, the value is chosen at random; making it difficult for the attacker to guess what the next value could be. This approach, although proves promising, may lead on to increase in protocol overheads.

We recommend an alternative way over randomisation. We suggest that the protocol will be better off discarding the use of the CSQ mechanism in the NACR mode. Instead, we suggest the CSQ mechanism to be introduced solely in the AGM mode rather than being an active component of the NACR mode operation. The presence of the CSQ mechanism in NACR is seemingly redundant. As a result, it leads to manipulation; launching successful attack on real-life systems. We suggest that the pseudo-random number (nonce) and the MAC algorithm in the challenge message will be effective (sufficient) components to run the NACR mode without difficulties. However, if for any reason the AGM mode is to be carried out for operations, then we advise on the following: 1) integrating the CSQ mechanism as part of the AGM operation and 2) making use of NACR’s challenge (‘the most recent challenge message’, which consists of the nonce + MAC algorithm). We illustrate our solution in Table 5, under the assumption that the CSQ mechanism is part of the AGM and not the NACR operation. Table 5 shows that after a request is received by the outstation, the outstation issues Chalg\(_0\) to the master station. It is to be noted that Chalg\(_0\) is composed of only \(H, N\) (HMAC algorithm and nonce). However, for the AGM operation, computation of the HMAC tag is based on Chalg\(_0\) (\(H&N\)) and includes Sn (represented as Chalg\(_1\)). This approach (i.e. introducing CSQ in AGM) still provides synchronisation among other stations. However, the real benefit is that the computation of the HMAC tag for the AGM request will be completed before an attack will be able to guess the CSQ. This approach deters tricking stations during the NACR operation to compute valid tag; which later can be replayed with a request to carry out an event. Finally, this approach also helps to prevent challengers (outstations) from updating their CSQs automatically when authentication fails during a NACR operation.
<table>
<thead>
<tr>
<th>Master Station</th>
<th>$C_{sk_{mo}}$</th>
<th>$\iff L_k \iff$</th>
<th>Outstation</th>
<th>$C_{sk_{mo}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NACR</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$Request \leftarrow (FC, OH)$</td>
<td></td>
<td></td>
<td>$Chal_{0} \leftarrow (H,N)$</td>
<td></td>
</tr>
<tr>
<td>$HMAC_1 \leftarrow H_{C_{sk}}(Chal_{0}, Request, U_{ID})$</td>
<td>$\iff Chal_{0}$</td>
<td>$\iff HMAC_1$</td>
<td>$HMAC_2 \leftarrow H_{C_{sk}}(Chal_{0}, Request, U_{ID})$</td>
<td>$\iff HMAC_2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Verify HMAC Tags</td>
<td>[ Verify HMAC Tags ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[ If(HMAC_2 = HMAC_1) then ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Response $\leftarrow (FC, OH, IIN)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Else</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EResponse $\leftarrow (FC, OH, IIN)$</td>
</tr>
<tr>
<td><strong>AGM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$HMAC_1 \leftarrow H_{C_{sk}}(Chal_{1}{Sn, H, N}, Request, U_{ID})$</td>
<td></td>
<td></td>
<td>$HMAC_2 \leftarrow H_{C_{sk}}(Chal_{1}{Sn, H, N}, Request, U_{ID})$</td>
<td>$\iff Request$</td>
</tr>
<tr>
<td>$Request \leftarrow {(FC, OH), HMAC_1}$</td>
<td>$\iff Request$</td>
<td>$\iff HMAC_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Verify HMAC Tags</td>
<td>[ Verify HMAC Tags ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[ If(HMAC_2 = HMAC_1) then ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Response $\leftarrow (FC, OH, IIN)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Else</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EResponse $\leftarrow (FC, OH, IIN)$</td>
</tr>
</tbody>
</table>

Table 5: Solution 2, Limiting CSQ to AGM operation
6.3. Specification of New DNP3-SA CPN model

Through parameterisation, we implement new behaviours (our proposed solutions 1 and 2) in DNP3-SA CPN model. The parameters of our solutions were set to false by default during the initial validation of DNP3-SA CPN. In this section, we present them and set their parameters to true to validate and verify the new DNP3-SA CPN model with the flaw identified.

