Securing DNP3 Broadcast Communications in SCADA Systems

Raphael Amoah, *Member, IEEE*, Seyit Camtepe, *Member, IEEE,* and Ernest Foo, *Member, IEEE*

***Abstract*—The Distributed Network Protocol version 3 (DNP3) provides Secure Authentication (DNP3-SA) as the mechanism to authenticate unicast messages from a master station to its outstations in SCADA systems. In large scale systems, it may be necessary to broadcast a critical request from a master station to multiple outstations at once. The DNP3 protocol standard describes the use of broadcast communication; however, it does not specify its security. This paper is the first to present DNP3 Secure Authentication for Broadcast (DNP3-SAB), a new lightweight security scheme for broadcast mode communication. This scheme is based on hash-chain and only makes use of the existing cryptographic primitives specified in DNP3-SA. The scheme integrates itself into the DNP3-SA key update process. The proposed scheme is modelled, validated, verified using Coloured Petri Nets (CPN) against the most common protocol attacks such as modification, injection and replay. Performance analysis on our scheme and the existing DNP3-SA modes (NACR and AGM) shows that DNP3-SAB reduces the communication overhead significantly at the cost of an increase with a constant term in processing and storage overhead. This benefit is maintained even when DNP3-SAB is under attack.**

***Index Terms*—SCADA, DNP3, DNP3-SA, DNP3-SAB, Formal Methods, CPN**

I. Introduction

Supervisory Control and Data Acquisition (SCADA) systems are predominant among the electricity grids, and the emergence of smart grids systems are revolutionising the energy industry [13], [22]. Reports [15] and research [23] on SCADA security have shown an increasing number of cyber incidents and attacks targeting critical infrastructure that use SCADA systems. These incidents and attacks are expected to increase in number and severity because legacy SCADA systems and protocols were not designed to work in insecure environments such as the Internet.

A SCADA network is composed of various components that range from hardware (IEDs, PLC, RTU) to standard protocols such as Profinet [27] and Modbus [24]. The Distributed Network Protocol version 3 (DNP3) [16] is one of the standard engineering protocols for SCADA systems. It is designed to mainly facilitate interactions between outstations and their master stations in an interconnected SCADA network. The DNP3 protocol supports multi-drop and the data concentrator architectures through which large numbers of outstations and master stations from multiple sites can concurrently interact. These types of large scale DNP3 deployments are mainly used for power grid automation. The DNP3 protocol provides a security mechanism called Secure Authentication (DNP3-SA) [11]. This is primarily a unilateral Keyed-Hash Message Authentication Code (HMAC) mechanism used to secure DNP3 messages. DNP3 is one of the first standardised SCADA protocols that attempt to provide a security mechanism.

In [3], [4], Amoah et al. presented formal analysis of DNP3-SA using CPN. The authors constructed models to analyse each of the DNP3-SA operational modes. Each security analysis of the models was primarily limited to unicast communication; one master station and one outstation. Unicast communication is one of the two communication modes supported by the DNP3 protocol [9, p70].

This paper focuses on DNP3 broadcast mode, which is the second of the communication modes supported by the DNP3 protocol. DNP3 Broadcast communication allows a master station to simultaneously communicate with multiple outstations with a single command. One of the advantages of this mode is that it reduces delay and overhead in large scale systems. Broadcast is an important feature for cases of emergencies. For example, an emergency shut-down of stations may be required to be broadcast due to an electrical incident. The DNP3 broadcast mode does not have any security mechanism described in the protocol specification. This gap in the protocol can lead to devastating problems. For example, an attacker who has access to the network may modify commands or execute unauthorised commands simultaneously on multiple outstations to interrupt services.

Hence, the contribution of this paper is two fold. First, we present a new security scheme for DNP3 broadcast mode called Secure Authentication for Broadcast (SAB). Our scheme secures DNP3 broadcast through the use of a one-way hash function (hash-chain) that can only be generated and distributed by the master station and not by any of the outstations. The outstations verify the parameters provided by the master station by using initial pre-distributed credentials. To our knowledge, this is the first proposal for DNP3 broadcast authentication. SAB extends the DNP3-SA protocol by reusing its existing cryptographic primitives with the hash-chain such that, it does not require a major upgrade in existing platforms.

The second contribution of this paper uses Coloured Petri Nets (CPN) to create an executable DNP3-SAB model and formally analyse the security behaviour of our scheme. CPNs [20] are a widely used formalism for designs, specifications, simulations and verifications. Its application in various domains

Manuscript received August 8, 2015. Accepted for publication June 25, 2016.

R. Amoah, S. Camtepe and E. Foo are with the School of Electrical Engineering and Computer Science, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001, Australia. E-mail: {r.amoah, seyit.camtepe, e.foo}@qut.edu.au

such as security protocols [2], [28] have proven to be trustworthy and reliable. This paper uses CPN because it has the ability to conveniently express characteristics such as concurrency, data distribution, asynchronous and non-deterministic features that are inherent in various communication and security protocols. As the DNP3 broadcast mode involves a master station interacting with multiple outstations, the use of CPN's hierarchical structure and parameterisation technique become an important requirement. This is because the hierarchical structure supports a modular approach for modelling, where the behaviour of large systems can be captured at different levels of abstraction. The parameterisation technique enables a number of attacks to be captured for analysis as part of the SAB model without the need to create a new model. By using this CPN tool, we show that our proposed scheme (the SAB CPN model) provides DNP3 broadcast authentication. We also show that our scheme reduces communication overhead at the cost of a minor increase in processing and storage overhead.

This paper is organised as follows. Section II presents the related work. Section III presents the overview of the DNP3-SA protocol. Section IV presents the proposed DNP3-SAB scheme. The CPN modelling of DNP3-SAB and its approach are presented in Section V. The CPN model description is presented in Section VI. The formal analysis of DNP3-SAB CPN model using state space tool is presented and discussed in Section VII. The performance analysis is presented in Section VIII. Finally, Section IX presents the discussions and conclusion of this paper.

II. Related Work

Many of the security flaws found in SCADA are associated with the communication protocols used in control systems. This is because SCADA protocols were not designed with a security mindset. For instance, in 2004 Byres et al. [5] presented the use of attack trees in assessing flaws in the Modbus protocol. The authors showed how to identify and compromise a Modbus device to execute arbitrary requests. Later in 2008, Huitsing et al. [14] presented an attack taxonomy on the Modbus protocol, where some of the attacks could be used to exploit Modbus's assets (serial and TCP) in various ways. In the subsequent year (2009) East et al. [10] presented a taxonomy of attacks on the DNP3 protocol before DNP3-SA was adapted from IEC 62351 [17]. Moreover, in the same year (2009), Akerberg and Bjorkman [1] explored flaws in the Profinet protocol. Similarly, in 2011, Cheminod et al. [6] presented a formal vulnerability analysis of the Fieldbus access protocol, where they presented flaws of the protocol that might be susceptible to potential replay attacks. In 2014 and 2015 respectively, Amoah et al. [3], [4] presented formal analyses of the DNP3-SA protocol using the CPNTools [18],where the authors also presented new flaws in the DNP3-SA protocol.

Many of the standardised SCADA protocols support different types of communication modes; however, traditional security solutions and SCADA security proposals are mostly limited to unicast. This paper focuses on security of DNP3 broadcast communication. Broadcast communication and its security have gained wide attention in the sensor network domain. Some of the solutions from sensor networks have been adapted to secure various communication modes in SCADA systems. Relevant to the topic of this paper, Choi et al. proposed an advanced key-management architecture for unicast and broadcast communications in SCADA systems [7]. The proposed scheme is extended to an efficient secure group communication scheme in SCADA systems [8]. Later in [21], it has been shown that solutions by Choi et al. are only feasible with the aid of trusted entities that facilitates key establishment between SCADA entities. Though, these schemes prove promising, they are constrained with availability issues. Each trusted entity forms a single point of failure; meaning that, failure of even one trusted entity may threaten continuity of the whole security process.

These schemes cannot be directly applied to DNP3 and other SCADA protocols. This is due to their specific architecture and application limitations. Moreover, existing devices with DNP3 capabilities have limited support for cryptographic methods (i.e., SHA-1-HMAC, SHA-256-HMAC, AES-GMAC, and AES-128 key wrap) due to their resource limitations. This requires the proposed SAB solution to be compliant to these cryptographic methods only.

