Long-Lived Authentication Protocols for Process Control Systems

Rasika Chakravarthy[a], Carl Hauser[a], David E. Bakken[a]

[a]*School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington 99164-2752, USA*

Abstract

Process control systems that manage critical infrastructures have to be available continuously; they may have nodes that once deployed cannot be easily accessed; and they need to be functional over long periods of time. Since the consequences of critical infrastructure disruptions are potentially serious and since critical infrastructures are under threats ranging from extortion to terrorism, it is vital to keep the security services up to current standards over many years of deployment.

The mutual authentication of process control system nodes is a fundamental building block of security. This paper describes authentication protocols for use in long-lived process control systems. The protocols address the issue of longevity by defining, as part of the protocol suites, a means for the cryptographic components of the authentication system to be replaced dynamically and securely. The correctness of the component update protocol is established using an extended version of the Burrows, Abadi and Needham (BAN) authentication logic, which incorporates primitives and rules for reasoning about the belief of the ability of cryptographic modules to maintain secrecy.

Email addresses: rasika.mudumbaichakravarthy@email.wsu.edu (Rasika Chakravarthy), hauser@eecs.wsu.edu (Carl Hauser), bakken@eecs.wsu.edu (David E. Bakken)

Keywords
Process control systems, authentication protocols, long-lived protocols, mutual authentication, BAN logic

Submitted: January 15, 2010
Accepted: September 25, 2010

1. Introduction

Critical infrastructures such as electric power grids, water infrastructures, and oil and gas pipelines increasingly make use of technological advances in networking to enable timely and reliable information exchange. The process control systems that manage these critical infrastructures face specific demands. They have to be available continuously without being shut down completely, they may have nodes that once deployed cannot be easily accessed, and they need to be functional over long lifetimes. Since the losses due to critical infrastructure disruptions are potentially serious and since critical infrastructures are under constant threats ranging from extortion to terrorism, it is necessary to keep the security services up to current standards over many years of deployment [1, 2, 4].

Securing the communications components that support process control systems is made more difficult by the longevity of the systems. The security field is constantly evolving: algorithms become easier to break with advances in technology and computing power, and newer and stronger algorithms are introduced to remedy the flaws in older algorithms. However, the communications subsystem of a process control system has to last a long time. Allowing the security of a process control system to become obsolete makes the system itself and the critical infrastructure asset that it manages vulnerable. Indeed, considerable efforts are being focused on securing process control systems for which the underlying security assumptions have changed [2]. A flexible architecture is required to prevent the security components in process control systems from becoming obsolete. Such an architecture, at the very least, must allow the replacement of the cryptographic subsystem throughout the lifetime of a process control system.

Cryptographic systems depend on the secrecy of the keys and the unbreakability of the underlying algorithms. Keys that are used for a long time become vulnerable; continuing to use keys until they are compromised is dangerous [19]. Consequently, it is important to be able to change keys safely when necessary using a "re-keying" protocol. Similarly, continuing to use a compromised module is also dangerous. Just as provisions are made to change keys, provisions must be made to change modules using a "re-moduling" protocol.

In a previous paper [20], we described a modular, reconfigurable security subsystem for a flexible, managed communications system that supports electric power grid monitoring and control [1]. Security functions such as confidentiality and message integrity are provided by stackable, replaceable modules that can be (re)configured during system operation, and by preloaded key material that is consumed over the lifetime of the system. The principal shortcoming of this work is that it does not specifically address authentication between the communicating parties (although aspects of authentication are implicit in how the shared keys were used by the parties). This paper adopts the replaceable module and pre-shared key strategies used in our earlier work, but applies them explicitly to end-point authentication. Once end-point authentication is achieved, it is straightforward to recover the modular confidentiality and message integrity functions of our previous work without direct reliance on pre-shared keys.

The principal contributions of this paper are pairwise authentication protocols that support secure re-keying and re-moduling. The protocols are robust against adversaries who can read, modify, insert and replay network messages. We assume that over a period of time an adversary may gain the ability to successfully attack a cryptographic algorithm, which then gives the adversary the ability (in a shorter but non-zero amount of time) to acquire the key being via the cryptanalysis of message traffic. We quantify the conditions under which successful re-moduling can be accomplished even when an attacker has the ability to crpytanalytically determine the keys used with the current module. This paper does not address the problem of protecting keys from direct attacks on nodes. Such protection is clearly essential, but is orthogonal to the problems addressed by the protocols presented in this paper.

2. Security Architecture Characteristics

The security environment for a wide-area process control system used in the electric power grid is very complex. In the power grid, multiple organizational entities (e.g., utilities, regulators and independent generators) control different portions of the grid, but they have to cooperate to provide highly reliable electric power. Each entity must protect its business-sensitive information from other entities, but it is sometimes the case that sensitive information has to be shared to achieve systemwide operational goals.