The behaviour details of proposed solution 1 is presented in Figures 8 & 9. Explanation of these figures are aligned with the MSC provided in Table 4 (HMAC for challenges messages). Transition Secure Challenge Data in $T_{\text{sub}}$ 0SecureCh in Figure 8 uses the elements of the challenge data together with the session key and the current user ID to create a HMAC tag on the challenge data. After the tag has been obtained from place $C_{\text{tag1}}$, the tag is later coupled with the challenge data and sent to the master station via place $\text{SendChlg2}$. In Figure 9, transition RecvCh of the master station receives the secure challenge message (i.e. challenge data and HMAC tag on the challenge). Using the same methodology, the master station computes a tag on the challenge data through transition $\text{ChlgAuth}$ for verification, via transition $\text{tagCheck}$. From $\text{tagCheck}$, if the tag match, a pass token is sent to enable the master station to reply to the challenge, as the outstation is waiting to process the request. Otherwise, the protocol generates a security log indicating unmatched tags for the challenge data and aborts the operation as a result of an attack.

Figure 10 & 11 illustrate the behaviour of the proposed solution 2 in DNP3-SA CPN model. Explanation of these figures are also aligned with the MSC provided in Table 5. In Figure 10, place $\text{CSQ}$ is independent of the NACR operation (refer to top left of the figure). But place $\text{RecentCHM}$ is an active element of both NACR and AGM. The reason as to why $\text{RecentCHM}$ is serving both modes of operation is because, $\text{RecentCHM}$ enables the AGM operation to make use of the most recently received challenge data from the outstation (refer to bottom left of Figure 10). In the same figure (Figure 10), transition $\text{SecureAGM}$ becomes enabled through its incoming arcs only if, an AGM request is about to be sent to the outstation. This causes $\text{SecureAGM}$ to compute an HMAC tag on the request. It is to be noted from the figure (Figure 10), the tag is computed by consuming tokens from $\text{RecentCHM}$, $\text{CSQ}$ and the surrounding places of transition $\text{SecureAGM}$. Then, when the incoming arcs of transition $\text{AGMPack}$ are satisfied, an AGM request (request to be sent and the computed HMAC tag) is sent to the outstation. In Figure 11, the outstation extracts and stores HMAC tags from the master station on
place $\text{StoreAMac}$. Then, when tokens from surrounding arcs of transition $\text{SecureO AG_mode}$ are consumed, an HMAC tag is created. In this process, it is also important to note that the place $\text{CSQ}$ is part of the computation of the AGM tag (refer to lower-left of Figure 11). The final step of this process is the verification of the tags. The tag verification is achieved with a CPN function $\text{AGMverifytags}$ on transition $\text{Tagcheck2}$. That is, if the tags match or fail, a pass or fail token is generated to place $\text{AuthStatus}$. The status of verification is then sent to $T_{\text{sub OPProcess}}$ (Figure 4) for processing.

### 6.4. Validation and Verification of the Proposed Solutions in DNP3-SA CPN

This section presents the analysis of our new DNP3-SA CPN model. Here, we validate the solutions as before to determine if the proposed solutions are
valid. Furthermore, we also verify each of the solutions against the attack model, $MRS^M$. It is to be noted that throughout this analysis, it is expected that we obtain a single dead marking just as previously stated in Section 5.

### 6.4.1. Validation of Solutions 1 and 2

IMCPN in Table 6 depicts the reports of our proposed solutions (refer to Figure 8 and 9, and Figure 10 and 11). For solution 1, a single dead marking (317) with 8 dead transitions were obtained. The report of solution 2 also presents a single dead marking; 434 with 15 dead transitions. Close inspections on both reports revealed that the obtained dead marking and
transitions are consistent with our expectations (as $\text{MRS}^M_0$ is to set `false`).