Identifying the gap of missing authentication for DNP3 broadcast communication, this paper employs the hash chain concept from the TESLA protocol [25] (Timed Efficient Stream Loss-tolerant Authentication) to secure DNP3 broadcast messages in SCADA systems. TESLA is a broadcast authentication protocol that uses hash chains and time delayed key disclosure. TESLA generates a key chain by using the relation of $H(Sk_i) = Sk_{i-1}$. Sender uses Sk_i to authenticate a broadcast message M_{ti} sent at time interval t_i. The broadcast message also includes the key used in the previous time interval (e.i., $M_{ti}||Sk_{i-1}$). That means, the key Sk_i will be disclosed along with a message in the coming time intervals, e.g., time interval t_{i+1}, and the message M_{ti} must be stored until then. Disclosed key Sk_i can then be used to authenticate the stored message M_{ti}. Moreover, the disclosed key Sk_i can be checked for its validity using its relation with the keys disclosed earlier (e.g., $H(Sk_i) = Sk_{i-1}$ or $H(H(Sk_i)) = Sk_{i-2}$). TESLA is considered secure because it allows all receivers of a given broadcast or multicast message to verify its integrity and authenticity. Moreover, it is considered efficient because of its low communication and computational overheads. It scales well on a large number of receivers. However, these benefits of TESLA come at the cost of storage overhead because received messages must be stored until the relevant keys are disclosed and validated.

TESLA and its derivatives [26] cannot be directly applied to the DNP3 protocol due to their domain differences and the nature of the DNP3 protocol. That is, TESLA and its derivatives depend on time synchronisation and the revelation of keys. Additionally, TESLA authenticates its initial packet with a digital signature, which can be seen as costly for SCADA systems because SCADA devices are constrained with processing power. Though, time is a crucial factor in SCADA systems, our approach to secure SCADA systems using the DNP3 protocol does not depend on strict time synchronisation as TESLA does. Moreover, DNP3 provides an authentication scheme for unicast communication (DNP3-SA) that makes use of certain cryptographic primitives and has a user and key management scheme that updates keys regularly after a certain period of time (say minutes up to weeks depending on the rate of communication) [16]. Hence, this paper presents a broadcast authentication scheme based on hash-chains and message authentication codes which integrates with the existing user and key management scheme of DNP3-SA without needing any major upgrade. Additionally, this paper presents a CPN model to validate and verify the security behaviour of SAB with adversarial settings.

III. Overview of the DNP3-SA Protocol

DNP3 is a layered and non-proprietary protocol, which is designed to facilitate communication between master stations and outstations via serial-line or TCP/IP protocols through encapsulation. DNP3-SA is the embedded security mechanism in the application layer of the DNP3 protocol. Primarily, DNP3-SA uses an HMAC mechanism with a secret key, k (i.e., 128 bits) to authenticate critical messages transmitted between interconnected stations. A critical DNP3 message can either be a request from a master station or an unsolicited response from an outstation that contains a mandatory code. A mandatory code is defined as any code that can control stations or set crucial parameters such as power grid voltages and frequencies. This implies that any station that sends a message which includes any mandatory code will be challenged by the receiving device to prove its identity. Authentication in DNP3-SA is unilateral but the mechanism operates in two different modes: two-pass authentication (NACR) and one-pass authentication (AGM) through the HMAC. However, the AGM operation is also dependent on NACR. This means that AGM can only be in-use if there has been one or more occurrences of NACR operations. This behaviour enables the AGM to make use of "the most recent challenge message" from the NACR operation.

An in depth view of the protocol is provided later in Section IV of this paper (i.e., when providing the SAB scheme). But to facilitate the understanding of the protocol, the operation of NACR and AGM are presented in Figures 1 and 2 respectively. It is important to note that in these figures, it is assumed that session key, Sk_{mo}, for each user have been securely established between the master station and the outstation (this will further be explained in Section IV). In Figure 1, it is depicted that the master station sends a request to an outstation. At the receipt of the message, the outstation sends a challenge message, which contains an algorithm (H), nonce (P) and a challenge sequence number (KSn) to the master. The master uses the H algorithm from the challenge received and the shared-key (Sk_{mo}) on the entire challenge data together with the request to create a MAC tag. The MAC tag is then sent to the outstation. On receiving the MAC tag, the outstation calculates its MAC tag by using the same computational approach and elements used by the master station. A match in the tags indicates an authentication success and leads to execution of the request. Otherwise, the outstation discards the request and sends an error message to the master station.

In Figure 2, a number of challenge-response messages are eliminated. It is assumed that the master station has the "most recent challenge message" from the most recent NACR operation. Taking the most recent challenge message into account, the master is always required to increase KSn from the challenge by 1 for all AGM operations before a legitimate AGM message can be composed. Figure 2 depicts the master station sending an AGM message to the outstation. The AGM message consists of a request, and a MAC tag. The MAC tag is computed by using the algorithm (H), the session key (Sk_{mo}), the "most recent challenge message" and the request. On receipt of the AGM message, the outstation temporarily stores the MAC tag, extracts relevant data, computes its own tag and verifies if the tags match. A successful verification will lead to processing the request and replying with a response message. A failed verification will lead to an error message.

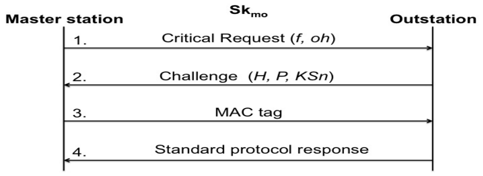


Fig. 1: DNP3-SA NACR operation.

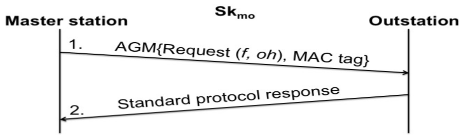

Master station
Sk_mo
Outstation
1. AGM{Request (f, oh), MAC tag}
2. Standard protocol response

Fig. 2: DNP3-SA AGM operation.

IV. Secure Authentication for Broadcast (SAB)

The DNP3-SA protocol works well under unicast communication, but it cannot be applied to broadcast communication. This is because as DNP3-SA uses the challenge-response approach, the master station will need to store, remember and exchange a number of challenges and response messages with each outstation. This is not practical because there will be delays and an increase in communication, processing and storage overheads. However, in the absence of an appropriate broadcast authentication mechanism, it is possible for an attacker to broadcast unauthorised commands to be performed by all outstations at once. This leads us to design a new security scheme, SAB, to secure broadcast mode communication. A detailed description of SAB is presented in Figure 3.

A. *Overview of SAB*

The concept of SAB is derived from the TESLA protocol presented by Perrig et al. [25]. SAB is a minimalist broadcast authentication protocol designed only to use existing cryptographic primitives of DNP3-SA (i.e., SHA-1-HMAC, SHA-256-HMAC, AES-GMAC and AES-128 key wrap) so that existing DNP3 devices can be made compatible with minimal software upgrades. Intuitively, SAB operates through a hash-chain, which is a linked hash values known to master station only. Using the pre-distributed credentials, the outstations can only verify the current hash value released with the current broadcast message but they cannot produce the hash value before it is released. Limiting this feature to only the master station implies that broadcast messages are generated by the master station and they are correct. This therefore, effectively secures DNP3 broadcast communication against modification, spoofing, replay, and injection attacks. This approach is better than some existing schemes because our scheme solves the key management problem associated with the HMAC mechanism. Thus, instead of the HMAC mechanism using different keys for each broadcast message, our scheme uses a hash-chain with a single symmetric key. Moreover, our scheme is not constrained by time like other schemes based on TESLA and most of its derivatives. Moreover, SAB does not require infrastructure like a Public Key Infrastructure (PKI), where processor intensive algorithms like RSA are used. A comparison of performance and functionality of SAB using other alternative methods is presented in Section VIII.

