TOKEN BASED KEY MANAGEMENT SCHEME FOR SCADA COMMUNICATION

Anupam Saxena, Om Pal, Sharda Saiwan and Zia Saquib

Centre for Development of Advanced Computing, Mumbai, India
{anupam, ompal, sharda, saquib}@cdacmumbai.in

ABSTRACT

Security of SCADA (supervisory Control and Data Acquisition) has become a challenging issue today because of its connectivity with the outside world and remote access to the system. One major challenge in the SCADA systems is securing the data over the communication channel.

PKI (public key infrastructure) is a well known framework for securing the communication. In SCADA system, due to limited bandwidth and rare communications among some nodes (Remote Terminal Units), there is a need of customization of general PKI which can reduce the openness of Public Key, frequent transfer of certificates and reduction in DOS (Denial of Service) attacks at MTUs (Master Terminal Units) and other nodes.

This paper intends to address the issues of securing data over communication channel in the constrained environment and presents the novel solutions pivoted on key distribution and key management schemes.

KEYWORDS

Key Distribution and Management, Public Key Infrastructure, SCADA security.

1. INTRODUCTION

Supervisory Control and Data Acquisition (SCADA) systems means for management, supervisory control, and monitoring of process control and automation systems via collecting and analyzing the real time data. Initially these systems were not intended to operate within the enterprise environment, this lead to inability within SCADA components to deal with the exposure to viruses, worms, malware etc. that are commonplace today within the enterprise network.

Due to connectivity of SCADA systems with Internet and the increased risk of cyber attacks, security of such systems have become a challenging issue today. Technology become vulnerable to attacks and technological vulnerability can cause a sever damage on critical infrastructures like electric power grid, oil gas plant and water management system. Protection of such Internet connected SCADA systems from intruders is a new challenge for researchers and therefore, it is necessary to apply information security principles and processes to these systems. Researchers recognized that these systems need to operate safely, efficiently, and securely; and pointed out that its cyber vulnerabilities are substantial and have already caused significant impacts including deaths [13].

SCADA system consists of a human-machine interface (HMI), a supervisory system (controller or MTU), Nodes (remote terminal units), programmable logic controllers (PLCs) and a communication infrastructure connecting the supervisory system to the nodes.

As the SCADA industry developed, vendors began to adopt open standards and the total number of SCADA protocols commonly in use was reduced to smaller number of protocols that were popular and were being promoted by industry, including MODBUS, Ethernet/IP, PROFIBUS, ControlNet, InfiNET, Fieldbus, Distributed Network Protocol (DNP), Inter-Control Center Communications Protocol (ICCP), Telecontrol Application Service Element (TASE) etc. The most widely used communication protocols in SCADA system are DNP3 (Distributed Network Protocol version 3.0), IEC 62351 and Modbus. In the beginning due to isolation of SCADA system from rest of the world, cyber security was not an issue when these protocols were designed. As the system is becoming more interconnected to the outside world, the necessity of securing the system is increased.

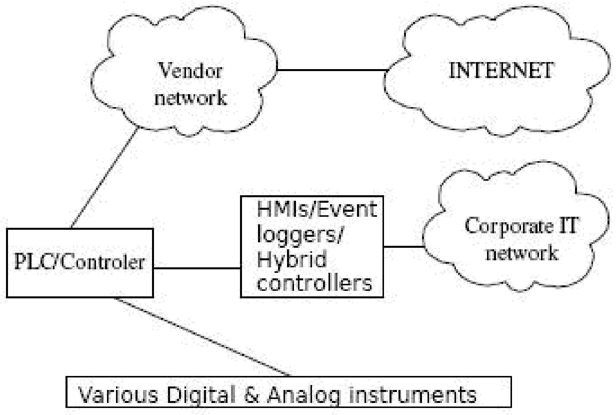

Vendor network
INTERNET
HMIs/Event loggers/ Hybrid controllers
Corporate IT network
PLC/Controler
Various Digital & Analog instruments

Figure 1. SCADA System Architecture

Cryptographic techniques are widely used for providing many security features like higher security, reliability, and availability of control systems etc. to the SCADA systems. There is a need of establishment of secure keys before application of cryptographic techniques. Specifically designed PKI are much easier to work with in order to address a particular problem, rather than using a "onesize-(mis)fits-all PKI design" [15].

In this paper first we discuss key distribution and key management issues and then we present our Token Based Key Management Scheme for securing the SCADA communication in an efficient way; the scheme is designed such that it also fulfils the essential requirement of availability along with integrity in the SCADA systems.

In next section, we discuss key challenges and related work, In Section 3 we present our proposed key distribution and key management technique with Protocol Security and Strength Testing in section 4, Conclusions in Section 5 and References at the end.

2. KEY CHALLENGES AND RELATED WORK

Along with the connectivity of SCADA system to the Internet, many security threats have emerged, like unauthorized access of devices, capturing and decoding network packets and malicious packet injection in the network.
For securing the SCADA system from these threats, there are certain security requirements, which can be classified as:

1. Authentication: It is very important to ensure that the origin of an object is what it claims to be.
2. Integrity: The manipulation of messages between nodes and insertion of new nodes can be hazardous. A malicious attacker could cause physical damage if they have the ability to alter or create messages.
3. Confidentiality: Ensuring that no one can read the message except the intended receiver.

4. Availability of resources: Insuring that resources are available for legitimate users. Insuring that the information is there when needed by those having authorization to access or view it.

For securing the system, these challenges along with installation and configuration limitations of the system need to be considered. Ludovic Piètre-Cambacédès[3] has pointed out the some constraints of SCADA system:

1. Limited computational capacity: The most of the nodes are having low computational capabilities.

2. Limited Space Capacity: Memory available in the most of the nodes is quite low.
3. Real-time processing: If transmission and processing of data in SCADA systems not become timely, then it may cause of latency problems.
4. Key freshness: In the absence of key freshness entities would keep reusing an ‘old’ key for longer time, which might have been compromised, so there is a need of key freshness for eliminating the possibility of such new security hole.
5. Small number of messages: Due to low bandwidth, number of messages exchanged between nodes need to be minimum and also length of messages need to be also small.

In the paper "Infrastructure Vulnerability Assessment Model (IVAM)" [11], Barry Charles presents a model that quantifies vulnerabilities of critical infrastructure (here, medium-sized clean water system) using the Infrastructure Vulnerability Assessment Model (I-VAM). Author emphesized on the use of said system to quantify vulnerabilities to other infrastructures, Supervisory Control and Data Acquisition Systems (SCADA), and Distributed Control Systems (DCS).

In the paper "Security for Critical Infrastructure SCADA Systems" [24], Andrew Hildick-Smith, gave a non-technical overview of critical infrastructure SCADA security. It gives relevant information of background on SCADA systems and the history of critical infrastructure concern. Various SCADA security threats, incidents and vulnerabilities are discussed and a broad range of security initiatives, observations and recommendations are provided.