**Verification of Solution 1**

$\text{MITM}^{\text{CPN}}$, $\text{Sol1}$ in Table 6 presents the report generated for solution 1. The report depicts a single dead marking and 20 dead transitions. As previously stated in Section 5.4, 72 instances of dead markings were obtained and later categorised in two (2) main groups: expected and unexpected. For this revised model, we have obtained a single dead marking (i.e. 317). Using the previous definition of the expected dead markings in Section 5.4, we conduct an investigation on our current results. This is to determine whether the marking obtained in this analysis is expected or not. Thus, in this analysis we expect a `true` value from our reachability statement. A truth value from the statement will imply that the model does not allow the attack model to execute any command to the outstation, after one or two initial requests have been issued by the master station. But, if it is possible for the attacker to execute a command after the expected initial requests from the master station, then it implies that the obtained dead marking is not expected. Henceforth, the authentication property does not hold. Figure 12 presents the results from the queries used. A function with the embedded $\text{SearchNodes}$ was used to capture the state where the attacker is potentially able to ‘fool’ the master station and send AGM requests to the outstation. The query returned the node value; 62. We then use the obtained value (62) with the single dead marking (72) to determine its reachability status. The reachability query has returned `true`; emphasising that an attacker cannot reach a state where it can launch any command to the outstation (hence verifying solution 1). This behaviour indicates the authentication property
holds because the model has detected the attack behaviour. Moreover, as a result of detecting the attack, the model aborts any further operations that will lead to the successful execution of a command on an outstation. As a result of this behaviour, 20 dead transitions were identified not to have occur (including solution 2 because its parameters have been set to false).

### Verification of Solution 2

**MITM**<sub>COPN</sub>, **Sol2** in Table 6 presents the report of our second solution against **MRS**<sup>66</sup>. The report presents a single dead marking (411) and 9 dead transitions. In this analysis, we use the simulation approach to conduct further studies on the obtained results. The sequential execution of the simulation revealed that the single dead marking (411) is consistent with our expectation (hence verifying solution 2). The single dead marking obtained represents the successful execution of legitimate requests from the master station to the outstation; leading to execution of operational transitions on **T<sub>sub</sub> OProcess** (refer to Figure B.20 of Appendix B for details). Other NACR or AGM requests issued by the attack rendered **T<sub>sub</sub> AuthErr** to execute; leading to authentication error messages (refer to Figure B.20 in Appendix B for details). Thus, every attempt to guess or manipulate the challenge data leads to a failure during authentication. The execution of **T<sub>sub</sub> AuthErr** as a result of attempts from the attack model accounts for the identification of only 9 dead transitions in the analysis. These dead transitions represent behaviours of solution 1 (Figure 8 and 9), which have been set to false in the DNP3-SA CPN model.
7. Discussion and Conclusion

The work in this paper has presented detailed behavioural analyses of the DNP3-SA protocol using Coloured Petri Nets. Analysis of the protocol has revealed a flaw, which was not previously been identified during the design phase of the protocol. In our model, we cause a DNP3 slave to replay a previously executed command for certain DNP3 objects. The flaw identified can lead to serious consequences as it allows an attacker to ‘fool’ a responder into producing valid HMAC tag. The tag can later be coupled with the previous request and be replayed as an AGM request to carry out an unexpected events on an outstation. For example, assuming during NACR mode of operation, a write function is executed to either set the time on an outstation for synchronisation of certain crucial data or set an analog deadband value on the outstation. An attacker who is successful in ‘fooling’ the master station into producing a valid tag for that operation can perform the same set of functions in the aggressive mode on the outstation and even write arbitrary values. In effect, the flaw violates the authentication property during the AGM operation.

CPN’s parameterisation methodology helps us combine multiple instances of different behaviours into one model. This enabled us to test for various attack scenarios and develop countermeasures against the flaw identified. We have shown that a CPN model can help developers to identify a flaw and implement the most effective countermeasures. Through our parameterised new models, we have formally evaluated our proposed approaches, which have proven to resolve the flaw identified through: 1) calculating HMAC on challenge data, and 2) implementing the CSQ as a sole active component of the AGM operation. With this work, we have shown the feasibility and power of formal methods in identifying and verifying SCADA protocol flaws. The complexities of initially building the model may be seen as a drawback, however, the model needs to be developed once and later can be used. Moreover, the model reduces the effort required by protocol designers to manually check all possible conditions that may lead to undesirable behaviours.

In the future work, we consider scaling the evaluation of the DNP3-SA over either a multi-drop or data concentrator architecture, which usually involves multiple master stations communicating with many outstations. This is to determine the impact of the flaw on large production systems. We will use the current model as a framework to broaden the scope of our work through parameterisation.
References


Appendix A. Description of Relevant DNP3-SA CPN Model

This section of this paper presents the details of the CPN declarations for the DNP3-SA CPN model to interested readers. Table A.7 presents detailed of CPN declarations of the DNP3-SA CPN model.