The GridStat middleware framework [1] was designed to meet the need for flexible, controlled sharing of data streams produced by sensor devices deployed throughout the power grid. The GridStat security architecture is based on two principles. First, data produced by any sensor can be delivered to consuming applications located anywhere else in the grid. For example, an oscillation detection algorithm for the Western North American grid might use synchrophasor data streams from utilities in British Columbia, Arizona, California, Washington and Oregon. Second, owners of data streams are responsible for the integrity of the data and must be able to control access to the data.

The GridStat design uses a "data plane" to provide the flexibility required by the first principle and a "management plane" to provide the control required by the second principle. In Figure 1, publisher, subscriber and forwarding engine nodes correspond to data plane components while security management service (SMS) nodes correspond to management plane components. Other management plane components, which are not shown, manage routing, resource utilization and other aspects of the data plane.

The authentication problem addressed in this paper concerns the mutual authentication of adjacent nodes in the SMS hierarchy and the mutual authentication of data plane devices with the leaf SMSs that manage them (Figure 1). Just as the pairwise authentication between Kerberos [13, 17] clients and servers provides the foundation for client-server authentication, the authentication of producers and consumers to their corresponding leaf SMSs is the foundation in GridStat for access control, message authentication, data confidentiality and other security services that are functions of the distributed SMS network. Authenticating data plane devices contributes to the novelty of the problem. These devices are located in substations or remote locations (and even on power poles), which makes them difficult and expensive to service, yet they have lifetimes that extend for decades. An au-

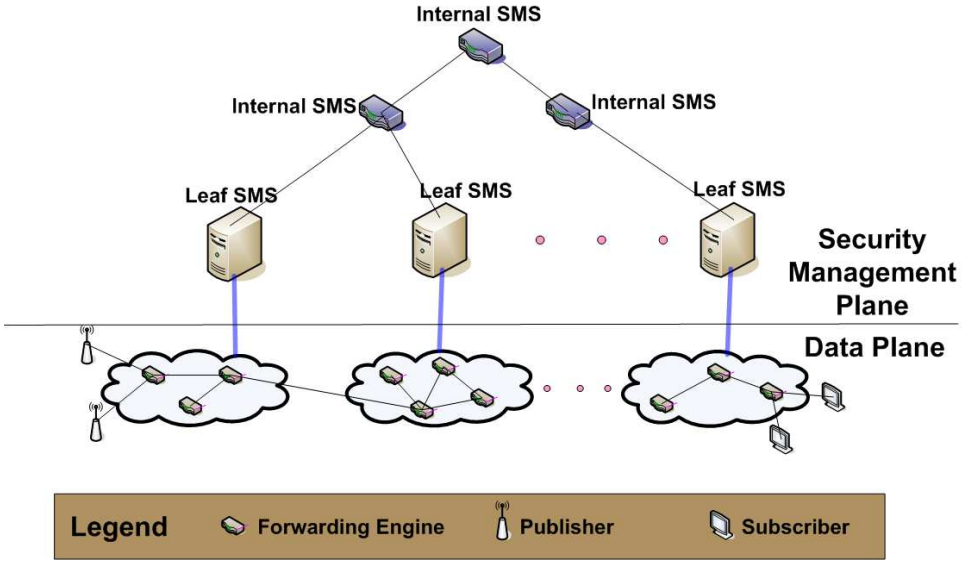

Figure 1: GridStat security architecture.

thentication design allowing cryptographic components to be replaced over the long lifetimes of these devices is attractive for security as well as financial reasons.

3. Key Distribution and Management

Unlike in the general Internet environment, the set of devices constituting a process control system changes slowly, with careful planning, well-defined roles for new devices and well-defined responsibilities for their management over their lifetimes. In the GridStat system, the management role for data plane devices is asssumed by the leaf SMSs and for SMSs by their parent SMSs. We adopt parent-child relationships in both cases. Note that these relationships are statically defined when the nodes are commissioned.

The first basis for the longevity of the authentication component is that each node and its parent are provided with a collection of key material at the time of their commissioning. This key material is kept confidential by the node and the parent (i.e., it is not shared with the node's siblings). In determining the amount of key material provided, the goal is to ensure that the supply is sufficient to always allow several protocol steps to be executed using a "fresh key" (i.e., one that has not been used before) drawn

from the initial supply. The second basis is the ability of the protocols to dynamically replace the encryption algorithms used for authentication. Thus, if at some time during the life of the system it is discovered that the Advanced Encryption Standard (AES) is breakable, then a module that implements a stronger encryption algorithm can be distributed.

Maintaining keys, synchronizing keys between nodes and changing keys regularly and safely are all known limitations of symmetric key algorithms. Nevertheless, our desire to make encryption modules dynamically replaceable forces the choice of symmetric key algorithms as the cryptographic basis. Symmetric key algorithms can be keyed with arbitrary bit strings of the appropriate length. Public key algorithms do not share this property – by their very nature the public and private keys have to maintain a particular mathematical relationship [19]. Since we cannot foresee the algorithms and key constraints of the future, the strategy of using preloaded key material does not work for public key algorithms. For a new symmetric key algorithm, however, an appropriate number of bits can simply be taken from the preloaded key material for use as a new key as shown in Figure 2.