B. Operation Details of SAB

Figure 3 presents the detailed behaviour of SAB. To understand the table, the following notations are worth noting. Master Station and Outstations represent the communicating entities. Sk_{mo} and $Sk^{'}_{mo}$ respectively represent the old and new session keys of a particular user, which are used to provide entity authentication for the data transmitted in the control direction (i.e., from master to outstations). Both session keys are derived from the long-term secret key, UpK. UpK is traditionally manually distributed to all the entities in the control system. R is a random bit-string with sufficient length, such as 128 bits, to prevent trivial pre-image attacks. H represents the hash function, and is used to compute the hash chain tags: $S_1 = H(R), S_2 = H(S_1), \ldots, S_n = H(S_n - 1)$ which represent hash chain tags that are produced from a single random value (R). S_n represents the last hash tag of the hash chain while n represents the index of S_n. KS and $KS^{'}$ represent challenge messages that contain old and new challenge data respectively. Each challenge contains a Key Change Sequence Number, KSn, Message Authentication Code (MAC) algorithm, H and a Nonce, P. E and D represent symmetric key encryption and decryption schemes, respectively agreed between all the entities. C (ciphertext) represents encrypted $(Sk^{'}_{mo}, S_n, n)$. $MAC_{Sk_{mo}}$ and $MAC_{Sk^{'}_{mo}}$ represent agreed MAC schemes with keys. tag and tag_A represent MAC tags generated by the master station while $tag^{'}$ and $tag^{'}_A$ represent MAC tags generated by outstations. KSR represents a request, $(FC)_{Req}$, that contains a user's key change data. KC represents the concatenated ciphertext, C and the MAC tag, tag. KSC represents a response that contains the status of the user's key update, $(FC)_{Resp}$, and a new challenge message, $KS^{'}$. BM represents a broadcast message that contains a request, $(FC, OH)_{Req}$, a hash tag, $\mathbf{S_i}$, the index of the hash tag, $\mathbf{i}$ and the MAC tag, tag_A. $ErrorResp$ represents an error response. U_{ID} represents a User Identification Number associated with the communicating entities. $\mathbf{S_{stored}}$, $\mathbf{N_{stored}}$, $\mathbf{S_{received}}$ and $\mathbf{N_{received}}$ store value from $\mathbf{S_n}$, $\mathbf{n}$, $\mathbf{S_i}$ and $\mathbf{i}$ (where $1 \leq \mathbf{i} < n$), respectively. $\mathbf{S_{calc}}$ is the variable used to calculate the next hash tag of the hash-chain. In this scenario, it is assumed that the necessary algorithms and session key, Sk_{mo} are pre-established between M and O, but the session key is outdated. Therefore, M wishes to update the old session key to a new key. But before M updates the key, it is required that M pre-creates a hash-chain for future broadcast communication. In creating the chain, M is required to choose an arbitrary number R (let's say 128-bits) and creates a series of hash tag values using hash chains as illustrated in step 1 of Figure 3. After creating the chain, M can now send a key status request, KSR, to all O to update the session key of a particular user (**step 2**). At the receipt of KSR, all O reply with key status message, KS, which contains the challenge data (H, P and KSn, from **step 3**). At the receipt of KS, M uses the pre-shared key, UpK, to encrypt the new session key, $Sk^{'}_{mo}$, and the last hash tag of the hash chains, $\mathbf{S_n}$, together with associated its index, $\mathbf{n}$ (**step 4**). Furthermore, M uses H from KS, and the existing (old) session key, Sk_{mo}, to keyed-hash KS and KSR in order to compute the MAC tag, tag (**step 5**). Then, M sends KC, which contains ciphertext, C, and the MAC tag, tag, to all O (**step 6**). On receipt of the message, all O first compute their MAC tag, $tag^{'}$ by using the same computational elements used by M, and then checks if the tags match (**step 7**). If the tags match, then all O performs the second action by decrypting KC using UpK and storing the contents ($Sk^{'}_{mo}$, $\mathbf{S_n}$, $\mathbf{n}$ refer to **step 8**). At this point, all O can respond to M with KSC to indicate that the new session key ($Sk^{'}_{mo}$), as well as the last hash tag value of the chain and its index are stored (**step 9**). Furthermore, KSC also includes a new challenge message, $KS^{'}$, that M uses to re-authenticate itself to all O in case a new key and hash updates (new hash chain) are required (**step 1–9**). It is worth noting that $KS^{'}$ is a variable of the DNP3-SA protocol and it is used for challenging the master station in the DNP3-SA protocol. $KS^{'}$ is not used in SAB but it is maintained for compatibility reasons with the DNP3-SA protocol. Analysis of the DNP3-SA protocol can be found elsewhere [3]. At this point, it is important to note that all stations (M and all O) are now set to communicate with normal DNP3 messages.

In the part labelled **Continuation of New Scheme**, we make use of the AGM behaviour as explained in Section III. That is, before any broadcast message can be sent, M must first compute a MAC tag based. Computation of the MAC tag is based on the request about to be sent, a hash tag from the hash chain and the index value associated with the tag. It is to be noted that the index is always linked to a particular hash value chosen. In computing tag_A, M uses the agreed MAC scheme with the recent established key, $Sk^{'}_{mo}$, on the message $(\mathbf{S_i}, \mathbf{i}, (FC, OH)_{Req}, U_{ID})$. After tag_A is obtained, M sends BM (broadcast message), which constitutes $(\mathbf{S_i}, \mathbf{i}, (FC, OH)_{Req}, U_{ID})$ and tag_A to all O. On the receipt of the message, all O first check if i associated with hash tag, S_i is valid (non-negative value); and i is less than what is previously stored ($\mathbf{S_{stored}}$). There are two main reasons why this action (i checking) is required: 1) to prevent potential replay attacks of a previously recorded message with a valid MAC tag and 2) to take into consideration subsequent rounds

Fig. 3: A New Security Scheme for Broadcast Mode Communication

Master station $(UpK,\, Sk_{mo})$		**Outstations** $(UpK,\, Sk_{mo})$
	Beginning of New Scheme	
(1) $R \leftarrow \{0,1\}^*$ $S_1 = H(R), S_2 = H(S_1), \dots, \mathbf{S_n} = H(\mathbf{S_{n-1}})$		
	Existing Key updating Process	
(2) $KSR = (FC)_{Req}$	$\xrightarrow{KSR}$	
	$\xleftarrow{KS}$	**(3)** $KS = (H, P, KSn)$
(4) $C = E_{UpK}(Sk^{'}_{mo}, \mathbf{S_n}, \mathbf{n})$ **(5)** $tag = MAC_{Sk_{mo}}(KS, KSR)$ **(6)** $KC = (C, tag)$	$\xrightarrow{KC}$	
		Verifying MAC tags for Key Update **(7)** $(tag^{'} = MAC_{Sk_{mo}}(KS, KSR)$ If $(tag == tag^{'})$ **(8)** $(Sk^{'}_{mo}, \mathbf{S_n}, \mathbf{n}) = D_{UpK}(C)$ $\mathbf{S_{stored}} = \mathbf{S_n}; \mathbf{N_{stored}} = \mathbf{n}; Sk^{'}_{mo}$ $KS^{'} = (H^{'}, P^{'}, KSn^{'})$
	$\xleftarrow{KSC}$	**(9)** $KSC = ((FC)_{Resp}, KS^{'})$
	Continuation of New Scheme	
$1 \leq \mathbf{i} < n$ **(10)** $tag_A = MAC_{Sk^{'}_{mo}}(\mathbf{S_i}, \mathbf{i}, (FC, OH)_{Req}, U_{ID})$ **(11)** $BM = \{\mathbf{S_i}, \mathbf{i}, (FC, OH)_{Req}, U_{ID}, tag_A\}$	$\xrightarrow{BM}$	
		Verification Process $\mathbf{S_{received}} = \mathbf{S_i}, \mathbf{N_{received}} = \mathbf{i}$ **(12)** if $!(1 <= \mathbf{N_{received}} < \mathbf{N_{stored}})$ {Error Resp} Step 1: Check Authentication **(13)** $tag^{'}_A = MAC_{Sk^{'}_{mo}}(\mathbf{S_i}, \mathbf{i}, (FC, OH)_{Req}, U_{ID})$ If $(tag_A ! = tag^{'}_A)$ {Error Resp}
		Step 2: Check Freshness $\mathbf{S}_{calc} = \mathbf{S_{received}}$ **(14)** For $(x = \mathbf{N_{received}}; x < \mathbf{N_{stored}}; x++)\{$ $\mathbf{S}_{calc} = Hash(\mathbf{S_{calc}})\}$ if $(\mathbf{S}_{calc} == \mathbf{S_{stored}})\{$ $\mathbf{S_{stored}} = \mathbf{S_{received}}; \mathbf{N_{stored}} = \mathbf{N_{received}};$ **(15)** Process $BM\ [(FC, OH)_{Req}]\}$
	$\xleftarrow{ErrorResp}$	Else{Error Resp}