<table>
<thead>
<tr>
<th>Remarks</th>
<th>Components</th>
<th>CPN Declarations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requests</td>
<td>FC</td>
<td>Val allrqc = 1’(&quot;0x01&quot;)@1++ 1’(&quot;0x02&quot;)@2++ 1’(&quot;0x03&quot;)@3; Colset fcode = string timed; Var f:fcode</td>
</tr>
<tr>
<td></td>
<td>OH</td>
<td>Val alloh = 1’(&quot;g20v1&quot;)@1++ 2’(&quot;g20v7&quot;)@2; Colset oheader = string timed; Var oh:oheader</td>
</tr>
<tr>
<td></td>
<td>CPN Request</td>
<td>Colset Request = product fcode*oheader</td>
</tr>
<tr>
<td></td>
<td>CPN AGM Request</td>
<td>Colset AGrequest = product Request*hmac</td>
</tr>
<tr>
<td>Responses</td>
<td>FC</td>
<td>Val allrspfc =1’(&quot;0x81&quot;)++ 1’(&quot;0x83&quot;); Colset fcode = string; Var f:fcode</td>
</tr>
<tr>
<td></td>
<td>OH</td>
<td>Colset oheader = string; Var oh:oheader</td>
</tr>
<tr>
<td></td>
<td>IIN</td>
<td>Colset IIN = string; Var IIN</td>
</tr>
<tr>
<td></td>
<td>CPN Response</td>
<td>Colset Response = product fcode * oheader * IIN</td>
</tr>
<tr>
<td>Dynamic States</td>
<td>Eg: Master waiting for response</td>
<td>Colset states = with Ready</td>
</tr>
<tr>
<td>Challenge Data</td>
<td>MAC Algorithm</td>
<td>Colset algorithm = string; Var alg:algorithm</td>
</tr>
<tr>
<td></td>
<td>Sequence Number</td>
<td>Colset seq = int; Var sn:seq</td>
</tr>
<tr>
<td></td>
<td>CPN challenge Data</td>
<td>Colset chlgdata = product seq<em>algorithm</em>PseudoRand; Var cd:chlgdata</td>
</tr>
<tr>
<td>HMAC Tag</td>
<td>Session Key</td>
<td>Colset seckey = int; Var sk:session</td>
</tr>
<tr>
<td></td>
<td>Challenge Data</td>
<td>Colset chlgdata = product seq<em>algorithm</em>PseudoRand; Var cd:chlgdata</td>
</tr>
<tr>
<td></td>
<td>User ID</td>
<td>Val allid =1'; Colset useid = int; Var u:useid</td>
</tr>
<tr>
<td></td>
<td>FC</td>
<td>Colset fcode = string; Var f:fcode</td>
</tr>
<tr>
<td></td>
<td>CPN MAC Tag</td>
<td>Colset mac = record f:fcode* cdc:chlgdata*uu:useid</td>
</tr>
<tr>
<td></td>
<td>CPN HMAC Tag</td>
<td>Colset hmac = product algorithm<em>session</em>mac; Var Mntag, omtag:hmac</td>
</tr>
<tr>
<td>Nonce</td>
<td>Pseudo Random</td>
<td>fun PRandom(p:PseudoRand)=let val(pp) in if p=3 then 1’70 else if p=70 then 1’100 else if p=153 then 1’10000 else if p=10 then 1’6564 else empty end;</td>
</tr>
<tr>
<td>Parameters</td>
<td>Adversary</td>
<td>val Spoof=true</td>
</tr>
<tr>
<td>Verify Tags</td>
<td>HMAC tags</td>
<td>fun verifytags (Mtag; hmac; Otag; hmac)= let val(taM)=Mtag val (tagO)=Otag in if Mtag=Otag then 1’Authpass else 1’ Authfail end;</td>
</tr>
<tr>
<td>Functions</td>
<td>Responses FC &amp; OH</td>
<td>fun Vresp(f:fcode,oh:oheader)= let val(f)=f val(ohh)= oh in if f=&quot;0x02&quot; andalso oh=&quot;g20v1&quot; then 1’(&quot;10—01—10&quot;) else if f=&quot;0x02&quot; andalso oh=&quot;g20v7&quot; then 1’(&quot;11—00—11&quot;) else empty end;</td>
</tr>
<tr>
<td>For Response</td>
<td>Responses IIN</td>
<td>fun OutIIN(f:fcode,oh:oheader)= let val(f)=f val(ohh)= oh in if f=&quot;0x01&quot; then 1’(&quot;00_HI,1&quot;) else if f=&quot;0x02&quot; then 1’(&quot;01_HI,2&quot;) else empty end;</td>
</tr>
<tr>
<td></td>
<td>Response OH</td>
<td>fun Resp(oh:oheader)= let val(ohh)=oh in if oh=&quot;10-01-10&quot; orelse oh=&quot;11-00-11&quot; then 1’(&quot;Written&quot;) else if oh=&quot;11-11-11&quot; orelse oh=&quot;00-11-11&quot; then 1’(&quot;NewValue&quot;) else empty end;</td>
</tr>
</tbody>
</table>
Appendix B. Description of DNP-SA CPN Model Specification