American Gas Association in its report 12 "Cryptographic Protection of SCADA Communications"[14], identifies both the threat agents (entities who might harm the system) and the kinds of attacks that might be mounted. This ranges from hacker with spare time whose motivation may be fun, challenge or fame, to terrorists with computer skills, spying, money whose motivation is to terrorize, finance operations and economic damage. Maintenance communication channel protection components are explained for Cryptographic experts to understand the special constraints of SCADA systems, and for SCADA engineers and designers in order to protect against cyber attack.

In its SCADApedia pages [12], Digital Bond gives insight of various SCADA protocols including DNP3, Secure DNP3, Foundation Fieldbus HSE, Modbus, PROFIBUS/PROFINET. It includes the protocol description, its deployment and use in relevant protocol layer(s) with changes (if any) required for that particular layer of TCP/IP or OSI model.

A depth defence and proactive solutions [4] to improve the security of SCADA control systems ensures the future of control systems and survivability of critical infrastructure. This paper describes the key requirements and features needed to improve the security of the current SCADA control systems. For example, in assessing the risk for SCADA systems, use of general methods for risk analysis including specific conditions and characteristics of a control system need to be applied.

Wang et al. [7] presents a suite of security protocols optimized for SCADA/DCS systems which include: point-to-point secure channels, authenticated broadcast channels, authenticated emergency channels, and revised authenticated emergency channels. These protocols are designed to address the specific challenges that SCADA systems have.

In National Communications system (NCS) [8], an overview of SCADA is provided, and security concerns are addressed and examined with respect to National Security and Emergency Preparedness (NS/EP) communications and Critical Infrastructure Protection (CIP) implementation.

T. Paukatong, in his paper "SCADA Security: A New Concerning Issue of an In-house EGAT-SCADA" [10], described the measures to ensure security for Electricity Generating Authority of Thailand -SCADA. This paper gives an insight of potential attacking techniques from insiders and outsiders. They recommended North America Electricity Reliability Council (NERC) standard 1300-Cyber Security as an important source of security guidelines.

There is a need to keep these constraints in mind before building a security mechanism for the system. Many efforts have been made in the area of key distribution and key management for securing the System but still there is a scope for improvements.

Sandia National Laboratories [1] proposed a cryptographic key management and Key Establishment approach for SCADA (SKE) in 2002. This technique, divides the communication into two categories: first is 'controller to subordinate (C-S) communication' and second is 'subordinate to subordinate (S-S) communication'. The C-S is a master-slave kind of communication and is ideal for symmetric key technique. The C-C is a peer-to-peer communication and it can use asymmetric key approach. In C-S communication, each controller has a Long Term Key (LTK) shared with its subordinate. The controller also has its own General Seed Key (GSK), which it sends to each of its subordinates. The General Key (GK) is a 128 bit hash of GSK. For communication, the sender obtains a Session Key (SK) from GK. And this SK is used for encryption/decryption. All keys used are of 128 bit in length.

Information Security Institute, Queensland University of Technology, Australia [2] proposed Key Management Architecture for SCADA systems (SKMA). In this scheme a new entity 'Key Distribution Center (KDC)' came into picture, which is used to maintain long term keys for every node. Whenever a new node joins the system, a node-KDC key is manually installed in it. When two nodes want to communicate then with help of node-KDC key, a long term 'node-node key' is generated. Again using the node-node key, a session key is generated for data communication.

In 2002, Mingyan Li [5] proposed a key management approach with multicast and broadcast facility. This approach specifies the shared keys to be stored in the database of MTU (2n-1 keys) and nodse (1+log 2n keys) and these keys are used at run time, where 'n' is number of nodes. However, this approach provides multicasting in a limited fashion.

Donghyun Choi[6] also proposed a multicast and broadcast scheme with additional computation at run time at MTU side, by doing so the number of keys at MTU is 'n-1' lesser than Mingyan approach. Like Mingyan's approach, this approach also provides multicasting in a limited fashion.

Simple Public Key Infrastructure (SPKI) was developed starting in 1995. Simple Distributed Security Infrastructure (SDSI) is a new design for a public-key infrastructure, designed by Professors Ronald L. Rivest and Butler Lampson of MIT's Laboratory for Computer Science, members of LCS's Cryptography and Information Security research group [18]. The SPKI/SDSI facilitates to build a secure distributed computing system which may be scalable. SPKI/SDSI builds public keys as principals and each public key as a certificate authority itself [17]. Each principal can issue certificates. SPKI/SDSI provides two types of certificates; these are "name certificates" and "authorization certificates". Name certificate defines a local name in the local name space of certificate issuer. Authorization certificate grants authorization to the subject of the certificate. A single certificate cannot define both i.e. a name and granting an authorization; so a certificate is either a name certificate or an authorization certificate, but can not be both.

SCADA system is an interconnected infrastructure, where smooth, reliable and continuous operations are desired. Protecting such infrastructures includes a number of challenges, such as secure interaction among nodes, resilience and robustness of entire system. The Wireless Sensor Networks (WSN) have intelligent distributed control capabilities, and the capability to work under severe conditions, so some of the schemes of this area may be useful for securing SCADA systems, as μPKI.

In the paper "Lightweight PKI for WSN μPKI", Benamar Kadri , Mohammed Feham , and Abdallah M'hamed proposed a lightweight implementation of Public Key Infrastructure (PKI) [16]. Their proposed protocol called μPKI uses public key encryption only for some specific tasks as session key setup between the base station and sensors giving the network an acceptable threshold of confidentiality and authentication. μPKI only implements a subset of a PKI services. Here all sensors are connected to a Base station, which is having more computational and energy power compared to sensors; and each sensor is capable to use both symmetric and asymmetric encryption. The public key of the base station is installed at sensor node with the help of an off-line dealer. It ensures that only legitimate sensors can authenticate base station trough its public key. The public key is used to authenticate the base station by the sensors in the network, and private key is used by the base station to the decrypt data sent by sensors, which ensures confidentiality. For secure end to end transmission between nodes and Base station, μPKI uses two types of handshakes. The first handshake is between the base station and sensors where a sensor generates a random key, encrypts it with the public key of the base station and sends to Base station, by decrypting it , the base station saves the session key in a global table where are saved all the session keys corresponding to each sensor in the network. The second handshake is for securing sensor to sensor communication; where one of the two sensors sends a request (which contains the identifier of the corresponding node) to the base station to establish a secure tunnel with the other sensor. When base station receives this request, it decrypts this and generates a random key, then encrypts a copy of this key for each sensor using the corresponding session keys, and sends it to each sensor [16].

Tanveer Ahmad Zia proposed a novel security framework for wireless sensor networks WSNSF (Wireless Sensor Networks Security framework) [9] that includes a secure key management scheme, secure routing algorithm, secure localization technique and a malicious node detection mechanism.

