Public Key Cryptography Based Approach for Securing SCADA Communications

Anupam Saxena, Om Pal, and Zia Saquib

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

Abstract. SCADA (supervisory Control and Data Acquisition) security has become a challenge in today's scenario because of its connectivity with the world and remote access to the system. Major challenge in the SCADA systems is securing the data over the communication channel.

PKI (public key infrastructure) is an established framework to make the communication secure. SCADA systems, normally have limited bandwidth and low computation power of RTUs (Remote Terminal Units); through the customization of general PKI we can reduce its openness, frequent transfer of certificates and possibility of DOS (Denial of Service) attacks at MTUs (Master Terminal Units) and RTUs.

This paper intends for securing data over communication channel in SCADA systems and presents the novel solutions pivoted on key distribution and key management schemes.

Keywords: Broadcasting, Key Distribution and Management, Multicasting, Broadcast, Public Key Infrastructure, SCADA security.

1 Introduction

Supervisory Control and Data Acquisition (SCADA) systems facilitates management, supervisory control, monitoring of process control and automation systems through collecting and analyzing data in real time. Primarily these systems were not build to operate within the enterprise environment, this lead to inability within SCADA components to deal with the exposure to viruses, worms, malware etc.

Internet connectivity of SCADA systems increased the risk of cyber attacks, and hence security of these systems became a challenging issue. Technology become vulnerable to attacks and this may cause 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 makes necessary to apply information security principles and processes to these systems.

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

With the development of SCADA industry, vendors started adopting open standards, which reduced the total number of SCADA protocols to a smaller number of protocols that were popular and were being promoted by industry. These protocols include MODBUS, Ethernet/IP, PROFIBUS, ControlNet, InfiNET, Fieldbus, Distributed Network Protocol (DNP), Inter-Control Center Communications Protocol (ICCP), Tele-control 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. Initially due to isolation of SCADA system from outside world, cyber security was not an issue. With the interconnection of system to the outside world, the necessity of securing the system is increased.

Cryptographic techniques are widely used for providing many security features of 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. In this paper first we discuss key distribution and key management issues and then we present our PKI based approach for securing the SCADA system; this scheme is designed to fulfill 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 Conclusions in Section 4 and References at the end.

2 Key Challenges and Related Work

Connectivity of SCADA system to the Internet, leads to emergence of many security threats, like unauthorized access of devices, capturing and decoding network packets and malicious packet injection in the network.

To protect SCADA system from these threats, certain security requirements are needed to be considered:

1. Authentication: It is 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 may cause physical damage if it has the ability to alter or create messages.
3. Confidentiality: It is to ensure that no one can read the message except the intended receiver.
4. Availability of resources: Resources should be available for legitimate users. It insures that the information is there when needed by those having authorization to access or view it.

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

1. Limited computational capacity: The most of the RTUs have low computational capabilities.
2. Limited Space Capacity: Memory available in the most of the RTUs is low.
3. Real-time processing: Lack of timely transmission and processing of data in SCADA can cause 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 this possibility.
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.
6. Broadcasting: There should be facility of common message announcement to all devices.

These constraints are needed to keep in mind before building any security mechanism for the system. Efforts have been made in this area, but still there exists a wide scope for improvements.

A cryptographic key management and Key Establishment approach for SCADA (SKE) was proposed by Sandia National Laboratories [1] in 2002, which 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.

Information Security Institute, Queensland University of Technology, Australia [2] also proposed a Key Management Architecture for SCADA systems (SKMA). In this scheme a new entity 'Key Distribution Center (KDC)' was introduced, which is maintains long term keys for every node.

In this paper, we concentrate on accomplishment of fundamental security goals of communications. We are using the PKI approach in an optimized manner, which reduces the openness of Public Key in such a way that it makes the communication secure, and provides broadcasting and multicasting (for any group of RTUs) in the limited environment of SCADA system.

3 Proposed Methodology

Our scheme assumes that messaging takes place among two entities namely MTU and RTU. Scheme assumes that MTU has high computing capability to take most of the computational load in order to provide security in SCADA systems.

Scheme uses asymmetric key approach while putting restriction on accessibility of keys. Scheme uses 'Key and CounterKey pair'. Though these keys are analogues to Private-Public keys ('Key' corresponds to Private Key, and 'CounterKey' corresponds to Public key), but are not same in the true sense, because in the scheme, the 'CounterKeys' are not publicly accessible to all.

Scheme assumes that, for a sub section, there is an MTU (with high computational power) and n number of RTUs (with low computational power). MTU and RTUs are attached to each other.