The use of symmetric keys does, however, raise the possibility of reflection attacks. A reflection attack occurs when an intruder tricks an authentic node into signing its own challenge, leveraging the fact that the signatures of the parent and child look alike. A straightforward solution is to use different keys [11], which is easily accomplished by drawing separate keys from the available key material for signing by the parent and child.

Finally, because the roles of the manager and the managed are well defined in the system architecture, the protocols can be based on a command paradigm instead of a negotiation paradigm. The decision to initiate a key or module change always resides with the parent node. It may be triggered as a result of a policy decision made higher up in the hierarchy in response to information received from an intrusion detection system or even in response to messages received from the child.

4. Authentication Protocol

When two nodes need to communicate with each other, they start with an authentication protocol. The re-keying and re-moduling protocols are independent of the authentication protocol. Therefore, any authentication protocol that can be adapted to work with a set of preloaded keys can be used. Following the successful execution of the authentication protocol, the

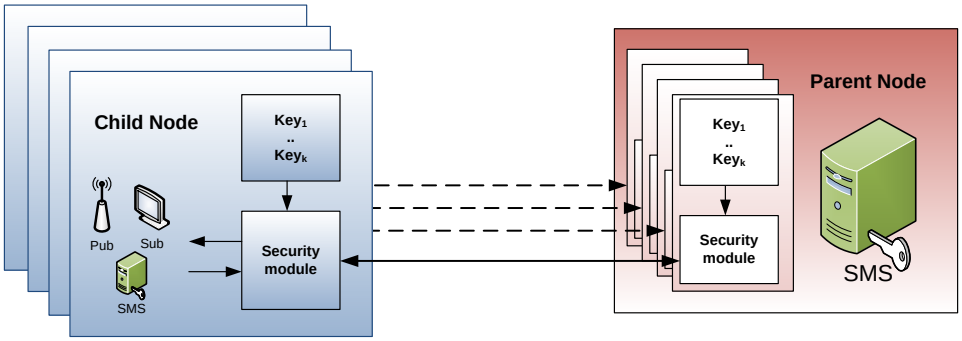

Figure 2: Preloaded shared keys.

Protocol 1 Authentication

1: **Child:** {*Hello*, *Diffie-Hellman Parameters*} [Parent calculates DHK]
2: **Parent:** {*Hello*, *Diffie-Hellman Parameters*,$\{R1\}_{DHK}$} [Child calculates DHK]
3: **Child:** $\{\{\{R1+1\}_{DHK}\}_{CK}, \{R2\}_{DHK}\}$
4: **Parent:** $\{\{\{R2+1\}_{DHK}\}_{PK}, \{Session\ Keys\}_{DHK}\ \}$

two nodes share a session key (or keys) known only to the two nodes. Our Authentication Protocol (Protocol 1) incorporates a simplification of the mutual authentication steps in the Internet Key Exchange Protocol [11, 10] used with IPSEC [12]. Note that the corresponding protocol with the parent initiating is obtained by switching the roles of parent and child in the protocol.

As in the Internet Key Exchange Protocol, the Diffie-Hellman algorithm is used to exchange proofs of identity and also to exchange the encryption keys. The main difference in our scenario is that the nodes have a control hierarchy, which means that the parent nodes dictate the algorithms used by the child nodes. Consequently, there is no negotiation of the cryptographic suites to be used.

In the protocol listing, DHK indicates a Diffie-Hellman key, CK is the current child authentication key and PK is the current parent authentication key (as determined by the Re-Keying Protocol described below). $R1$ and $R2$ are random values used as challenges. Encryption is performed using the current algorithm – each pair of nodes is provided with at least one initial algorithm. Over the lifetime of the system, the Re-Moduling Protocol (described below) is used from time to time to ensure that the nodes always

Protocol 2 Re-Keying

1: **Parent:** $\{Switch\ Key\}_{SK}$
2: **Child:** $\{R1\}_{SK}$
3: **Parent:** $\{\{R1\}_{NPK}, R2\}_{SK}$
4: **Child:** $\{\{R2\}_{NCK}\}_{SK}$
5: **Parent:** $\{\{R2+1\}_{NPK}, Acknowledgement\ \}_{SK}$

share an algorithm that is secure in the current environment.

This protocol is only one example of many possibilities for authentication using shared-key cryptography. Interested readers are referred to Boyd and Mathuria [3] for descriptions of other protocols.

5. Re-Keying Protocol

The Re-Keying Protocol (Protocol 2) – perhaps more appropriately called the "Key Advance Protocol" because it moves to the next unused key in the preloaded key material – moves both nodes to use "fresh" (i.e., unused) authentication keys. As noted above, the key and module changes are always initiated by the parent node. To authenticate the switch key request, the child returns a value to be signed by the parent (Step 2). The parent uses the next parent key (i.e., a fresh key, not the current one) to sign the child's challenge. An intruder would not know the new key via the communication channel because it has not been used before. Similarly, the child uses its new authentication key to sign the parent's reply challenge.