Several schemes have been proposed in the area of token based key management for the security of data & information. The 'hybride scheme' [19] incorporates an electronic token and biometric verification. In this scheme the template against which the user's biometric is validated is encrypted and divided into two parts. One is recorded on electronic media as part of the user's token and the other is retained inside the secured system. The key is generated independently instead of using user's biometric. This is also encrypted, split and stored in the same two locations. The only drawback of this scheme is, it can not be used in automated systems because user interference is required for biometric verification.

Another hybrid cryptographic technique [20] uses a combination of RSA and elliptic curve cryptography (ECC) to achieve efficient mutual authentication and key agreement. Here a trusted third party generates the certified tokens and the nodes need to present their credentials (Social Security Number SSN of the user, name & address of the user, MAC address of the device) in order to get the tokens. In this protocol, there is no prior key distribution and key storage for making protected data transmission in vulnerable wireless link. This technique requires reasonable resources in terms of computation and communication overhead and provides higher security.

Gokhan Bal et al. [21] have proposed a key management architecture based on the capabilities of Trusted Computing (TC) Technologies. It uses Trusted Software Stack to implement its functionality. To achieve a maximum level of universality, the services provided by this architecture covers the needs of all applications dealing with privacy sensitive data. An application programming interface facilitates the utilization of the services.
In a ubiquitous security solution proposed by Jiejun Kong et al. [22], the certification authority functions are distributed through a threshold secret sharing mechanism, in which each entity holds a secret share and multiple entities in a local neighborhood jointly provide complete services.

A large number of security protocols are being developed and deployed in order to provide secure communication. The design of any security protocol is an intuitive process which is severely error-prone. That's why a more rigid framework required; within which one can safely design protocols. BAN logic is the most important tool to have a formalization analysis of authentication protocols [23]. BAN Logic does not properly deal with the issues of certificates and the use of Public Key Infrastructure (PKI), in the paper "Extending BAN Logic for Reasoning with Modern PKI-based Protocols" [25], Sufatrio and Roland H.C. Yap proposed an extension to BAN Logic that focuses on certificate processing within the PKI setting. Their work makes possible to use BAN Logic on PKI-based protocols

The analysis of security protocols is being difficult for humans, as many protocols were found to be awed after deployment. The prior efficient approaches to do the automated falsification or verification of such protocols were ProVerif or the Avispa tools. In the paper "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols" [26], Cas Cremers presented a push-button tool, called Scyther, for the verification, the falsification, and the analysis of security protocols.

Though the use of any such tool eliminates the possibility of human error, but still the selection of such automated tool is very important in order to find out the correct results. Only few tools explore all possible behaviours, whereas others explore strict subsets. Ignoring these kinds of differences leads to completely wrong interpretations of the output of a tool. In their report "Comparing State Spaces in Automatic Security Protocol Verification" [27], Cas Cremers and Pascal Lafourcade applied study of state space relations in performance comparison of several well-known automatic tools for security protocol verification. After the analysis of performances of tools over comparable state spaces, they find in their conclusion about the efficiency of the tools that Scyther and ProVerif are the fastest, their approximation techniques are effective, and both can handle unbounded verification. Scyther tool has the advantage of not using approximations.

In this paper, we concentrate on accomplishment of fundamental security goals of communications, where secure communication is needed with limited resources. We are presenting the Token Based Key Management approach for the constrained environment of SCADA communication. The strength and security of the proposed protocol is tested by the well known Scyther Tool and also mathematically tested with BAN Logic.

3. PROPOSED METHODOLOGY

Our scheme assumes that messaging takes place among two entities namely MTU and RTU (end nodes). Scheme assumes that MTU has high computing capability to take most of the computational load in order to provide security in SCADA systems.
The Scheme uses asymmetric key approach and the Token fields for securing the communication.

In actual SCADA network, there are Sub-MTUs associated to MTU which takes care of a particular section of nodes. For simplicity, we are taking MTU in place of Sub-MTU, which takes care of their corresponding subsection of nodes to provide security. In turn, these various subsections communicate with each other with the help of their representative MTUs which can communicate with each other by establishment of trust among peer MTUs. Scheme assumes that for a sub section, there is an MTU (with high computational power) and n number of nodes (with low computational power). MTU and nodes are attached to each other.

Initially long term keys are stored manually at each node, 'n' unique keys stored at MTU (corresponding to each NODE), and one at each node which belongs to that corresponding node.

NODE Ri has key L_i where i = 1,2,...n. The MTU passes Private Keys pair to the corresponding nodes by using the pre shared-keys (Li, where i = 1,2,...'n'). Also it passes MTU's Public key to each node.

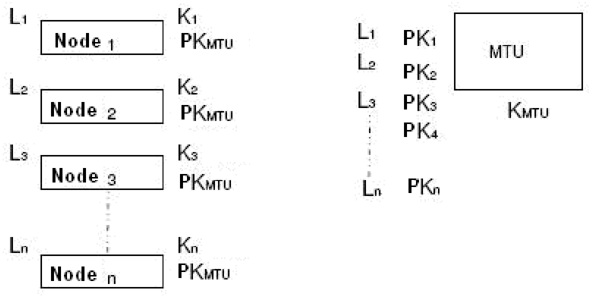


Figure 2. Initial Key Setup

For maintaining key freshness, there is a provision of re-distribution of Private Keys after certain period of time using long term keys, and also these long term keys would be replaced manually after a long time period. This fixed time can be adjusted depending on the requirement of the system.
In the scheme, Private Key is denoted by K, and Public Key is denoted by PK.

We propose the specific use of Tokens and its fields for securing the communications. A token can be generated by an MTU and only MTU can distribute it among participating entities. A Token contains the following fields:

- Addresses (both IP and MAC address) of communicating two NODEs.
- Expiry time of the token.
- Signed (by K_{MTU}) Hash of all above values; we call it “Token Fingerprint”.

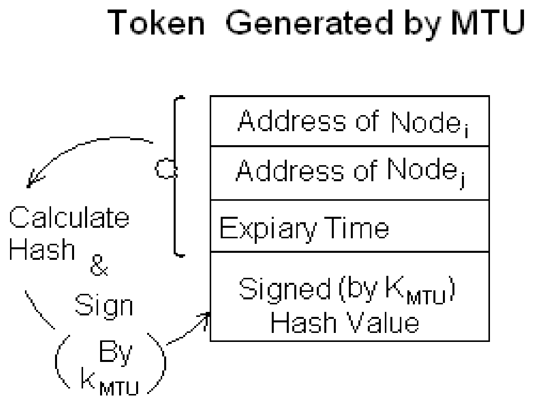


Fig.3. Token

In rest of the paper, we represent a Token by T, i.e. token for NODE A & NODE B is represented by T_{AB}, token for MTU & NODE B is represented by T_{MB}, token for NODE A & MTU is represented by T_{MA}. We also represent the token fingerprint by the '\$' sign.
Whenever a MTU wants to communicate to any node, it can communicate after generating and sending a token for itself & that node. When, a NODE wants to communicate to another NODE or to MTU, it needs a token from MTU.