of authentication for legitimate broadcast messages issued by the master station. For the replay attack, assuming i is valid, then all O will continue to perform a MAC tag verification for message authentication (**line 13**), otherwise the operation is aborted. In checking for message authentication, all O compute $tag^{'}_A$ and verify them against tag_A received from the master. A match implies success, which further leads all O to check for freshness (the status of hash tag received, **line 14**) before the broadcast request can be processed. In checking the hash tag, all O use i ($\mathbf{N_{received}}$) to determine the number of hash iterations that must be performed on $\mathbf{S_i}$ to obtain the final hash tag result, $\mathbf{S_{calc}}$. $\mathbf{S_{calc}}$ is then compared with the stored hash tag, $\mathbf{S_{Stored}}$. If the hash tags match, then, $\mathbf{S_{calc}}$ is stored as the new hash tag value ($\mathbf{S_{Stored}}$), and the broadcast request ($BM[(FC, OH)_{Req}]$) is processed. However, if the tags do not match, then $\mathbf{S_{calc}}$ is discarded and an error response is sent to M. This action or behaviour applies to multiple rounds of authentication within SAB. Thus, since every hash tag is linked to an index, it implies that every broadcast message will have a unique hash tag and its associated index for every round of authentication required. Therefore, for all rounds of authentication if the indexes are not valid, the outstations will respond with error messages just as before in the replay attack. But, if the indexes are valid in each round, then the outstations are required to perform the MAC tag verification (**line 13**). If the MAC tags verification do not match for each round, the outstations will respond with error messages or abort the operation. Otherwise, the outstations will continue to verify the status of all hash tags for each round (**line 14**) before the broadcast request can be further processed. A successful hash tag verification in each of the rounds will lead to discarding the old hash tag and storing the new hash, and processing the broadcast request as expected. But failures in each round will lead to an error message.

V. CPN MODELLING OF THE DNP3-SAB PROTOCOL

A CPN model is a bipartite graph that comprises of two forms of nodes, *places* (drawn as ellipses) and *transitions* (drawn as rectangles), and directed edges that are known as *arcs* that connect places to transitions. The places and transitions respectively represent states and events in a given model. They are associated with *colour sets* (simple and compound colour sets) that enable a place to store data (*token(s)*). Simple colour sets are made of data types such as *string*, *integer* and *boolean* values. Compound colour sets (*product*, *record* and *union*) are colour sets that are constructed on simple coloured sets. For example, `colset Eg = product string * integer`. The distribution of a token from one place to another through a transition represents a *state change*. As previously, we use the CPN tool mainly because of its ability to conveniently express characteristics such as concurrency, parallel, data distribution, asynchronous and non-deterministic features that are inherent in many systems. With such abilities, different level of abstractions of the protocol can be captured in order to validate and verify the correctness of our scheme. In addition, essential security primitives and behaviours of SAB can be captured and represented as "black-boxes" through the CPN functional programming language Standard ML (SML) [12]. This is because SML creates the flexibility to virtually capture different types of data or cryptographic parameters for various types of operations. The effectiveness of CPN is also supported by the embedded state space tool [19], which can be used to perform validation and verification analysis through a summary report that is provided after the execution of the CPN model.

A. Our CPN Modelling Approach

To understand the CPN formalism, this section presents specific processes involved in modelling the DNP3 broadcast protocol with SAB. These processes are useful for representing various data such as requests and responses, cryptographic primitives and operations as well as other behaviours such as adversaries. The processes include 3 steps namely:

Protocol Primitives Abstraction: This approach is used to capture various data such as requests and responses and security primitives (cryptographic keys, MAC tags, hashes) from the DNP3-SAB protocol. In capturing these primitives and behaviours in CPN, we needed to first represent data as CPN colour sets and then address their operations by using CPN SML functions. As previously stated, colour sets (both simple and compound) are made of data types. Therefore, representing data (requests, responses and security parameters) with these colour sets create the flexibility and inclusiveness to virtually capture different types of cryptographic parameters for various types of operations required. For example, the CPN *record* and *product* data types help encode relevant data needed to represent a cryptographic parameter. Relevant to this paper, `colset Request=product of fcode * oheader` is an example of the CPN colour set used to represent the protocol initial request from the master station. Here, *fcode* and *oheader* are both a string data type. This implies that an integer request will be void because the model is only accepting specific string values as requests. On the other hand, we use CPN's SML functions to simulate the operations of the primitive captured. But it is worth noting that SML functions used are mostly symbolic rather than the real operation. For example, the outstation's way of generating appropriate responses based on a particular request from the master, does not perform the actual generation of responses expected from the protocol specification. Rather, the SML functions only mimic the behaviour of the protocol by using certain restrictions. This approach helps one to reuse functions, which also leads to obtain a clean and precise model.

Hierarchical Net-structure: In this paper, we choose to model a single master station that is concurrently communicating with 3 outstations. This configuration is set-up to facilitate the understanding of the paper and model can be extended to support more stations. To achieve this configuration, we use the CPN hierarchical technique to hide the details of the net structure in order to simplify complexities. This approach is suitable for creating large models because the behaviours of entities or security operations can be represented as substitution transitions (rectangular-shaped but with double outline). Each substitution transition may link to at least a sub-net structure page, where detail of the net structure can be found. This makes it possible to capture different levels of abstraction in the same CPN model. For example, we demonstrate this approach in Figures 4, 5, 6, 7 and 8), where Figures 4 present the main page (top-level) while Figure 5, 6, 7 and 8) present details of some substitution transition in our main page. This paper presents pages (top and third-level pages) of the model that are relevant to the contribution of this paper.

Model Parameterisation with Adversaries: Relevant to this paper, this approach is used to avoid the tendencies of creating different models to capture different behaviours. Thus, using this approach enabled us to construct a CPN model such that it is also possible to incorporate different types of adversarial behaviours, where one or more of the attack behaviours can be enabled or disabled depending on the environment for analysis (i.e., validation or verification). There are many types of attack that could be launched, but we scope this work to only consider common Internet attacks such as modification, replay, injection. The detail of the attack models is provided in Section VII-B.

VI. MODEL DESCRIPTION

A. Model Assumptions

Before the CPN model for DNP3-SAB is presented, it is important to note that some assumptions were considered while modelling. These assumptions are made to help avoid the possibility of generating large state space values and they reduce complexity in constructing the model. Some of these assumptions include:

- The master station communicates with 3 outstations.
- All default algorithms (i.e., long-term secret key, session keys, MAC scheme) used in DNP3-SA among stations have been pre-established.
- The underlying layers of the DNP3 protocol are reliable to ensure that there no communication failures.

- Not all DNP3 and DNP3-SA data variations are captured. This is because some data variation do not contribute to the security analysis of this paper. Examples of such variations include binary, analog inputs and outputs data.

B. The CPN model of DNP3-SAB

Taking the above assumptions into consideration, the DNP3-SAB CPN model is constructed in 3 different level of details or pages. The pages were created to improve the readability of this paper. These 3 pages include the top-level, second-level and third-level pages. The top-level page is the main abstract view of our CPN model. The second-level page shows brief details of the top-level while the third-level pages present the actual details of the modules (substitution transitions). Because of space constraint, top and third-level pages that directly contribute to this paper are presented; Figures 4, 5, 6, 7 and 8.