Initially long term keys are stored manually at each node, 'n' unique keys stored at

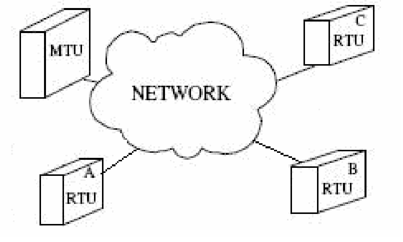

Fig. 1. SCADA System

MTU (corresponding to each RTU), and one at each RTU which belongs to that corresponding RTU.

RTU R_i has key L_i where i = 1,2,...n. The MTU passes 'Key' (K) to corresponding RTUs by using the pre shared-keys (L_i, where i = 1,2,...'n'). Also it passes MTU's CounterKey (CK_{MTU}) to each RTU to each RTU.

For maintaining key freshness, there is a provision of re-distribution of "Key-CounterKey pair" after certain period of time using long term keys; and also these long term keys would be replaced manually after a long (fixed) time period. This fixed time can be adjusted depending on the requirement of system.

Each RTU maintains a database of counter keys of other RTU to which communicates and if the required CounterKey is not available in its database then it requests to MTU for obtaining required CounterKey before initiation of communication. All keys and CounterKeys are confidential, any node can request to MTU to obtain the CounterKey of other node only when it wants to communicate to that node. After the initial communication, both parties (sender and receiver) store the CounterKeys of each other in their database for future use.

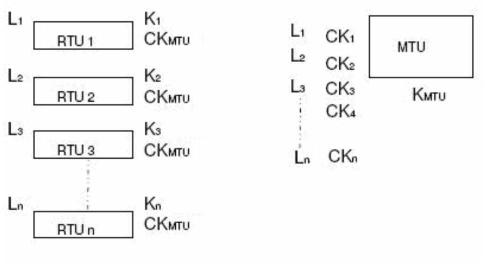

Fig. 2. Initial Key Setup

In general PKI architectures, a CA authenticates any node in the network by issuing certificate to that node but in this case, where MTU takes the responsibilities of a CA, extra computational overhead of certificate might be an issue for some RTUs because of their low processing power. In such cases we have two options, first one is to reduce the 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 encrypts hash of its own MAC address with its own key and forwards this value to other node, to prove its authenticity.

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

Proposed scheme categorizes the communication into three categories as follows

A RTU to RTU communication
B MTU to RTU communication
C RTU to MTU communication

A RTU to RTU Communication

If an RTU wants to communicate with another RTU, it will not store CounterKeys of all RTUs but it will only store CounterKey of other RTU at run time if it is needed, which it takes from MTU.

If RTU A wants to communicate with RTU B then it checks the CounterKey of RTU B in its database if it is there then RTU A encrypts the message with CounterKey of RTU B and sends to RTU B. If CounterKey of RTU B is not available in database of RTU A then it calculates Hash of its own MAC address, signs it by its own Key, and then encrypts the resulting block (along with address of RTU B) by MTU's CounterKey, and sends to MTU. When MTU gets the request, it decrypts it by its own Key, and then checks the signature of RTU A with the help of Hash Function and by using CounterKey of RTU A. After checking the validity of the RTU B, MTU prepares a response. If both RTUs are genuine, then MTU sends this response (which contains CounterKey of corresponding RTU B) to RTU A by signing the Hash of MAC address of MTU by its own key and encrypting the resulting block by the CounterKey of RTU A.

After getting the response of MTU, RTU A decrypts the Block by its own Key and after decrypting hash value by CounterKey of MTU, compares hash of MAC address of MTU with received hash value from MTU, if values match, then RTU A stores the CounterKey of RTU B in its database.

Table 1. Counterkey Storage in Databases of Communicating Rtus

States	**RTU A**	**RTU B**	**Encryption**
Initial State	CK_{MTU}, CK_A	CK_{MTU} CK_B	
State after Message 1	"	"	Message encrypted with CK_{MTU}
State after Message 2	CK_{MTU}, CK_A, CK_B	"	Message encrypted with CK_A
State after Message 3	"	"	Message encrypted with CK_{MTU}
State after Message 4	"	"	Message encrypted with CK_{MTU}
State after Message 5	"	CK_{MTU}, CK_B, CK_A	Message encrypted with CK_B

Now RTU A sends message to RTU B by encrypting the message with CounterKey of RTU B. After receiving it, the RTU B starts communication if it has the CounterKey of RTU A in its database, otherwise it does not respond.