In general PKI architectures, a CA authenticates any node on the network by issuing certificate to that node but in this case, Here the MTU takes the responsibilities of a CA, extra computational overhead of certificate might be an issue for some nodes because of their low processing power. In this case we have two options, first one is to reduce certificate as much as possible by removing unnecessary extra fields from it [3], and second is to replace certificate value with a single unique value like "MAC Address" of the corresponding entity. In the proposed scheme, node uses Token fingerprint (\$), to prove the authenticity.

For load balancing, and to avoid an MTU to become a single point of failure, scheme also recommends the deployment of distributed MTUs.

Table 1. Notations used in the scheme

Long Term Key	L
Public Key	PK
Private Key	K
Token	T
Token Fingerprint	\$
Nonce	N

3.1. Proposed scheme categorizes the communication into three categories as follows

1. NODE to NODE communication.
2. MTU to NODE communication.
3. NODE to MTU communication.

1. **NODE to NODE communication**

In some cases, a NODE may be interested in communicating with another NODE, in this case NODE will not store Public Keys of all nodes but it will only store Public Key of other NODE at run time if it is needed, which it takes from MTU along with a Token that ensures that both nodes can communicate with each other for the specified time period.

If NODE A wants to communicate with NODE B then it checks the Public Key of NODE B in its database, if it is there then it shows that NODE A and NODE B already has the Token for communication. NODE A encrypts the message and ‘Token Fingerprint ($)’ with Public Key of NODE B and sends to NODE B. If Public Key of NODE B is not available in the database of NODE A; NODE A then encrypts nonce N and address of NODE B the Private Key of itself and again by Public Key of MTU, and finally sends it on open channel to MTU.

When MTU gets this request, it decrypts the encrypted value by its Private Key and by Public key of NODE A. MTU fetches the nonce ‘N’ and prepares a Token for both nodes A and B, i.e. T_{AB}. It chooses a nonce ‘N1’, encrypt it along with the fetched nonce ‘N’ , Public Key of B and Token by public key of NODE A. MTU sends it to the NODE A.

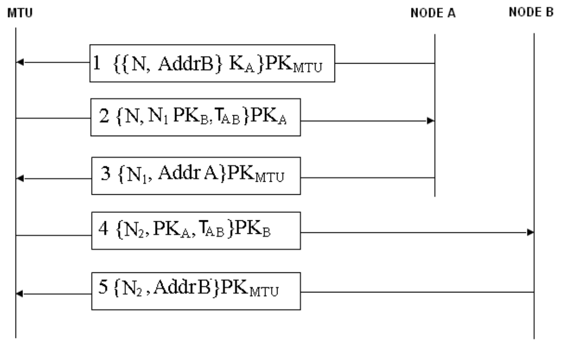

MTU
NODE A
NODE B
1 $\{\{N, AddrB\} K_A\}PK_{MTU}$
2 $\{N, N_1 PK_B, T_{AB}\}PK_A$
3 $\{N_1, Addr A\}PK_{MTU}$
4 $\{N_2, PK_A, T_{AB}\}PK_B$
5 $\{N_2 , AddrB\}PK_{MTU}$

Fig. 4. NODE to NODE Token establishment

After receiving the response from MTU, NODE A decrypts it by its own Private Key and fetches the values from it.

NODE A also decrypts hash value of Token Fingerprint ($) of token T_{AB} by the Public Key of MTU, and compares hash of Token fields with received hash value, if values match, then only NODE A stores the Token Fingerprint ($) in its database.

After this, NODE A stores the Token Fingerprint ($) and Public Key in its database. NODE A then matches the nonce N with the stored value, if it matches then it prepares an Acknowledge response by encrypting the nonce ‘N1’, and ’Address of A’ by Public Key of MTU and sends it to MTU.

On receiving proper nonce ‘N1’ from NODE A, MTU recognizes that the NODE A got the Token and Public Key of NODE B. Then the MTU takes another nonce ‘N2’ and encrypt it along with the Public Key of ‘A’, and the Token by the Public Key of NODE B and sends the resulting block to NODE ‘B’.

After receiving this block from MTU, NODE ‘B’ decrypts it by its own Private Key, and fetches the values from it.

NODE B also decrypts hash value of Token Fingerprint ($) of token T_{AB} by the Public Key of MTU, and compares hash of Token fields with received hash value, if values match, then only NODE B stores the Token Fingerprint ($) in its database.
After this, NODE B stores the Token Fingerprint ($) and Public Key in its database. NODE B then sends the Acknowledge response by encrypting the nonce 'N2', and 'Address of B' by Public Key of MTU and sends it to MTU.

On receiving proper nonce 'N2' from NODE B, the MTU recognizes that the NODE B got the Token and Public Key of NODE A.

Now NODE A sends message to NODE B by encrypting the message and 'Token Fingerprint' with Public Key of NODE B. After receiving it, the NODE B starts communication (as the NODE B already received the Public Key of NODE A along with the token).

2. MTU to NODE communication

If MTU wants to communicate to NODE B then it generates a Token for NODE B and MTU, i.e. T_{MB} and sends this along with a nonce 'N' to NODE B by encrypting it with the Public Key of NODE B.

After getting this type of message of MTU, NODE B decrypts the Block by its own Private Key, it fetches the values from it.

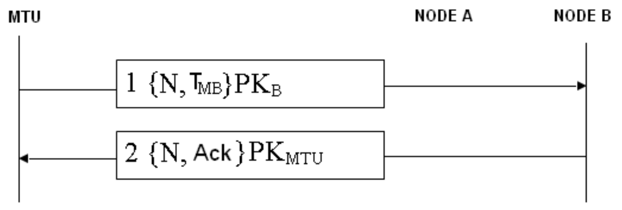


Fig. 5. MTU to NODE Token establishment

NODE B also decrypts hash value of Token Fingerprint ($) of token T_{MB} by the Public Key of MTU, and compares hash of Token fields with received hash value, if values match, then only NODE B stores the Token Fingerprint ($) in its database.

After storing the Token fingerprint ($), NODE B it prepares an Acknowledge response by encrypting the nonce 'N', and 'Address of A' by Public Key of MTU and sends it to MTU..

On receiving proper nonce 'N' from NODE B, MTU recognizes that the NODE B got the Token and Public Key of NODE B. Now in order to communicate to NODE B, MTU fetches the Public Key of that NODE from its database, then encrypts the message and the 'Token Fingerprint ($)' by the Public Key of NODE B and sends to NODE B.
As both the MTU and NODE B contain Public Keys of each other and the Tokens, in their databases, so they can communicate with each other.