In the protocol listing, SK is the session key used for encryption, NPK the next parent authentication key and NCK the next child authentication key drawn from the preloaded key set. Note that in Protocols 2 through 4 the communications are shown as occurring under a session key SK. This is not essential for correctness, but it a good practice. Even if SK has been broken, the protocols can only succeed between nodes that share fresh keys.

6. Re-Moduling Protocol

The Re-Moduling Protocol (Protocol 3) addresses the fact that algorithms may become obsolete or vulnerable due to technological advances. Module changes can be initiated only by the parent. In general, the new module is not available at a child node so the Re-Moduling Protocol incorporates secure

Protocol 3 Re-Moduling

1: **Parent:** $\{Change\ Module\}_{SK}$
2: **Child:** $\{R1, \text{Request}\}_{SK}$
3: **Parent:** $\{\{R1, NM\}_{<NPK,OM>}, \{R1\}_{<NPK,NM>}, R2, NM\}_{SK}$
4: **Child:** $\{\{R2\}_{<NCK,NM>}\}_{SK}$
5: **Parent:** $\{Acknowledgement\}_{SK}$
6: **Both:** move to the key following NPK for parent authentication.

delivery of the new module from the parent to the child as shown in protocol listing. The protocol moves to a new, presumably secure, encryption module even if the current key and module have been broken by relying on fresh keys drawn from the preloaded key material.

In the protocol listing, OM is the old authentication module and NM is the new authentication module. The parent sends the change module command to the child. The child requests the new module from the parent node along with challenge $R1$. The parent then signs $R1$ and NM using the next authentication key. Note that it would be sufficient to sign NM because the modules are not required to be kept secret.

Step 3 shows NPK being used with the current and the new authentication modules. Since it is possible that the two modules use different key lengths, NPK is merely a representation of the next preloaded key. The number of bits in each preloaded key is the maximum of the number of bits needed for OM and the number of bits needed for NM. In Step 3, the parent uses the current and new authentication modules to sign $R1$. One might expect that the use of only the new module would establish the initiator's identity since the next authentication key is used with it. However, it is possible for the initiator to send a malicious module or a dummy module as the new module, which would pass the authentication tests. A signature on the challenge with the old module and the new key ensures that the initiator knows the new key and is, therefore, genuine.

Note that if OM has been broken, an attacker might be able to intercept the parent's message, cryptanalytically derive NPK, and replace the parent's message with another message containing a dummy new module. Therefore, it is important to assume that, even if a module has been broken, an attacker is unable to determine the value of a fresh key (in this case NPK) used with the module in less time than it takes for Steps 3 and 4 to be executed. Furthermore, because NPK has been used with OM, it is not safe to continue

Protocol 4 Optimized Re-Moduling

1: **Parent:** $\{Change\ Module\}_{SK}$
2: **Child:** $\{R1\}_{SK}$
3: **Parent:** $\{\{\{R1\}_{<NPK,NM>},\ R2\}_{SK}\}$
4: **Child:** $\{\{\{R2\}_{<NCK,NM>}\}_{SK}\}$
5: **Parent:** $\{Acknowledgement\}_{SK}$

to use it; consequently, the module change consumes two new parent keys. In the event that the parent and child detect that there has been an attack using this protocol, they cannot continue to use the current module and key. The only other option is to execute the Re-Moduling Protocol (Protocol 3) again, but this has the risk of exhausting the key material.

The protocol can be optimized if the child already has the new module loaded via an out-of-band technique or supplied during an earlier run of Protocol 3. In Step 3 of the Optimized Re-Moduling Protocol (Protocol 4), the child simply sends a challenge $R1$ to the parent and does not request the new module. The parent only has to sign the challenge $R1$ with the new module and the next authentication key. This is because the child knows the new module and, thus, an attacker cannot initiate the protocol and transmit a dummy module in the current run of the protocol. The parent also sends a random value $R2$ for the child to sign. Steps 4 and 5 of the protocol are the same as in Protocol 3. Note that the additional change of the parent key in the previous protocol is not required because the new key was not used with the old module.

In any these protocols, an attacker who breaks the session key can masquerade as the parent to a child. The protocol will subsequently fail, so this is not a security vulnerability. In fact, a failure to complete a protocol is potentially a warning of an attempted attack. For example, in Protocol 3, when the child checks the parent's signatures in Step 3, there are three cases of failure, which are shown in the first three rows of Table 1. Each case provides different information for alerting a security monitoring service.

The third row in Table 1, where the old module passes the authentication test, but where the new module fails, is quite interesting. It clearly shows that the encryption module and/or keys are not safe any longer and must be changed. Also, it shows that the old module is not safe, even with the new key. This may indicate a bug in the implementation on the initiator's side. Another possibility is that the initiator is malicious although it has not

Table 1: Implications of challenge-response in the Re-Moduling Protocol.