Figure 4 presents the top-level page of the DNP3-SAB CPN model. The model represents one master station communicating with three outstations for the purpose of illustrating and evaluating the contribution of this paper using sized models with sufficient details. The behaviour of the master station (Figure 4 - Left), network (Figure 4 - Middle) and outstations (Primary Outstation A, B & C, Figure 4 - Right) are presented with substitution transitions (i.e., double-lined square boxes). A substitution transition is a high-level module that is used to hide the detailed operation of a net structure. For example, the Network substitution transition hides the details of the network activities between the stations (see Figure 5). Remaining `SendA`, `sendRq`, `TrigRep`, `DataB`, and `sendRsp` places model the transmission of data via the network between the stations. The `ctrlkey` and `UserID` places capture users and their associated keys. Figure 5 presents the third-level page of the network in our CPN model (the *Network* substitution transition in Figure 4). Figure 5 presents two behaviours; network behaviour and parameterised attack models. The network behaviour specifies the way that the master station and the outstations exchange messages. In the Figure 5, the network behaviour is modelled with the `SendA` and `sendRq` places and the *ConnectA* transition. These places and transition allows the master to send data to the outstations. The colour sets (*colset Packet* and *PacketRsp*) represent all data (request, response and cryptographic parameters) exchanged between all the stations. The labelled part "**Receiving Responses**" of Figure 5 forms part of the network behaviour, where all outstations respond to requests, if required. The parameterised attack models mark the adversarial behaviours on the network. These behaviours include attacks replay, injection and modification. These attack behaviours are set to false (turned OFF) and will be discussed later in Section VII-B.

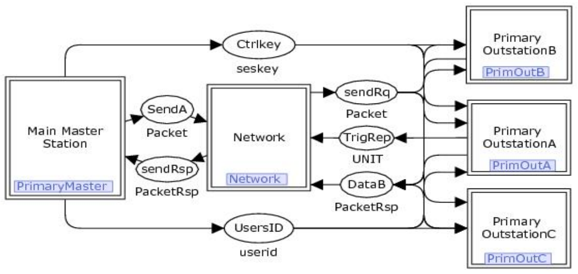


Fig. 4: DNP3-SA CPN on Multi-drop – Top-level page.

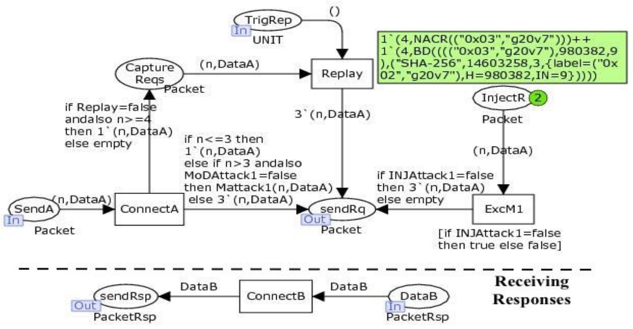

TrigRep
UNIT
()
Capture Reqs
Packet
(n,DataA)
Replay
1`(4,NACR(("0x03","g20v7")))++ 1`(4,BD(((("0x03","g20v7"),980382,9),("SHA-256",14603258,3,{label=("0x02","g20v7"),H=980382,IN=9}))))
InjectR
Packet
if Replay=false andalso n>=4 then 1`(n,DataA) else empty
3`(n,DataA)
if n<=3 then 1`(n,DataA) else if n>3 andalso MoDAttack1=false then Mattack1(n,DataA) else 3`(n,DataA)
(n,DataA)
if INJAttack1=false then 3`(n,DataA) else empty
SendA
Packet
ConnectA
sendRq
Packet
ExcM1
[if INJAttack1=false then true else false]
sendRsp
PacketRsp
DataB
ConnectB
DataB
PacketRsp
Receiving Responses

Fig. 5: The network and attack models.

Figure 6 presents the third-level page of all the outstations processing requests. It is assumed that all outstations will be processing the same request. We model the broadcast behaviour of all outstations with the *T_write* transition from the *Sub Processes* substitution transition, the *SenSor* and *Enabler* transitions, and the `Actuator1` and `Binary` places. These places and transitions are used to capture the executions of all broadcast operations. For example, the *SenSor* transition must execute the token value "`Rlsed Pressure`" three times (3x) when the appropriate broadcast request for this operation is received.

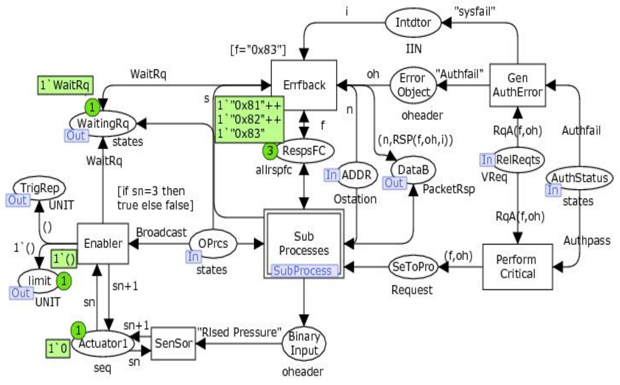


Fig. 6: Processing requests on all outstations.

Figure 7 depicts the behaviour of SAB on the master station. The `HashChain` place presents all the generated hash tag values and their respective indexes (the left-down bubble in Figure 7) using one of the DNP3-SA hash function specified in its standard. This behaviour represents step 1 in Figure 3. The `MAC` place models the MAC token (values) obtained from the *BSecure* transition (refer to the inputs of *BSecure*, representing step 10 in Figure 3). The `BdRequest`, `HValue` and `Index` places model the actual broadcast request about to be sent, the hash tag from the hash chain and the index of the hash tag, respectively. These tokens are grouped as a single broadcast message through the *BPackA* transition and transmitted to the network through the *SendRQ* place. This captures the behaviour (step 11) illustrated in Figure 3.

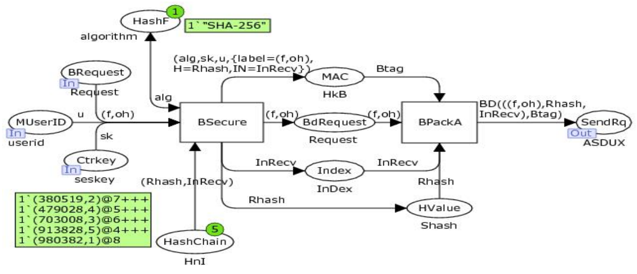


Fig. 7: The behaviour of SAB on the master station.

Figure 8 presents the behaviour of SAB on the outstations. The *Stored Values* substitution transition models the storage area for valid hash tags and their indexes. This includes the last hash tag from the hash chain, which is assumed to be securely established before the communication began between the stations (to the centre of the Figure 8). The *BrdCast Requests* transition models the receipt of every broadcast message from the network through the `sendRq` place (lower left of the Figure 8). The *Check Index Value* transition models the behaviour for checking whether the incoming index values of the hash tag are valid. This transition is there for subsequent rounds of authentication and to avoid replayed messages. If the received index of the hash tag is valid, then a token is placed on the `Trigger` place to verify the authenticity of the broadcast message. Otherwise, a token is placed on the `AuthStatus` place to signify an error (to the centre-right of the Figure 8). This behaviour reflects the verification process (step 12) in Figure 3. Moreover, the *ECV* substitution transition models the behaviour, where the received hash tag from the master station is extracted and hashed a number of times to obtain a particular hash value. This behaviour represents step 14 in Figure 3. The verification of MAC tags for message authentication and hash tags for freshness is achieved through the *Tagcheck* transition; representing step 13 and 14 in Figure 3.

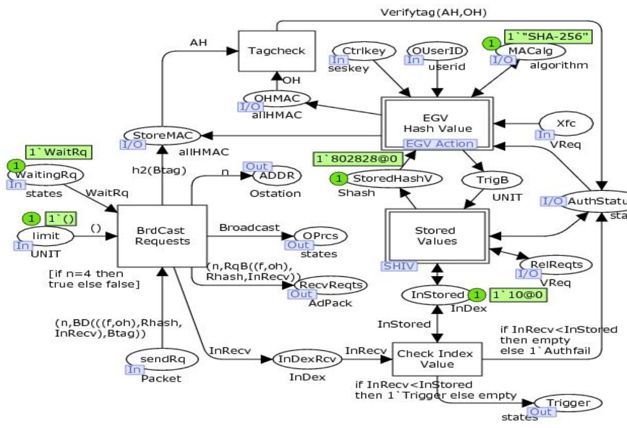


Fig. 8: The behaviour of SAB on all outstations.

VII. The Formal Analysis of DNP3-SAB CPN model

In this section, we present the validation and verification of the DNP3-SAB CPN model under the set-up of 1 master to 3 outstations by using CPNs state space tool. This set-up provides a model with a proper size for illustration and with sufficient details for security analysis. However, a generic performance and overhead analysis showing how the proposed solution scales with increasing number of outstations is presented in Section VIII.