3. NODE to MTU communication

If NODE A wants to communicate to MTU then it checks the corresponding Token fingerprint ($) in its database, if it is there then it encrypts the message and 'Token Fingerprint ($)' by Public Key of MTU and sends to MTU. If the corresponding Token fingerprint ($) for MTU is not available in the database of NODE A then NODE A encrypts a nonce 'N' by its Private Key and again by the Public Key of MTU. NODE A then sends this block to MTU.

When MTU gets this request, it decrypts the encrypted value by its Private Key and by Public key of NODE A. MTU fetches the nonce 'N' and then it generates a Token for NODE A, i.e T_{MA}. MTU then encrypts the fetched nonce 'N', and Token by public key of NODE A. MTU sends it to the NODE A.
After getting this response from MTU, NODE A decrypts the Block by its own Private Key, it fetches the values from it.
NODE A also decrypts hash value of Token Fingerprint ($) of token T_{MA} by the Public Key of MTU, and compares hash of Token fields with received hash value, if values match, then only NODE A stores the Token Fingerprint ($) in its database.

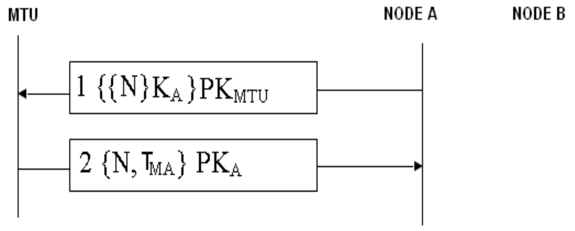


Fig. 6. NODE to MTU Token establishment

After storing the Token fingerprint ($), NODE B it prepares an Acknowledge response by encrypting the nonce 'N', and 'Address of A' by Public Key of MTU and sends it to MTU.

In order to communicate to MTU, NODE A, sends a message to MTU along with the 'Token Fingerprint ($)', by encrypting it by Public Key of MTU. After receiving it, the MTU starts communication as it also has the Public Key of that NODE A and the corresponding Token fingerprint in its database.

Note about communications:

Nodes stores Token fingerprints ($) with them, and keep track of the expiry time of the Token (T); they store the corresponding Public Keys (which comes along with the Token) in their database tables only till the time of Token expiration. After expiry of a token both parties delete the Public Keys (PK values), and Token Fingerprints ($) from their databases and shift all the rows below it by one to above. After expiry if any request comes then it is simply rejected.
Also, the MTU can revoke Public Key of any NODE due to many reasons; in this case it sends a broadcast request to all nodes to delete that particular Public Key and corresponding Token Fingerprint from their databases.

3.2. Dynamic arranged database for optimal key storage

Due to low memory, MTUs and other NODEs can store a limited number of Public Keys in their databases. This limited storage of keys can cause an extra overhead at run time, if required key is not available in database of MTU/NODE.

To overcome this problem, scheme uses 'dynamic arranged database for optimal key storage'. Each NODE/MTU stores Public Key of MTU in first row of database. Always new Public Key will be stored in second row of database and all keys will shift downward by one row. Key at the bottom row is removed if database is already full. If any Public Key is used by the node from its database then this used key will be shifted to second row and all Public Keys (which were above in database from used Public key) will shifted downward by one row. The place of Public Keys those are at lower position from used Public keys will be unchanged.

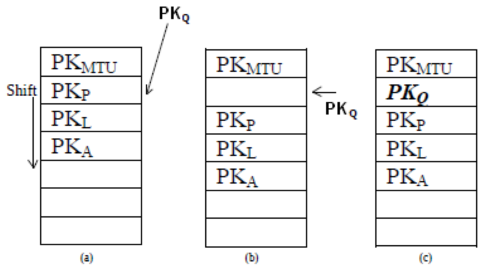


Figure 7. Insertion of new Public Key, when database is partially filled

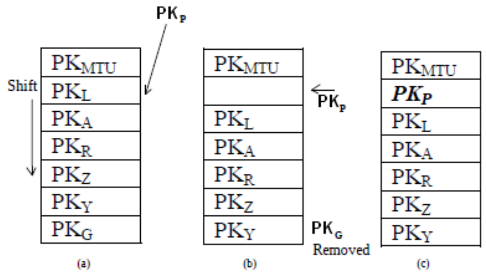


Figure 8. Insertion of new Public Key, when database is fully filled

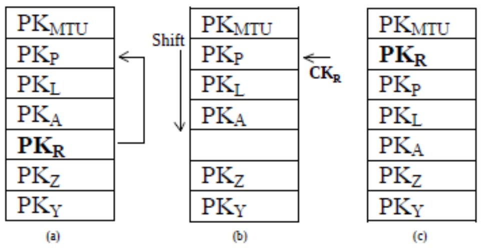


Figure 9. Shifting of existing Public Key within the database, with its use

4. Protocol Strength And Security Testing

The design of any security protocol is an error-prone intuitive process that requires a more rigid framework to safely design protocols. BAN logic is the most important tool to have a formalization analysis of authentication protocols [23]. BAN Logic now also works on PKI-based protocols [25]. In order to test the strength and the security of the proposed protocol, we have used the BAN Logic and Scyther Tool as well, as the analysis of security protocols is being difficult for humans, and many protocols were found to be awed after deployment. Tools like ProVerif, Avispa and Scyther available for the verification, the falsification, and the analysis of security protocols. Again, the selection of such automated tool is very important in order to find out the correct results. Based on the performance comparison report of several such well-known automatic tools "Comparing State Spaces in Automatic Security Protocol Verification" [27], by Cas Cremers and Pascal Lafourcade we find that Scyther and ProVerif are good for the getting the correct testing results. We finalized Scyther tool for protocol strength and security testing, because of its advantage of not using approximations. We have used the BAN Logic for testing Initial Key distribution and Scyther Tool for testing the entire scheme.

4.1. Formal Proof of Protocol by Using BAN Logic

We have done formal proof of proposed protocol by using BAN logic. Formal proof enhances the belief of security on proposed protocol. Formal proof of proposed protocol is given below:

$\xleftarrow{K} A$	: K is Private Key of A
$\xleftarrow{PK} A$	: PK is Public Key of A
$A \overset{K}{\leftrightarrow} B$	: K is symmetric Key between A & B
$A \models X$	: A believes X
$A \Rightarrow X$	: A controls X
$\#(X)$	: X is fresh
$\{X\}_K$	: X encrypted with Key K
$A \triangleleft X$	: A sees X
$A \mid\!\sim X$	: A once said X
$\Phi \mid\!\sim X$	: Somebody said X

Figure 10. BAN logic symbols

Verifications of Key-Public Key Distribution (say, for NODE A)

I Goal: NODE A and NODE B want to establish K_A, PK_A, and PK_{MTU}.
II Share Long Term Symmetric Key L_a