Response with Old Module	**Response with New Module**	**Implication**
Fail	Fail	No evidence of compromised modules or keys. Evidence of an attempted attack.
Fail	Pass	No evidence of compromised old modules or keys. Evidence of an attempted attack. New module is malicious.
Pass	Fail	Bug exists in new module or possible breach in old module and/or key.
Pass	Pass	Initiator is successfully authenticated. Switch to new module.

made use of the fact that it could use a trivial module as the new module to conduct a successful attack.

7. Evaluation of the Re-Moduling Protocol

It is necessary to ensure that the protocols work as expected. To achieve this, the protocols have to be analyzed and verified. The Burrows, Abadi and Needham logic (BAN logic) [5] has been widely used to verify authentication protocols [15] despite certain limitations [9, 15, 22]. BAN logic does not contain terms for describing the beliefs of protocol participants concerning the security of the encryption algorithms used in a protocol, so it is insufficient for verifying the Re-Moduling Protocol. However, we desire to present arguments for the correctness of the Re-Moduling Protocol and find the problem to be essentially one of belief. Therefore, this section summarizes the BAN logic formalisms and uses them in a less formal manner to establish the essential properties of the Re-Moduling Protocol. Note that at this level flaws due to cryptographic attacks, incorrect implementation or vulnerabilities are not considered.

7.1. BAN Logic Constructs and Postulates

The BAN logic formally expresses the beliefs held by the participating entities. In the logic, objects are differentiated according to their type, e.g., principals (entities or nodes), encryption keys and formulas (statements). The basic constructs used to express a protocol in BAN logic are:

1. P **believes** X: Principal P believes that formula X is true.
2. P **sees** X : Some principal sent X to P.
3. P **said** X : P said X at some point of time (past or present).
4. P **controls** X: P has jurisdiction over X, i.e., P is the authority in the truth of X and should be trusted in this aspect.
5. **fresh**(X) : X has been sent in the present epoch and not in the past.
6. $P \xleftrightarrow{K} Q$: P and Q have a shared key K that is not known to any other principal except P and Q.
7. $\{X\}_K$: Assertion X encrypted with key K.

The postulates of BAN logic permit reasoning about the participants' beliefs based on the messages they have received. Conjunction is denoted by a comma and implication by a horizontal bar. Note that for convenience, we use different names for some of the original BAN logic rules (Freshness Rule (Equation (4)) and Belief Rule (Equation (5))):

Message-Meaning Rule:

$$\frac{P \textit{ believes } (P \xleftrightarrow{K} Q),\ P \textit{ sees } \{X\}_K}{P \textit{ believes } Q \textit{ said } X} \tag{1}$$

Nonce-Verification Rule:

$$\frac{P \textit{ believes } fresh(X),\ P \textit{ believes } Q \textit{ said } X}{P \textit{ believes } Q \textit{ believes } X} \tag{2}$$

Jurisdiction Rule:

$$\frac{P \textit{ believes } Q \textit{ controls } X,\ P \textit{ believes } Q \textit{ believes } X}{P \textit{ believes } X} \tag{3}$$

Freshness Rule:

$$\frac{P \textit{ believes } fresh(X)}{P \textit{ believes } fresh(X, Y)} \tag{4}$$

Belief Rule:

$$\frac{P \textit{ believes } X,\ P \textit{ believes } Y}{P \textit{ believes } (X, Y)} \tag{5}$$

The goal of an authentication protocol is to establish that at the completion of the protocol a session key K satisfies the following four beliefs:

$$A \textit{ believes } A \overset{K}{\longleftrightarrow} B \tag{6}$$

$$B \textit{ believes } A \overset{K}{\longleftrightarrow} B \tag{7}$$

$$A \textit{ believes } B \textit{ believes } A \overset{K}{\longleftrightarrow} B \tag{8}$$

$$B \textit{ believes } A \textit{ believes } A \overset{K}{\longleftrightarrow} B \tag{9}$$

The first two goals indicate that principals A and B on their own are convinced that K is a good shared key between A and B (it will not be discovered by any principal other than A and B from the messages sent in the protocol). In other words, each principal trusts the key. This does not, however, imply anything about the other party's beliefs. The third and fourth goals ensure that each party believes that the other party also trusts the key so that it can be used for communication.

The mechanics of a BAN logic proof involve restating a concrete protocol (e.g., Protocol 1 above) using the BAN logic formalisms to obtain an "idealized" formulation of the protocol. In the idealized form of a protocol, plaintext messages are deleted (because they do not contribute to the security of the result), and semantic implications of the contents of messages are encoded in BAN logic formulas. Note that the idealization steps are seen by some as a weakness of BAN logic because they are carried out informally.