The state space tool provides relevant information on how the model proceeded from one state to another in consistent manner. This vital information is presented in a summary report with a number of behavioural properties. Some of these properties include the nodes and arcs of the state space, the Strongly Connected Components (SCC), live transitions, home markings, dead markings and dead transitions. The state space nodes and arcs are properties used to determine the number of nodes and arcs involved in the entire model. Likewise, SCC is a property used to determine the reachability of nodes and iterations involved in the model. This paper is more interested in dead markings and dead transitions. A marking or state is declared dead when there exist no binding elements to make the state active. In our model, a dead marking signifies a termination point. Similarly, a transition is declared dead when the transition has no path from reachable states to enable it. A dead transition in our model may imply that an expected or unexpected action may have occurred. It is to be noted that dead transitions in our model are subject to change. Depending on the behaviour to be validated or verified, the number of dead transitions might vary because certain parameterised behaviours might be set to *true* or *false*. For example, if all attack models are set to *false* (turned OFF) while validating the model may generate a large number of dead transitions.

A. Validation of the DNP3-SAB CPN model

The aim of this section is to validate the DNP3-SAB CPN model without any attack model turned on (i.e., *true*). The purpose of this action to determine whether the model reflects the DNP3 broadcast behaviour expected from the protocol specification. In this analysis, it is expected to obtain a single dead marking as the result of a single initial request from the master station to all outstations. This single dead marking will indicate that every request issued by the master station, is received and processed simultaneously by all the outstations. Additionally, it is also expected to obtain 37 dead transitions. These transitions will consist of parameterised attack models that have been set to *false*, inactive DNP3-SAB operations (because we are validating), and security transitions such as *Errfback* and *Gen AuthError* that could not execute, as there are no attacks enabled.

Table I presents the full state space report of the DNP3-SAB CPN Model. The table has two parts; Initial DNP3-SAB CPN Model and Attack Models on DNP3-SAB CPN model. The Initial DNP3-SAB CPN model presents the validation of the SAB model while the latter is the verification of DNP3-SAB against the most common attack models (this is presented in the subsequent section).

In Table I, Initial DNP3-SAB CPN Model the report presents identical values obtained for nodes and arcs of the state space and that of the SCC. Achieving these matching

TABLE I: State Space Analysis of SAB on Multi-drop

State Space Report for CPN SAB Model					
Initial SAB CPN Model		**Attack Models on SAB**			
		Injection	*Modification*	*Replay*	
State Space Nodes	13,829	23,831	821	44,789	
State Space Arcs	50,117	50,120	1,045	89,056	
SCC Graph Nodes	13,829	23,831	821	44,789	
SCC Graph Arcs	50,117	50,120	1,045	89,056	
Dead Markings	1	3	1	3	
Dead Transitions	37	25	53	22	

values for both properties implies that our model has no loops and moreover the model has a finite sequence of occurrence. Furthermore, the report presents a single dead marking with 37 dead transitions. These values (single dead marking and 37 dead transitions) are consistent with our expectation in Section VII. This is because a close inspection of the model revealed that the single dead marking represents the state where a broadcast request from the master station has been successfully authenticated and processed by all outstations. As a result, it led to the execution of the *Sensor* transition, which further led to overwrite the default token value of '`0`' to '`3`' on the `Actuator1` place on all outstations (refer to the bottom of Figure 6). Additionally, the inspection also revealed that all 37 dead transitions were consistent with our expectations. This is due to the absence of attack models and certain security operations such as *Errfback* and *Gen AuthError* transitions, which model authentication failure operations that could not be executed as there are no attacks enabled. In summary, these behaviours are expected as SAB is enabled and no attack models have been enabled.

B. Verification of the DNP3-SAB CPN model

This section presents the evaluation of DNP3-SAB CPN model with adversaries enabled (i.e., attacks set to *true*) to determine whether the authentication property of DNP3-SAB is provided. But before we discuss the evaluation of DNP3-SAB, this section will provide the details of the attack models used for verifying the model. As previously emphasised in Section V-A, the adversarial behaviours considered for this analysis include replay, injection and modification attacks. The overall objective of these attacks is to broadcast a request and get it processed on the 3 outstations.

Figure 9 presents the attack model considered for this analysis. The attacks are modelled with places, transitions and parameters as before (see dotted area of Figure 9). These attack behaviours were set to false during the validation of the model. But to verify the model, we have enabled these attacks such that we can revalidate the security of the model. In Figure 9, the replay attack is modelled through the *Capture Reqs* and *TrigRep* places and the *Replay* transition. At this point the attack is able to store and replay previous messages from the master station. The injection attack is modelled through the `InjectR` place and the *ExcM1* transition. This allows the attacker to forge and inject an existing request data with random strings via the network to the outstations (see the box on top of *InjectR*, Figure 9). The modification attack is modelled through the function `MoDAttack1`. The function

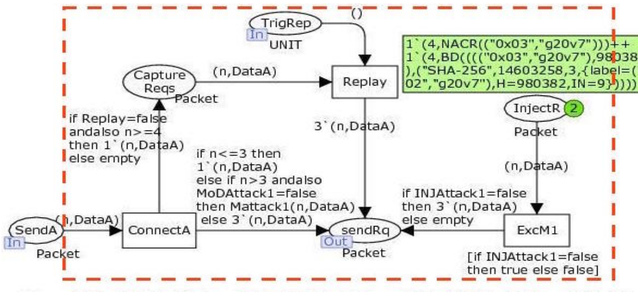

TrigRep
In
UNIT
()
Capture Reqs
Packet
(n,DataA)
Replay
1`(4,NACR(("0x03","g20v7")))++
1`(4,BD(((("0x03","g20v7"),980382,9),("SHA-256",14603258,3,{label=("0x02","g20v7"),H=980382,IN=9})))))
InjectR
2
Packet
if Replay=false andalso n>=4 then 1`(n,DataA) else empty
3`(n,DataA)
(n,DataA)
if n<=3 then 1`(n,DataA) else if n>3 andalso MoDAttack1=false then Mattack1(n,DataA) else 3`(n,DataA)
if INJAttack1=false then 3`(n,DataA) else empty
SendA
In
Packet
(n,DataA)
ConnectA
sendRq
Out
Packet
ExcM1
[if INJAttack1=false then true else false]

Fig. 9: The attack models in the network module.

is defined as a parameter on the arc inscription binding the *ConnectA* transition and the `sendRq` place. This implies that the attack can actively modify data (request contents and network addresses) while in transmission through the network.

In Table I, the attack models on SAB show 3 dead markings for the injection and replay attack models, and a single dead marking for the modification attack. Obtaining these markings with their respective dead transitions led us to simulate the model to investigate the behaviour obtained for each attack model. The investigation of the modification attack revealed that the single dead marking obtained represents the execution of the *Gen AuthError* and *Errfback* transitions in all outstations; indicating an authentication failure for the initial model state. This behaviour is expected because the attacker is actively modifying the content of the packet since the message in the protocol is in 'cleartext'. As a result, the initial request from the master station does not get executed because the integrity of the initial message has been compromised. This led to 53 dead transitions because the initial request was not executed on any of the outstations. Investigating the injection attack model revealed that one of the obtained dead markings represented the successful execution of the *SenSor* transition on all the outstations (see Figure 6); leading to obtaining the token value '*3*' on each of the outstations. This implies that authentication for the initial broadcast message from the master station was successful. This is expected. However, the remaining dead markings (2) of the injection attack represent the case of failure to terminate the protocol. This failure to terminate the protocol led to the execution of the *Errfback* and *Gen AuthError* transitions. A further investigation showed that this failure occurred because the parameters provided by the attacker could not be verified by SAB as required. As a result, the execution of *Errfback* and *Gen AuthError* transitions indicate an authentication failure. Moreover, the analysis also showed that the 25 dead transitions obtained from the model were as a result of the failure; as certain transitions could not be executed due to the lack of data. The model reveals that it is computationally hard for the attacker to forge and provide the expected data. This is because the attacker has no prior knowledge of the hash-chain parameters pre-established between the stations. For this reason, this failure will give an insight to an administrator through its log files.