$M \rightarrow A: \{N_a, K_A, PK_A, PK_{MTU}\}L_a$
$A \rightarrow M: \{N_a, K_A, PK_A, PK_{MTU}, \text{Message-1}\}L_a$

Idealizing Protocol

$M \rightarrow A: \{N_a, \xleftarrow{K_A} A, \xleftarrow{PK_A} A, \xleftarrow{PK_{MTU}} M\}L_a$
$A \rightarrow M: \{N_a, \text{Message-1}\}\ L_a$

Assumptions:

$M \models \# N_a$ -- (A_1)
$A \models M \overset{L_a}{\leftrightarrow} A$ -- (A_2)
$A \models M \overset{L_a}{\leftrightarrow} A$ -- (A_3)

Soundness of idealized message 1

$M \models \# N_a$: Holds by (A_1) -- (R_1)

A's beliefs after message 1:

$A \lhd \{N_a, \xleftarrow{K_A} A, \xleftarrow{PK_A} A, \xleftarrow{PK_{MTU}} M\}L_a$ ------------------------------ (R_2)

$A \sqsubseteq M \mid\!\sim \{N_a, \xleftarrow{K_A} A, \xleftarrow{PK_A} A, \xleftarrow{PK_{MTU}} M\}L_a$ by R_2 and A_2 ----------- (R_3)

$A \sqsubseteq \#\{N_a, \xleftarrow{K_A} A, \xleftarrow{PK_A} A, \xleftarrow{PK_{MTU}} M\}L_a$ by A_1, R_3 and fresh inject rule ------- (R_4)

M's beliefs after message 2:

$M \lhd \{N_a, K_A, PK_A, PK_{MTU}, \text{Message-1}\}L_a$ ------------------------------------ (R_5)

$M \sqsubseteq A \mid\!\sim \{N_a, K_A, PK_A, PK_{MTU}, \text{Message-1}\}L_a$ by R_5 and A_3 ---------------- (R_6)

$M \sqsubseteq \# \{N_a, K_A, PK_A, PK_{MTU}, \text{Message-1}\}L_a$ by A_1, R_6 and fresh inject rule -------- (R_7)

Summary of protocol:

$M \sqsubseteq \xleftarrow{K_A} A$

$M \sqsubseteq A \sqsubseteq \xleftarrow{K_A} A$

$M \sqsubseteq \xleftarrow{PK_A} A$

$M \sqsubseteq A \sqsubseteq \xleftarrow{PK_A} A$

$M \sqsubseteq \xleftarrow{PK_{MTU}} M$

$M \sqsubseteq A \sqsubseteq \xleftarrow{PK_{MTU}} M$

A's guarantees:

$A \sqsubseteq \xleftarrow{K_A} A$

$A \sqsubseteq M \sqsubseteq \xleftarrow{K_A} A$

$A \sqsubseteq \xleftarrow{PK_A} A$

$A \sqsubseteq M \sqsubseteq \xleftarrow{PK_A} A$

$A \sqsubseteq \xleftarrow{PK_{MTU}} M$

$A \sqsubseteq M \sqsubseteq \xleftarrow{PK_{MTU}} M$

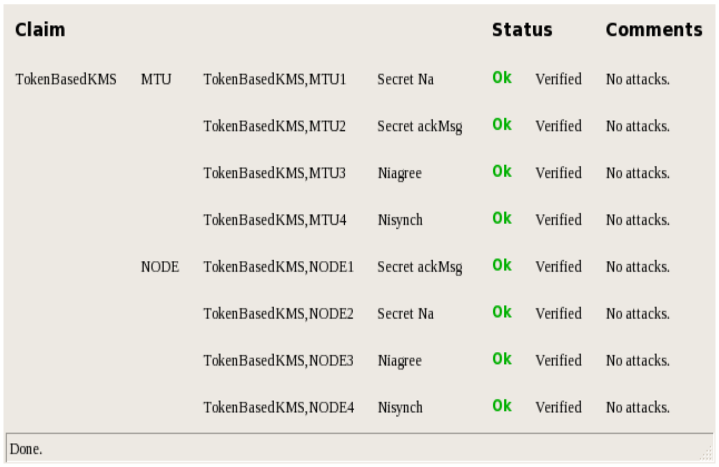
Result: Initial Key distribution is tested and proved secure.

4.2. Protocol Testing with Scyther Tool

Scyther is a tool for the verification, falsification and the analysis of security protocols, where it is assumed that all cryptographic functions are perfect. Scyther provides a number of novel

features that include the possibility of unbounded verification with guaranteed termination, analysis of infinite sets of traces in terms of patterns and support for multi-protocol analysis.
Scyther is based on a pattern refinement algorithm, providing concise representations of (infinite) sets of traces. This allows the tool to assist in the analysis of classes of attacks and possible protocol behaviors, or to prove correctness for an unbounded number of protocol sessions.

1. **Verification of Initial Key Distribution, using Scyther**



Claim				Status		Comments
TokenBasedKMS	MTU	TokenBasedKMS,MTU1	Secret Na	Ok	Verified	No attacks.
		TokenBasedKMS,MTU2	Secret ackMsg	Ok	Verified	No attacks.
		TokenBasedKMS,MTU3	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,MTU4	Nisynch	Ok	Verified	No attacks.
	NODE	TokenBasedKMS,NODE1	Secret ackMsg	Ok	Verified	No attacks.
		TokenBasedKMS,NODE2	Secret Na	Ok	Verified	No attacks.
		TokenBasedKMS,NODE3	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,NODE4	Nisynch	Ok	Verified	No attacks.

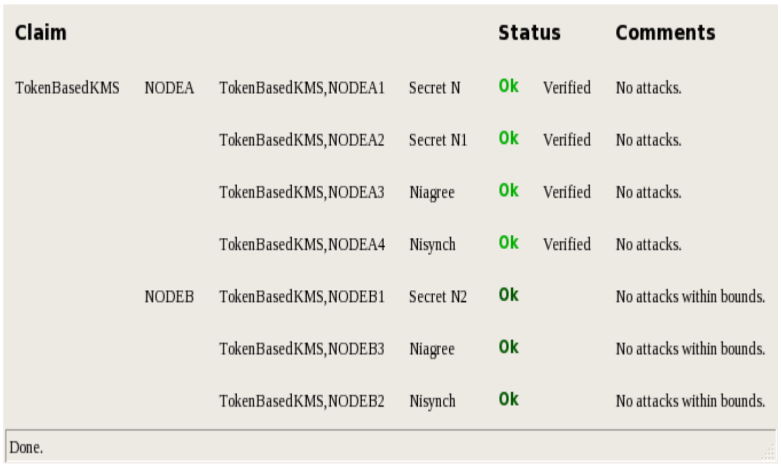
Done.

Figure 11. Initial Key Distribution

The result window shows that the secrecy of Na & ackMsg and the claim for non-injective agreement and non-injective synchronization at both nodes are successfully verified, where Na is a nonce and ack is the acknowledge message sent by NODE A to MTU to acknowledge the receipt of the token.