The Authentication Protocol (Protocol 1) is an example of the kind of protocol that is amenable to proofs using BAN logic. Since the point of the Authentication Protocol is not to be novel and the point of the Re-Moduling Protocol is to permit the replacement of the authentication module if it is found to be insecure, in the interest of space, we do not discuss the Authentication Protocol any further.

7.2. Analysis of the Re-Moduling Protocol

The purpose of the Re-Moduling Protocol is to allow the parent node to re-establish secure, authenticated communication with the child when the parent has come to suspect the security of the module currently in use. To address this situation, the parent needs to put in place an authentication module that it believes to be secure and to do so in a secure way.

Since BAN logic does not explicitly express beliefs about modules, it cannot be used directly to make arguments about the correctness of the Re-Moduling Protocol. Therefore, we use the general structure of BAN belief arguments and introduce a new logical term and modified rules concerning the security of modules. The formalization of the extended logic is beyond the scope of this work.

To capture the semantic notion of the ability of a cryptographic module to protect secrets, we use the new terms $safe(Module)$ and $oneTimeSafe(Module)$, and adjust the inference rules so that the participants believe signed claims based on their belief in the secrecy of the key in use as well as in the safety of the module in use. The BAN Message-Meaning Rule (Equation (1)) is re-stated to include beliefs about modules as follows:

$$\frac{\begin{array}{c} P \textit{ believes } \{P \overset{K}{\longleftrightarrow} Q\}, \\ P \textit{ believes } safe(M), P \textit{ sees } \{X\}_{<M,K>} \end{array}}{P \textit{ believes } Q \textit{ said } X} \tag{10}$$

The above rule, however, cannot be used to bootstrap belief in the safety of a module. Recall that one goal of the Re-Moduling Protocol is to allow the re-establishment of secure authentication after the participants have learned that the current module in use may be insecure. Referring to the earlier discussion of Protocol 3, note that the assumption that the old module (even if is now considered unsafe in general) is still safe if used with a previously-unused key.

This assumption appears to require a non-monotonic logic for its formulation, which takes us beyond the monotonic BAN logic. Therefore, we proceed with a simpler rule, pushing to the idealization step the establishment of the required $oneTimeSafe(M)$ property, which asserts that a module can be used safely with a previously-unused key:

$$\frac{\begin{array}{c} P \textit{ believes } \{P \overset{K}{\longleftrightarrow} Q\}, \\ P \textit{ believes } oneTimeSafe(M), \\ P \textit{ believes } fresh(K), P \textit{ sees } \{X\}_{<M,K>} \end{array}}{P \textit{ believes } Q \textit{ said } X} \tag{11}$$

In the above rule, if the message X is $safe(M')$, then the recipient of the message acquires belief in the safety of M' from its belief in the one-time-safety of M. It is worth noting that the keys used with one-time-safe modules

may become unsafe and that this is handled by the principals individually refusing to regard them as fresh after they have been used.

The following symbols are used in the arguments concerning the Re-Moduling Protocol:

P : Parent
C : Child
NPK : The next preloaded key used for messages from Parent to Child
NCK : The next key to be used from Child to Parent
OM : The (old) module currently being used for authentication
NM : The new module offered by the Parent for authentication in the future

7.2.1. Concrete Version of the Re-Moduling Protocol (Protocol 3)

We restate Protocol 3 in a brief manner below. In the listing, we do not show the encryption of messages using the session key because we wish to establish that the protocol works even when the session key has been broken by an attacker.

$Msg1.\ P \rightarrow C:\ ChangeModule$
$Msg2.\ C \rightarrow P:\ \{R1,\ Module\ Request\}$
$Msg3.\ P \rightarrow C:\ \{NM,\ R2,\ \{R1\}_{<NM,NPK>}, \{NM, R1\}_{<OM,NPK>}\}$
$Msg4.\ C \rightarrow P:\ \{\{R2\}_{<NM,NPK>}\}$
$Msg5.\ P \rightarrow C:\ Ack$

7.2.2. Initial Assumptions

The initial assumptions about the preloaded authentication key set, the current and new authentication modules, and the challenges ($R1$, $R2$) generated by the Child and Parent are:

$$P\ believes\ P \xleftarrow{NPK} C \tag{12}$$

$$P\ believes\ P \xleftarrow{NCK} C \tag{13}$$

$$P\ believes\ fresh(NPK) \tag{14}$$

$$P\ believes\ fresh(NCK) \tag{15}$$

$$P \text{ believes } \{oneTimeSafe(OM),\ Safe(NM)\} \tag{16}$$

$$P \text{ believes } fresh(R2) \tag{17}$$

$$C \text{ believes } P \xleftrightarrow{NPK} C \tag{18}$$

$$C \text{ believes } P \xleftrightarrow{NCK} C \tag{19}$$

$$C \text{ believes } fresh(NPK) \tag{20}$$

$$C \text{ believes } fresh(NCK) \tag{21}$$

$$C \text{ believes } \{oneTimeSafe(OM)\} \tag{22}$$

$$C \text{ believes } fresh(R1) \tag{23}$$

In our system, the parents have authority over security; therefore, we assume that for any module M:

$$C \text{ believes } P \text{ controls } safe(M) \tag{24}$$

7.2.3. Idealized Protocol

The next step in applying a BAN-like logic is to create the idealized protocol that models protocol messages using statements in the logic. $Msg1$ and $Msg2$ are not part of the idealized protocol because they are in plaintext. The idealized version of $Msg3$ is:

$$C \text{ sees } \{Safe(NM), R1\}_{<OM,NPK>} \tag{25}$$

and the corresponding version of $Msg4$ is:

$$P \text{ sees } \{Safe(NM), R2\}_{<NM,NCK>} \tag{26}$$

7.2.4. Protocol Verification

Using Equations (18), (20) and (25), along with the new Message-Meaning Rule (Equation (11)), we also have:

$$\frac{\begin{array}{c} C \text{ believes } (P \xleftrightarrow{NPK} C),\ C \text{ believes } fresh(NPK), \\ C \text{ believes } oneTimeSafe(OM), \\ C \text{ sees } \{Safe(NM)\}_{<OM,NPK>} \end{array}}{C \text{ believes } P \text{ said } safe(NM)} \tag{27}$$

Since $Msg3$ has signed $R1$ using OM and since C believes that $R1$ is fresh (Equation (23)), it is easy to derive that $R1$ in Equation (25) is also

fresh. Upon applying the Nonce-Verification Rule with Equations (23) and (27), we obtain:

$$\frac{C\ believes\ fresh(Safe(NM),\ C\ believes\ P\ said\ safe(NM)}{C\ believes\ P\ believes\ safe(NM)} \tag{28}$$

Next, the Jurisdiction Rule applied to Equations (24) and (28) yields:

$$C\ believes\ P\ controls\ safe(NM),$$

$$\frac{C\ believes\ P\ believes\ safe(NM)}{C\ believes\ safe(NM)} \tag{29}$$

After Msg4 (Equation (26)) and using Equations (16) and (13) along with the Message-Meaning Rule (Equation (10)), we obtain:

$$P\ believes\ (P \xleftrightarrow{NCK} C),\ P\ believes\ safe(NM),$$

$$\frac{P\ sees\ \{Safe(NM)\}_{<NM,NCK>}}{P\ believes\ C\ said\ safe(NM)} \tag{30}$$

Since $Msg4$ has signed $R2$ using NM and since P believes that $R2$ is fresh (Equation (17)), it also believes that C's utterance of $Safe(NM)$ is fresh. Upon applying the Nonce-Verification Rule, we obtain:

$$\frac{P\ believes\ fresh(Safe(NM)),\ P\ believes\ C\ said\ safe(NM)}{P\ believes\ C\ believes\ safe(NM)} \tag{31}$$

Thus, there are now four beliefs about the modules from the assumption (Equation (14)) and the results (Equations (29), (31) and (28)). They are:

$$P\ believes\ safe(NM)$$
$$C\ believes\ safe(NM)$$
$$P\ believes\ C\ believes\ safe(NM)$$
$$C\ believes\ P\ believes\ safe(NM)$$

Thus, both the participants believe in the safety and usefulness of the new module. However, since NPK and NCK were used with the unsafe module OM, no assumptions can be made about them. At the end of the protocol, both nodes change to the next keys in the list after NPK and NCK.

The procedure for verifying Protocol 4 is generally the same as that for verifying Protocol 3. However, the child now possesses a new belief at the beginning of the protocol:

$$C \; believes \; safe(NM)$$

This is because the child already has a copy of NM and, consequently, this does not need to be established in the argument.

8. Related Work

While security standards such as PKI, IPSEC and TLS are candidates for security architectures for process control systems, they cannot be used without modifications. Consider, for example, if PKI is used in the same manner as pre-shared secret keys in order to reduce the vulnerability due to the wide use of a single root certificate [18]. This usage of PKI introduces the same key complexity and PKI essentially becomes a symmetric key system, eliminating the need to use PKI in this scenario. Another important aspect is that the use of public keys in certificates introduces dependence on the algorithms. This affects longevity because it not possible to change public key algorithms on top of pre-shared keys.

A fundamental problem with IPSEC is that it was not designed with multicast transmissions in mind. Work has been done to provide multicast support for IPSEC, but there is no widely used or tested standard that is suitable for our scenario. Although IPSEC supports different algorithms, it is necessary to ensure that the safety of the modules is reasonably guaranteed if support is extended to distribute them and to use them dynamically [7].

TLS does not support datagram traffic [6]. Although DTLS was designed to add datagram support, algorithms such as RC4 are not supported by DTLS [16]. Consequently, DTLS cannot be relied upon to support new algorithms that are created after a system is deployed.