Similarly, with the replay attack 1 out of the 3 dead markings indicated the successful execution of the *SenSor* transition on all the outstations. This also implies that the initial state of the model (request from the master station) was successfully authenticated before processing. However, both remaining dead transitions represented authentication failures, which lead to the execution of *Errfback* and *Gen AuthError* transitions. In the replay attack, *Errfback* and *Gen AuthError* transitions occurred because the replayed messages lacked freshness; new hash tags and MAC tags. As a result, 22 dead transitions occurred. In summary, these behaviours indicate that all parameterised attacker behaviours in this model have been detected and stopped. As a result, no unauthorised commands were possible to execute on the outstations without a proper authentication performed.

VIII. Performance Analysis and Comparison

This section provides a comparison of storage, communication and computation overheads of SAB against its unicast counterparts: DNP3-SA NACR and AGM. The comparisons are based on the total headcount of messages within the protocol but not in terms of bytes sizes. This is because most of the messages have similar byte sizes. Moreover, the performance of these authenticated unicast and broadcast communication schemes are compared under adversarial conditions. The purpose of this presentation is to show that SAB is usable because our lightweight security solution improve performance and reduce overheads.

The following notations are used in Table II. Here, N represents the number of outstations a master station is communicating with. Here n denotes the approximate number of commands to be exchanged between the master station and its N outstations per user and during a time interval between two key updates. Additionally, n also represents the number of hash tags in a hash chain generated as part of SAB. SAB uses one hash tag per request as illustrated in Figure 3. H denotes the hash function used to compute the hash chains. E and D denote encryption and decryption respectively. DA_S, DA_N and DA_A denote the Drop attack on SAB, NACR and AGM, respectively. In the drop attack, an adversary drops a request from the master station with the probability P_a. MIR_S, MIR_N and MIR_A denote the modification, injection and replay attacks on SAB, NACR and AGM respectively. The probability that these attacks will target a request from the master station is represented as P_a. For the sake of comparability, we have represented all cryptographic transformations used in DNP3-SA and DNP3-SAB as C (i.e., C=$\{H, E, D, MAC\}$). Similarly, the stored parameters used in DNP3-SA and DNP3-SAB are represented as K (i.e., K=$\{S_n, n, UpK, Sk_{mo}, P, KSn\}$).

To understand the overheads analyses presented in Table II, three things must be noted. 1) There are N outstations communicating with a single master station. 2) There exists a key update process that occurs before the NACR or AGM operates. The key update process has 4 headcount of messages per round and per user (refer to Figure 3). 3) NACR is a prior requirement to AGM operation. This implies that the number of communicated messages in NACR will also be considered in AGM operation. Table II SAB shows that the total messages involved in SAB communication is $4N+ n$. This value ($4N+ n$) is obtained because there are 4 messages exchanged during the key update process to N outstations and a single (n) broadcast requests are sent (refer to the exchanged messages in Figure 3). This value corresponds to O(N+n). In processing SAB, the master needs to generate a hash chain and a MAC tag. This is represented in the table as $2n$+2 $\approx$ O(n) (including the key update set-up). Similarly, processing SAB on each outstation requires $4Nn$ (including the keys update). This is because that each of the N outstations computes a hash and MAC tag. The value on the outstations ($2Nn$) corresponds to O(Nn) while that of the master corresponds to O(n). For SAB storage overhead, the table shows $2n$+2 for the master and $8N$ for the outstations. The value $2n$+2 on the master represents a hash value (Si) with its index (i), and the storage of 2 keys (i.e., U_{pK} and SK_{mo}). Similarly, $8N$ on the outstations imply that each of the N outstations stores 8 parameters consisting of 2 keys (U_{pK} and SK_{mo}), $chalg$, S_n, i, S_{calc}, S_{stored} and N_{stored}. Here, $8N$ corresponds to O(N) and ($2n$+2) corresponds to O(n).

In Table II, the overheads of NACR and AGM are also presented. The overheads of NACR is almost as the same as that of the AGM. Except that overhead of the AGM include the NACR behaviour as prior requirement. However, both modes take into account N outstations. This is done to compare SAB to NACR and AGM. From Table II, comparing the overheads of SAB to the existing NACR and AGM overheads, the table shows that SAB has less communication overhead over its counterparts: NACR and AGM. This is because in SAB operation, the master sends a single request to N outstations at once, rather than sending a request to each of the N outstations (as this is the case for NACR and AGM). However, the table shows a slight (minor) increase in storage and processing overheads on the N outstations during SAB. This slight increase is asymptotically insignificant because SAB is designed to run on large systems rather than a small scale systems. In case NACR and AGM are to be used on large scale systems, the processing and storage overheads will dramatically increase. This is because both modes will have to individually exchange n of messages to N outstations. The advantage of SAB communication overhead becomes more clearer when attacks such as drop, modification, injection and replay attacks are considered (refer to Table II). Thus, in the attack models (DA_S and MIR_S), SAB requires much less retransmission (nP_a) for each request attacked with probability P_a than its counterparts. In addition, SAB causes less processing overhead on the master station compared to NACR and AGM. The extra processing caused on the master station due to an attacked request is much less for SAB than for NACR and AGM.

Based on this analysis, we conclude and make it clear that there is always a trade-off of performance when providing security. Table II has shown that, there exists the benefits in the entire communication overhead of SAB as well as processing overhead on the master stations. But these benefits come at the cost of an extra storage used on the master stations to store hash chains and their indexes. Additionally, calculations on the outstations also cause a slight increase in processing

TABLE II: Performance Analysis and Comparison

	Communication (# messages)	Processing (# Crypto Operations C)		Storage (# Parameters K)	
		Master	N Outstations	Master	N Outstations
SAB	$4N$+n $\approx$ O(N+n)	$2n$ + 2 $\approx$ O(n)	$4Nn$ $\approx$ O(Nn)	$2n$+2 $\approx$ O(n)	$8N$ $\approx$ O(N)
NACR	$4N$(n+1) $\approx$ O(Nn)	Nn $\approx$ O(Nn)	Nn $\approx$ O(Nn)	4 (Constant)	$4N$ $\approx$ O(N)
AGM	$8N$+$2Nn$ $\approx$ O(Nn)	N(n+1) $\approx$ O(Nn)	Nn $\approx$ O(Nn)	4 (Constant)	$4N$ $\approx$ O(N)
DA_S	nP_a $\approx$ O(n)	nP_a $\approx$ O(n)	NnP_a $\approx$ O(Nn)	-	-
DA_N	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	-	-
DA_A	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	-	-
MIR_S	nP_a $\approx$ O(n)	nP_a $\approx$ O(n)	nP_a $\approx$ O(n)	-	-
MIR_N	$2NnP_a$ $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	-	-
MIR_A	$2NnP_a$ $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	NnP_a $\approx$ O(Nn)	-	-

TABLE III: Performance and Functionality of SAB using Alternative Methods

Functionality				Performance		
Schemes	Mechanism	Infrastructure	Security Properties	Communication	Computation	Storage
Proposed SAB	Hash-chain	None	-Authentication -Integrity -Resistant to dishonest outstations*	1 Key update per n message	n*HMAC n*MAC 1*AES	1 Symmetric Key 1 Hash value per outstation
SAB_{Sym}	Message-based key establishment	None	-Authentication -Integrity -Require honest outstations	n Key update per n message	n*HMAC n*MAC n*AES	1 Symmetric Key 1 nonce value per outstation
SAB_{Asym}	Digital signature	PKI	-Authentication -Integrity -Non-repuation	Public key distribution	n*PKI	1 Public Key 1 nonce value per outstation

overheads by a constant term. This slight difference disappears in the attack scenarios. SAB, NACR and AGM causes the same processing overhead on outstations when a request message is attacked (i.e., drop, modification, injection and replay) with a probability P_a. Furthermore, as SAB utilises only the existing security primitives from NACR and AGM (refer to step 2–6 of Figure 3), it implies that there are no additional cryptographic updates required for devices supporting DNP3-SAB. Existing devices that support DNP3-SA will have sufficient resources to support DNP3-SAB as well. Also, the time it takes a master station to send a request to N outstations securely using DNP3-SA is more than the time required for DNP3-SAB. This is because, using DNP3-SA, the master station has to contact N outstations individually in sequence, causing some outstations to receive the request later than others.