2. **Verification of Token Distribution, using Scyther**

CASE1: Token attainment for Node to Node communication



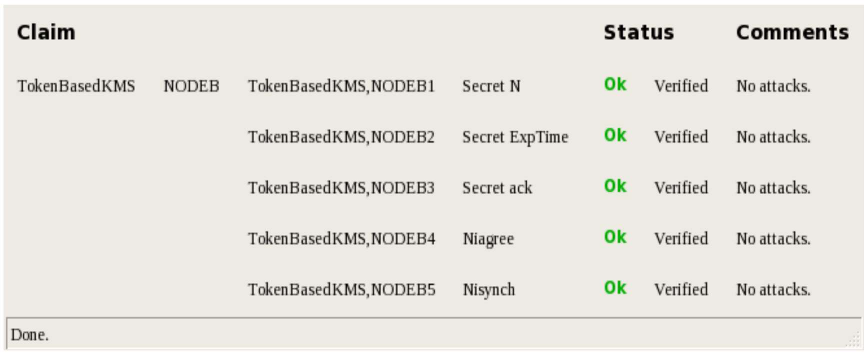
Claim				Status		Comments
TokenBasedKMS	NODEA	TokenBasedKMS,NODEA1	Secret N	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA2	Secret N1	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA3	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA4	Nisynch	Ok	Verified	No attacks.
	NODEB	TokenBasedKMS,NODEB1	Secret N2	Ok		No attacks within bounds.
		TokenBasedKMS,NODEB3	Niagree	Ok		No attacks within bounds.
		TokenBasedKMS,NODEB2	Nisynch	Ok		No attacks within bounds.

Done.

Figure 12. Token attainment for Node to Node communication

The result window shows that the secrecy of N and N1 at NODE A and N2 at NODE B is successfully verified; where N, N1 and N2 are the nonces. The claim for non-injective agreement and non-injective synchronization at both nodes is also verified.

CASE2: Token attainment for MTU to Node communication.



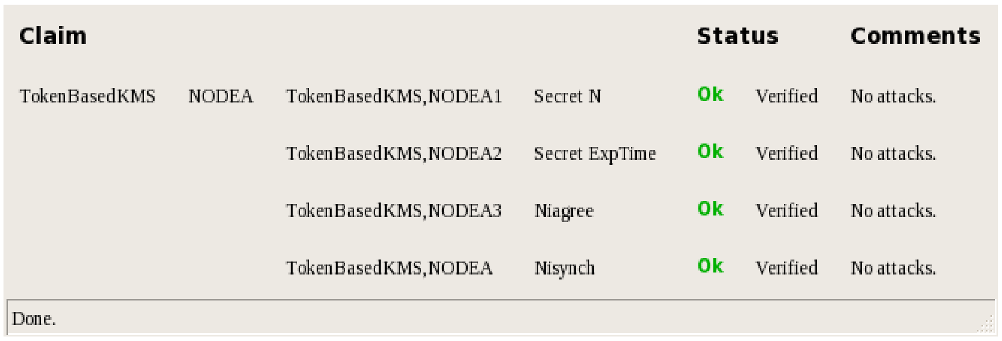
Claim				Status		Comments
TokenBasedKMS	NODEB	TokenBasedKMS,NODEB1	Secret N	Ok	Verified	No attacks.
		TokenBasedKMS,NODEB2	Secret ExpTime	Ok	Verified	No attacks.
		TokenBasedKMS,NODEB3	Secret ack	Ok	Verified	No attacks.
		TokenBasedKMS,NODEB4	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,NODEB5	Nisynch	Ok	Verified	No attacks.

Done.

Figure 13. Token attainment for MTU to Node communication

The result window shows that the secrecy of N, ExpTime & ack and the claim for non-injective agreement and non-injective synchronization at NODE B are successfully verified, where N is a nonce, ExpTime is the expiry time of the token and ack is the acknowledge message sent by NODE B to MTU to acknowledge the receipt of the token.

CASE3: Token attainment for Node to MTU communication



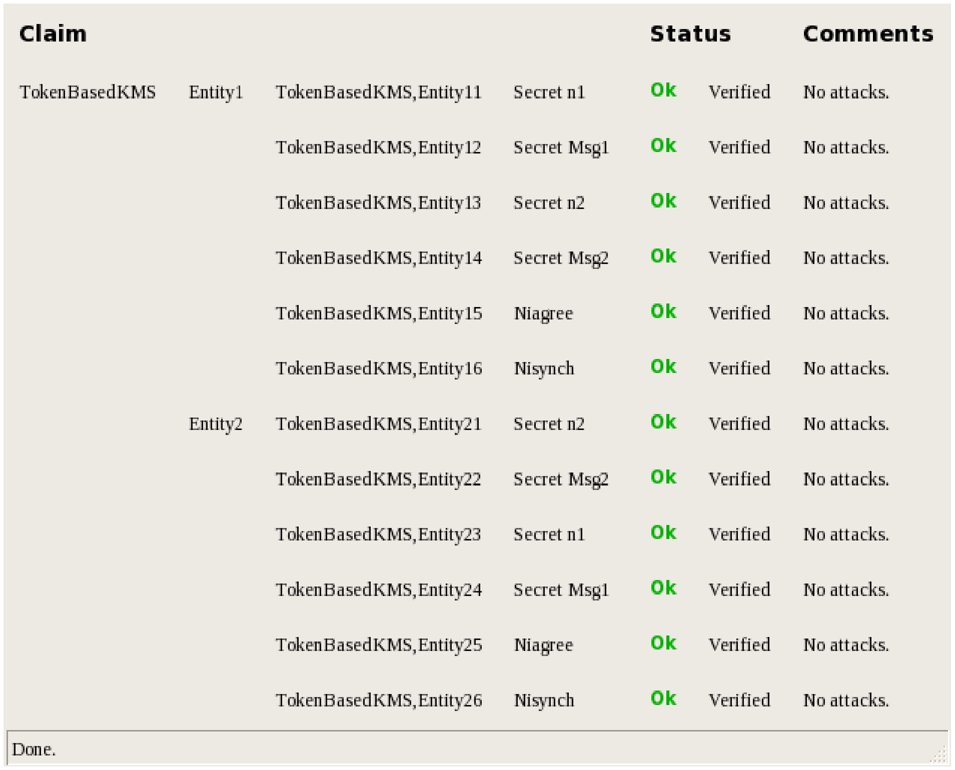
Claim				Status		Comments
TokenBasedKMS	NODEA	TokenBasedKMS,NODEA1	Secret N	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA2	Secret ExpTime	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA3	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,NODEA	Nisynch	Ok	Verified	No attacks.

Done.