This section presents all relevant third-level pages of the DNP3-SA CPN model to interested readers. Figure B.13 depicts $T_{\text{sub}} \text{SendRequests}$ from the master station. It is the first step involved in the protocol. In this figure, the master station can compose either a NACR or an AGM request, and transmits it through the network to the outstation. Initial composition of requests is through place; Ready, Function Codes, Object Headers and LPN. These places have markings that form the initial state of DNP3-SA CPN model (Bubbles of Figure B.13). The occurrences of the markings create a token on place IniRequests, which has been assigned with a colour set Request. Request is a product of the data type fcode and oheader (representing a DNP3 request, see Table A.7). Similarly, when the incoming arc of transition SendPack is satisfied, tokens are created and respectively distributed to various output places. For example, the presence of a token on place AGMRqsts indicates that the master station may transmit an AGM request after completion of the NACR operation.

Figure B.14 depicts the third step of the specification; $T_{\text{sub}} \text{OutRecv}_{\text{Send}}$. From the left of the figure, the outstation receives NACR requests from the master station through sendRq. The middle part of the figure depicts the outstation receiving AGM requests from the master station. On the far-right, the outstation receives HMAC tags from the master station. The figure also presents that any time a request is received by the outstation, it causes the outstation’s state to change from waiting for requests to processing (i.e. WaitRq to Processing). Similarly, whenever the outstation receives HMAC tags from the master station, we model a mechanism that triggers the outstation to compute its HMAC tags (refer to Figure B.19 for details).

Figure B.15 depicts details of $T_{\text{sub}} \text{CHGenerator}$. It is the fourth process

![Diagram of Tsub SendRequests](image-url)
involved in the specification. It presents the behaviour as to how the outstation generates challenge messages. Transition **Create Challenge Data** is enabled to create a challenge message whenever a token (request) is received through place **Critical Operate**. In creating challenge messages, the transition uses following markings; **HMACalg**, **PseudoR** & **Seqnum** to create challenge data and later forwards it to transition **OFCHI** through place **OCHD1** to be sent to network via place **SendChlg1**. The mentioned markings mark the HMAC algorithm agreed between the stations, the nonce and CSQ (refer to surroundings of transition **Create Challenge Data** in the figure). Places **RecentGH**, **NextCH1** and **NextCH2** model recent challenge messages that can be during the AGM operation. 

T\_\text{sub} \text{OSecureCH} models a similar behaviour of creating challenge messages. However, the behaviour is set to false because it is not yet considered to be part of the DNP3-SA protocol. Table ?? presents how we mimicked the behaviour of the nonce by defining the CPN function **PRandom**. The function randomly generates a number each time a challenge message is about to be issued. Additionally, in order to be consistent with the MSC provided in Figure 1, we have defined certain expressions to the arcs of place **Seqnum** between transition **Create Challenge Data** and F\_\text{ward AGM Challenge Data**}. This is to ensure that for every session, every challenge message issued must have its CSQ updated by 1. Places **RecentCH**, **NextCH1** and **NextCH2** represent the recently issued challenge message for AGM. They are activated only when their arc statements are valid (refer top-center of the figure).

Figure B.16 presents details T\_\text{sub} \text{MasSecure} (refer to Figure 4). It is a SecondLevel page, that consists of three T\_\text{sub}; NACRSec, AGMSec & ChalgAuth. All transitions model the security behaviour of master station. The initial
behaviour of the protocol deals first with \textbf{NACRSec}, before \textbf{AGMSec}. $T_{\text{sub}}$ \textit{Chal-gAuth} runs in parallel with $T_{\text{sub}}$ \textit{OSecureCH}, which is not part of the normal behaviour of the protocol. Therefore, we set it to \textit{false}.