Different formal analysis methods concentrate on different aspects of protocols. Lowe's method [14] focuses on recentness and degree of agreement. The approaches in [8, 21] consider sequences of events that occur in protocols. Yet other approaches [15, 22] employ logics that are similar to BAN in the way they express beliefs. To our knowledge, no other formal analysis method (including [8, 14, 9, 22, 15, 21]) addresses the need for reasoning about beliefs in the security of modules used in protocols.

Since they are well tested, security standards should be prime candidates for new security architectures for process control systems. However, current standards do not have provisions to deal with situations where the cryptographic algorithms on which they rely are discovered to be broken. It is essential for a system with long life expectancy to be flexible enough in its design to accommodate new algorithms.

9. Conclusions

The protocols presented in this paper specifically address the longevity needs of process control systems used in critical infrastructures. The Authentication Protocol makes use of a preloaded key set as key material to perform mutual authentication. The Re-Keying and Re-Moduling Protocols are vital to maintaining the authentication capabilities of process control systems. The Re-Keying Protocol facilitates the move to fresher and stronger keys in a safe manner. The Re-Moduling Protocol facilitates the distribution of new modules and the transitioning to their use in a secure manner. The protocols utilize the preloaded key set both minimally and efficiently.

Acknowledgements

This work was supported by the National Science Foundation under Grant CNS 05-24695 and by the U.S. Department of Energy under Subcontract 49944 with the Pacific Northwest National Laboratory. We would also like to thank Erik Solum for the initial research related to this work and the anonymous reviewers for their helpful comments.

References

[1] D. Bakken, C. Hauser, H. Gjermundröd and A. Bose, Towards more flexible and robust data delivery for monitoring and control of the electric power grid, Technical Report EECS-GS-009, School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington (gridstat.net/publications/TR-GS-009.pdf), 2007.

[2] A. Bessani, P. Sousa, M. Correia, N. Neves and P. Verissimo, The Crutial way of critical infrastructure protection, *IEEE Security and Privacy*, vol. 6(6), pp. 44–51, 2008.

[3] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer-Verlag, Berlin Heidelberg, Germany, 2003.

[4] M. Brändle and M. Naedele, Security for process control systems: An overview, *IEEE Security and Privacy*, vol. 6(6), 24–29, 2008.

[5] M. Burrows, M. Abadi and R. Needham, A logic of authentication, *ACM Transactions on Computer Systems*, vol. (8)1, pp. 18–36, 1990.

[6] T. Dierks and C. Allen, The TLS Protocol (Version 1.0), RFC 2246 (www.ietf.org/rfc/rfc2246.txt), 1999.

[7] N. Doraswamy and D. Harkins, *IPSec: The New Security Standard for the Internet, Intranets and Virtual Private Networks*, Prentice Hall, Upper Saddle River, New Jersey, 2003.

[8] F. Fabrega, J. Herzog and J. Guttman, Strand spaces: Why is a security protocol correct? *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 160–171, 1998.

[9] D. Gollmann, What do we mean by entity authentication? *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 46–54, 1996.

[10] D. Harkins and D. Carrel, The Internet Key Exchange, RFC 2409 (www.ietf.org/rfc/rfc2409.txt), 1998.

[11] C. Kaufman, R. Perlman and M. Speciner, *Network Security: Private Communication in a Public World*, Prentice Hall, Upper Saddle River, New Jersey, 2002.

[12] S. Kent and R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401 (www.ietf.org/rfc/rfc2401.txt), 1998.

[13] J. Kohl and C. Neuman, The Kerberos Network Authentication Services (v5), RFC 1510 (www.ietf.org/rfc/rfc1510.txt), 1993.

[14] G. Lowe, A hierarchy of authentication specifications, *Proceedings of the Tenth Computer Security Foundations Workshop*, pp. 31–43, 1997.

[15] W. Mao and C. Boyd, Towards formal analysis of security protocols, *Proceedings of the Sixth Computer Security Foundations Workshop*, pp. 147–158, 1993.

[16] N. Modadugu and E. Rescorla, The design and implementation of datagram TLS, *Proceedings of the Eleventh Network and Distributed System Security Symposium*, 2004.

[17] C. Neuman and T. Ts'o, Kerberos: An authentication service for computer networks, *IEEE Communications*, vol. 32(9), pp. 33–38, 1994.

[18] R. Perlman, An overview of PKI trust models, *IEEE Network*, vol. 13(6), pp. 38–43, 1999.

[19] B. Schneier, *Applied Cryptography*, Wiley, New York, 1996.

[20] E. Solum, C. Hauser, R. Chakravarthy and D. Bakken, Modular over-the-wire configurable security for long-lived critical infrastructure monitoring systems, *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, 2009,

[21] D. Song, S. Berezin and A. Perrig, Athena: A novel approach to efficient automatic security protocol analysis, *Journal of Computer Security*, vol. 9(1/2), pp. 47–74, 2001.

[22] S. Yang and X. Li, A limitation of BAN logic analysis on a man-in-the-middle attack, *Journal of Information and Computing Science*, vol. 1(3), pp. 131–138, 2006.