Figure 14. Token attainment for Node to MTU communication

The result window shows that the secrecy of N, ExpTime and the claim for non-injective agreement and non-injective synchronization at NODEA are successfully verified, where N is a nonce, ExpTime is the expiry time of the token.

3. **Verification of Messages exchange between any MTU/NODE to any other MTU/NODE after token attainment, using Scyther**



Claim				Status		Comments
TokenBasedKMS	Entity1	TokenBasedKMS,Entity11	Secret n1	Ok	Verified	No attacks.
		TokenBasedKMS,Entity12	Secret Msg1	Ok	Verified	No attacks.
		TokenBasedKMS,Entity13	Secret n2	Ok	Verified	No attacks.
		TokenBasedKMS,Entity14	Secret Msg2	Ok	Verified	No attacks.
		TokenBasedKMS,Entity15	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,Entity16	Nisynch	Ok	Verified	No attacks.
	Entity2	TokenBasedKMS,Entity21	Secret n2	Ok	Verified	No attacks.
		TokenBasedKMS,Entity22	Secret Msg2	Ok	Verified	No attacks.
		TokenBasedKMS,Entity23	Secret n1	Ok	Verified	No attacks.
		TokenBasedKMS,Entity24	Secret Msg1	Ok	Verified	No attacks.
		TokenBasedKMS,Entity25	Niagree	Ok	Verified	No attacks.
		TokenBasedKMS,Entity26	Nisynch	Ok	Verified	No attacks.

Done.

Figure 15. Secure messaging after getting tokens and storing the Token-Fingerprint

The result window shows that the secrecy of nonces N1, N2, Messages msg1, msg2 and the claim for non-injective agreement and non-injective synchronization at both nodes are successfully verified, where N is a nonce, ExpTime is the expiry time of the Token.

5. CONCLUSIONS

SCADA system works in constrained environment, and so there is a need of an efficient key distribution and management scheme to deal with its issues and challenges (as discussed in the above sections). Our attempt is to secure the data over the communication channel in the constrained environment. We have devised and proposed a new key distribution and key management scheme which works in constrained environment, and also utilizes the strength of PKI, by using a token based communication mechanism. The strength and security of the proposed protocol is tested by the well known Scyther Tool and also mathematically tested.

REFERENCES

[1] C. L. Beaver, D.R. Gallup, W. D. NeuMann, and M.D. Torgerson "Key Management for SCADA (SKE)", printed at Sandia Lab March 2002
[2] Robert Dawson, Colin Boyd, Ed Dawson, Juan Manuel, Gonz´alez Nieto "SKMA – A Key Management Architecture for SCADA Systems", Fourth Australasian Information Security Workshop (AISW-NetSec 2006)
[3] Ludovic Piètre-Cambacédès, Pascal Sitbon "Cryptographic Key Management for SCADA Systems, Issues and Perspectives", Proceedings of the 2008 International Conference on Information Security and Assurance (isa 2008) Pages 156-161

[4] Mariana Hentea, “Improving Security for SCADA Control Systems“, Interdisciplinary Journal of Information, Knowledge, and Management Volume 3, 2008

[5] Mingyan Li, R. Poovendran and C. Berenstein “Design of Secure Multicast Key Management Schemes With Communication Budget Constraint”, IEEE Communications Letters, Vol. 6, No. 3, March 2002

[6] Sungjin Lee, Donghyun Choi, Choonsik Park, and Seungjoo Kim” An Efficient Key Management Scheme for Secure SCADA Communication”, Proceedings Of World Academy Of Science, Engineering And Technology Volume 35 November 2008

[7] Yongge Wang and Bei-Tseng Chu “sSCADA: Securing SCADA Infrastructure Communications”, August 2004

[8] http://www.ncs.gov/library/tech_bulletins/2004/tib_04-1.pdf

[9] Tanveer Ahmad Zia “A SECURITY FRAMEWORK FOR WIRELESS SENSOR NETWORKS” PhD Thesis, University of Sydney, February 2008

[10] T. Paukatong “SCADA Security: A New Concerning Issue of an Inhouse EGAT-SCADA”, Electricity Generating Authority of Thailand, 53 Charan Sanit Wong Rd., Bang Kruai, Nonthaburi 11130, Thailand

[11] Barry Charles Ezell "Infrastructure Vulnerability Assessment Model (IVAM)", Risk Analysis, Vol. 27, No. 3, 2007

[12] http://www.digitalbond.com/scadapedia/protocols/

[13] Joe Weiss PE, CISM “Assuring Industrial Control System (ICS) Cyber Security”

[14] American Gas Asociation, "Cryptographic Protection of SCADA Communications Part 1: Background, Policies and Test Plan (AGA 12, Part 1)"

[15] Peter Gutmann “PKI: It’s Not Dead, Just Resting” IEEE Computer, vol. 35, no. 8, pp. 41-49, Aug. 2002

[16] Benamar Kadri, Mohammed Feham, and Abdallah M’hamed “Lightweight PKI for WSN µPKI” accepted for International Journal of Network Security, Vol.10, No.3, PP.194–200, May 2010

[17] Carl Ellison “SPKI / SDSI ” October 2004

[18] http://groups.csail.mit.edu/cis/sdsi.html.

[19] David Argles, Alex Pease, Robert John Walters, "An Improved Approach to Secure Authentication and Signing," ainaw, vol. 1, pp.119-123, 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), 2007

[20] Ammayappan, K.; Sastry, V.N.; Negi, A.; , "Authentication and dynamic key management protocol based on certified tokens for manets," Global Mobile Congress 2009 , vol., no., pp.1-6, 12-14 Oct. 2009

[21] Bal, G.; Schmidt, A.U.; Kuntze, N.; , "Injecting trust to cryptographic Key Management," Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on , vol.02, no., pp.1197-1201, 15-18 Feb. 2009

[22] Jiejun, K.; Petros, Z.; Haiyun Luo; Songwu Lu; Lixia Zhang; , "Providing robust and ubiquitous security support for mobile ad-hoc networks," Network Protocols, 2001. Ninth International Conference on , vol., no., pp. 251- 260, 11-14 Nov. 2001 doi: 10.1109/ICNP.2001.992905

[23] Jinghua Wen, Mei Zhang, Xiang Li “The Study on the Application of BAN Logic in Formal Analysis of Authentication Protocols”, ICEC’05, August 15–17, 2005, Xi’an, China. Copyright 2005 ACM

[24] Andrew Hildick-Smith, "Security for Critical Infrastructure SCADA Systems", GSEC Practical Assignment, Version 1.4c, Option 1, February 23, 2005

[25] Sufatrio, Roland H.C. Yap, "Extending BAN Logic for Reasoning with Modern PKI-based Protocols", 2008 IFIP International Conference on Network and Parallel Computing

[26] Cas Cremers, “The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols”

[27] Cas Cremers and Pascal Lafourcade, “Comparing State Spaces in Automatic Security Protocol Verification”