In Figure B.17 (fifth step), the master station receives challenge messages from the outstation through place \textit{sendchlg1} (Lower-left of Figure B.16 & B.17). From the challenge data, the CSQ data is extracted and updated on place $MPrvSeq$, whiles place $Algo$ extracts the HMAC algorithm (refer to surroundings of \textsc{SecureNACR}). Then, place $Chdata$ sends a copy of the challenge
message to transition SecureNACR in order to compute a NACR HMAC tag. When all incoming arcs to the transition are genuinely satisfied, SecureNACR computes the HMAC tag and transmits it via the network through place SendARes to the outstation. The right-hand side of the figure depicts updates of CSQ in challenge messages for AGM operations (i.e. in case the master wishes to transmit aggressive requests). It is to be noted that this behaviour only becomes active after a successful completion of NACR operation. This is achieved through the CPN function NextSeq, NextSeq & NewSeq (refer to Table A.7 for the declaration). As previously emphasised, updating CSQ is considered to be the “most recently received challenge message” from the outstation (according to specification of the protocol). This leads us to T\textsubscript{OutSec}; depicted in Figure B.18.

In the AGM mode (Figure B.18), place AGMRqsts marks initial requests that the master station wishes to execute on the outstation (top of the figure). Place RecentCHM (Lower-left of the figure) marks the most recent challenge message (i.e. updated CSQ). Computation of HMAC tags in AGM is similar to NACR computation, that is, when all incoming arcs (bindings) of transition SecureAGM get satisfied, the transition is enabled to compute an AGM HMAC tag based on the request. Then, transition AGMPack assembles the initial request together with the HMAC tag calculated and transmits it to the network through place SendAGMRq (middle, far-right of the figure).

Figure B.19 depicts details of the sixth step of the specification; T\textsubscript{OutSec} (refer to Figure 4). In this figure, the outstation can either compute a NACR or AGM HMAC tag; depending on which request has to be
processed. By default, every HMAC tag received by the outstation is held in place \textit{StoreHMAC} until the outstation computes its tag and verifies the tags to authenticate the master station (see left-top of Figure B.19). Both transition \textit{Secure0 NACR} and \textit{Secure0 AGM} in this figure use the same method employed by the master to compute their HMAC tag (i.e. when all their respective bindings are satisfied). Any HMAC tag computed by the outstation is held in place \textit{OHMAC} for verification. This eventually enables transition \textit{Tagcheck} to verify both tags for equality. We defined and use the CPN function \textit{Verifytags(Mtag,Otag)} to perform the verification process (Refer to Table A.7 for the declaration). During the verification process, irrespective of the authentication status, a message is generated for the master station through $T_{\text{sub OProcess}}$ (where actual operations are carried out). This leads us to Figure B.20 (i.e. $T_{\text{sub OProcess}}$).

Figure B.20 presents details of the seventh step in Figure 4; $T_{\text{sub OProcess}}$. The figure presents the details of the outstation processing requests and generating responses for the master station; depending on the authentication status. Place \textit{AuthStatus} models the authentication status after the verification of tags from the outstation security mechanism (refer to top of this figure and lower-left of Figure B.19). $T_{\text{sub AuthErr}}$ on the figure (Figure B.20) marks the outstation generating error messages as responses for the master station. This always occurs when authentication fails (unmatched HMAC tags). On the other hand, the label part ‘\textit{Process legitimate requests}’ indicates the outstation processing every requests after a successful authentication. From \textit{AuthStatus}, a pass in authentication (i.e. \textit{Authpass}) enables
transition **Perform Critical** to send authenticated requests for processing. The authenticated request may instruct the outstation to perform certain actions such as deleting a configuration or reading or even writing a current value to the various IEDs. In doing so, the outstation also generates a standard response for the master station. On the other hand, if authentication fails (i.e. **Authfail**), requests are rather sent to $T_{sub}$ **AuthErr** to generate authentication failure messages and send them as responses to the master station. **SendRsp** models the output of messages (responses) from $T_{sub}$ **OProcess** to the master station via the network (Lower-left of the figure).