If the RTU A does not get reply from RTU B then it waits for a fixed time (this time duration may vary from system to system and will depend on requirements), after that it sends a communication initiation request to RTU B, by calculating hash of its own MAC address, signing it by its Key, and then encrypting the resulting block (along with address of RTU A) by MTU's CounterKey, and sends to RTU B. After receiving the request from RTU A, RTU B passes this request to MTU. When MTU gets the request, it decrypts it by its own Key, and then checks the signature of RTU A with the help of Hash function and by using CounterKey of RTU A. After checking the validity of the sender (who initiated the request), the MTU prepares a response. If initial sender A is genuine, then MTU sends a response (which contains CounterKey of RTU A) to RTU B by signing the hash of MAC address of MTU by its own key and with encrypting the resulting block by the CounterKey of RTU B.

After getting the response of MTU, RTU B decrypts the Block by its own Key and after decrypting hash value by CounterKey of MTU, compares the hash, if values match then RTU B stores the CounterKey of RTU A in its database. Now the communication can start.

Table 2. Counterkey Storage in Databases of Communicating Mtu And Rtu

States	MTU	RTU B	Encryption
Initial state	CK_{MTU}, CK_B	CK_{MTU}, CK_B	
Message 1	"	"	Message encrypted with CK_B

B MTU to RTU Communication

When an MTU wants to communicate to an RTU then it fetches the CounterKey of that RTU from its database, then MTU encrypts the message with CounterKey of that RTU and sends to it.

As both the MTU and RTU B contain CounterKeys of each other in their databases, so they can always communicate with each other.

C RTU to MTU Communication

When an RTU wants to communicate to MTU then it sends a message to MTU, by encrypting it with CounterKey of MTU (which is known to every RTU). After receiving it, the MTU starts communication as it also has the CounterKey of that RTU in its database.

Table 3. Counterkey Storage in Databases of Communicating Rtu and Mtu

States	**MTU**	**RTU B**	**Encryption**
Initial state	CK_{MTU}, CK_B	CK_{MTU}, CK_B	
Message 1	"	"	Message encrypted with CK_{MTU}

Dynamic Arranged Database for Optimal Key Storage

Due to low memory, MTUs and RTUs can store a limited number of counter 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/RTU.

To overcome this problem, scheme uses 'dynamic arranged database for optimal key storage'. Each RTU/MTU stores CounterKey of MTU in first row of database. Always new Counter 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 CounterKey is used by the node from its database then this used key will be shifted to second row and all CounterKeys (which were above in database from used counter key) will shifted downward by one row. The place of CounterKeys those are at lower position from used Counterkeys, will be unchanged.

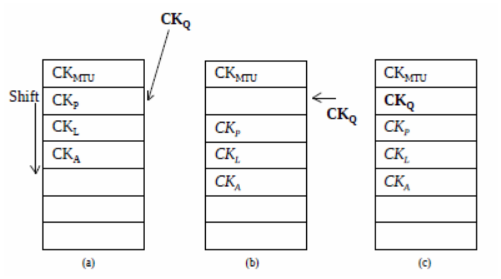

Fig. 3. Insertion of new CounterKey, when database is partially filled

4 Conclusion

In SCADA system, due to limited bandwidth and rare communications among some RTUs (Remote Terminal Units), there is a need of efficient use of general PKI, to reduce the openness of Public Key, frequent transfer of certificates and reduction in DOS (Denial of Service) attacks at MTUs (Master Terminal Units) and RTUs.

We have discussed a variety of existing issues, challenges, and schemes on cryptographic key distribution and key management for SCADA systems. We have devised and proposed this new scheme to emphasize on key distribution and key management in constrained environment using the strength of PKI.

Our attempt is to address the issues of securing the data over the communication channel in the constrained environment and to present a novel solution for key distribution and key management schemes. Proposed scheme supports broadcasting of messages and also provides multicasting as an additional feature for SCADA system.

References

1. Beaver, C. L., Gallup, D.R., NeuMann, W. D., Torgerson, M.D. : Key Management for SCADA (SKE): printed at Sandia Lab (March 2002)
2. Dawson, R., Boyd, C., Dawson, E., Manuel, J., Nieto, G.: SKMA – A Key Management Architecture for SCADA Systems. In: Fourth Australasian Information Security Workshop AISW-NetSec (2006)
3. Piètre-Cambacédès, L., Sitbon, P.: Cryptographic Key Management for SCADA Systems, Issues and Perspectives. In: Proceedings of International Conference on Information Security and Assurance (2008)
4. Hentea, M.: Improving Security for SCADA Control Systems: Interdisciplinary. Journal of Information, Knowledge, and Management 3 (2008)
5. Lee, S., Choi, D., Park, C., Kim, S.: An Efficient Key Management Scheme for Secure SCADA Communication. In: Proceedings Of World Academy Of Science, Engineering And Technology, vol. 35 (November 2008)
6. Wang, Y., Chu, B.-T.: sSCADA: Securing SCADA Infrastructure Communications (August 2004)
7. Paukatong, T.: 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