In Table III, the performance and functionality of SAB using two alternative methods: symmetric (SAB_{Sym}) and asymmetric key (SAB_{Asym}) cryptography are presented. The table shows that while the proposed SAB in this paper uses hash-chain values to authenticate each broadcast message, the SAB_{Sym} with symmetric key requires a broadcast key established for each broadcast message. SAB_{Asym} can use a single public key to authenticate all broadcast messages, but it requires an infrastructure to distribute public keys. All three schemes provide authentication and integrity. But the SAB_{Sym} requires all outstations to be honest. This is because a compromised outstation with the knowledge of the symmetric key can masquerade the master station. SAB_{Asym} can additionally be used for non-repudiation. In terms of communication, SAB_{Asym} distributes one public key for authenticating all broadcast messages, SAB requires a symmetric key and a hash-chain value distribution once for n broadcast messages. SAB_{Sym} distributes a symmetric key for each broadcast message. The advantage of SAB_{Asym} comes with high computation requirements, which are not available for most master and outstations in service today. SAB_{Sym} requires more computation than SAB due to key establishment requirements. All schemes require one key and random value per message to be stored. Excluding SAB_{Asym} due to its high computation and infrastructure requirements, SAB shows clear advantage over SAB_{Sym} in security and computation and communication overhead.

IX. Discussion and Conclusion

The HMAC mechanism of the DNP3-SA protocol alone is not sufficient to secure the DNP3 broadcast mode. This is because the HMAC mechanism will require a different key to compute a unique MAC tag for each broadcast message. The proposed SAB uses a hash-chain to solve the problem of key management. Hash values used for messages are linked together as a hash-chain. The usage of a single key together with the hash-chain removes need for a key per message. Additionally, because the hash-chains are also associated with indexes, it prevents potential replay attacks from occurring. Using only the index without hash values would not be sufficient because any compromised outstation that already has knowledge of the symmetric key and the index can masquerade as the master station and send broadcast messages to its counterparts. The analysis of the SAB CPN model in this paper has shown that the DNP3 broadcasts can be secured against attack vectors such as modification, injection, spoofing and replay, attacks listed by the DNP3 consortium in the protocol specification. Moreover, SAB's performance and comparison with alternative methods have shown SAB's advantage in security, computation and communication overhead.

In the future, we will extend our model to address communication on a data concentrator. A data concentrator is an architecture where one a master station interacts with multiple sub-master stations and outstations with different communication modes. The motive of this work will be to analyse various security properties on such a complex architecture.

References

[1] J. Akerberg and M. Bjorkman, "Exploring security in PROFINET IO," in *Computer Software and Applications Conference, 33rd Annual IEEE International*, vol. 1. IEEE, 2009, pp. 406–412.

[2] I. Al-Azzoni, D. G. Down, and R. Khedri, "Modeling and Verification of Cryptographic Protocols using Coloured Petri Nets and Design/CPN," *Nordic Journal of Computing*, vol. 12, no. 3, p. 201, 2005.

[3] R. Amoah, S. Camtepe, and E. Foo, "Formal modelling and analysis of DNP3 secure authentication," *Journal of Network and Computer Applications*, vol. 59, pp. 345 – 360, 2016.

[4] R. Amoah, S. Suriadi, S. Camtepe, and E. Foo, "Security analysis of the non-aggressive challenge response of the DNP3 protocol using a CPN model," in *Communications (ICC), IEEE International Conference on*, June 2014, pp. 827–833.

[5] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in SCADA systems," in *Proceedings of the International Infrastructure Survivability Workshop*, 2004.

[6] M. Cheminod, A. Pironti, and R. Sisto, "Formal vulnerability analysis of a security system for remote fieldbus access," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 1, pp. 30–40, 2011.

[7] D. Choi, H. Kim, D. Won, and S. Kim, "Advanced key-management architecture for secure SCADA communications," *Power Delivery, IEEE Transactions on*, vol. 24, no. 3, pp. 1154–1163, 2009.

[8] D. Choi, S. Lee, D. Won, and S. Kim, "Efficient secure group communications for SCADA," *Power Delivery, IEEE Transactions on*, vol. 25, no. 2, pp. 714–722, 2010.

[9] G. Clarke and D. Reynders, *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, an imprint of Elsevier, Linacre House, Jordan Hill, Burlington, MA., 2004, no. ISBN 07506 7995.

[10] S. East, J. Butts, M. Papa, and S. Shenoi, "A Taxonomy of Attacks on the DNP3 Protocol," *Critical Infrastructure Protection III*, pp. 67–81, 2009.

[11] G. Gilchrist, "Secure authentication for DNP3," in *IEEE Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008*, Pittsburg, PA, 2008, pp. 1–3.

[12] S. Gilmore, *Programming in Standard ML'97: A tutorial introduction*. University of Edinburgh, 1997.

[13] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: communication technologies and standards," *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 529–539, 2011.

[14] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 2008.

[15] ICS-CERT, "Monthly Monitor (ICS-MM201502), ICS-CERT Monitor," https://ics-cert.us-cert.gov/monitors/ICS-MM201502, 2015, Accessed: 10/03/2015.

[16] IEEE, "IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3)," *IEEE Std 1815-2012*, pp. 1–866, 2012.

[17] *IEC 62351Power systems management and associated Information exchange- Data and Communications security Part 1: Communication network and System security – Introduction to security issues*, International Electrotechnical Commission Std. IEC TS 62 351-1, Rev. First Edition, 2007.

[18] K. Jensen, L. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems," *Intl. Journal on Software Tools for Technology Transfer (STTT)*, vol. 9, no. 3, pp. 213–254, 2007.

[19] K. Jensen, S. Christensen, and L. M. Kristensen, *CPN tools state space manual*, University of Aarhus - Dpt. of Computer Science, Aarhus N, DK, 2006.

[20] K. Jensen and L. Kristensen, *Coloured Petri Nets*. Springer-Verlag Berlin Heidelberg, 2006.

[21] W. Kim, "On cyberwarfare," *International Journal of Web and Grid Services*, vol. 8, no. 4, pp. 321–334, 2012.

[22] X. Lu, W. Wang, and J. Ma, "An empirical study of communication infrastructures towards the smart grid: Design, implementation, and evaluation," *Smart Grid, IEEE Transactions on*, vol. 4, no. 1, pp. 170–183, 2013.

[23] B. Miller and D. Rowe, "A survey SCADA of and critical infrastructure incidents," in *Proceedings of the 1st Annual conference on Research in information technology*. ACM, 2012, pp. 51–56.

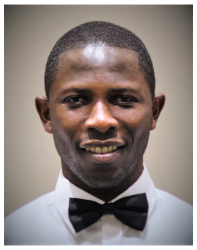
[24] I. Modbus, "Modbus Application Protocol Specification v1. 1a," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2004.

[25] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, 2005.

[26] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wireless networks*, vol. 8, no. 5, pp. 521–534, 2002.

[27] *PROFInet Architecture Description and Specification*, PROFIBUS International Std., August 2003, version 2.01. [Online]. Available: http://www.PROFIBUS.com

[28] S. Suriadi, C. Ouyang, and E. Foo, "Privacy compliance verification in cryptographic protocols," *Transactions on Petri Nets and Other Models of Concurrency VI: Lecture Notes in Computer Science*, vol. 7400, pp. 251–276, 2012.



Raphael Amoah (M'14) received his Ph.D. degree from the Queensland University of Technology (QUT), Brisbane, Australia in 2016. He has been a Sessional Academic with the Queensland University of Technology, Australia, since 2012. His current research interests include cyber-physical systems, formal methods, formal verification of communication protocols used in supervisory control and data acquisition (SCADA) and industrial controls systems security such as smart grid.

Seyit Camtepe (S'04 - M'07) received the Ph.D. degree in computer science from Rensselaer Polytechnic Institute, New York, USA, in 2007. From 2007 to 2013, he was with the Technische Universitaet Berlin, Germany, as a Senior Researcher and Research Group Leader in Security. Since 2013, he has been with Queensland University of Technology, Australia, as a Lecturer. His research interests include mobile and wireless communication, pervasive security, and applied and malicious cryptography.

Ernest Foo (M'99) received his Ph.D. degree from the Queensland University of Technology (QUT), Brisbane, Australia in 2000. From 2007, he has been a senior lecturer and researcher at the Information Security Discipline in the School of Electrical Engineering and Computer Science at QUT. Dr Foo's research interests can be broadly grouped into the field of secure network protocols with an active interest in the security of industrial controls systems such as SCADA and the smart grid.