International Journal of Communication Networks and Security

Volume 2 | Issue 1 Article 10

January 2013

LIGHT WEIGHT SECURITY FOR OPC UA BASED ARCHITECTURE

VARUN SHIVAPRASAD
Department of Computer Science & Engg., Siddaganga Institute of Technology, Tumkur, India.,
VARUNSHIVAPRASAD@gmail.com

KALLINATHA H. D
Department of Computer Science & Engg., Siddaganga Institute of Technology, Tumkur, India.,
KALLINATHA@gmail.com

LIGHT WEIGHT SECURITY FOR OPC UA BASED ARCHITECTURE

VARUN SHIVAPRASAD[1], KALLINATHA H. D.[2]

[1]4th sem M.Tech, [2]Assistant Professor, Department of Computer Science & Engg., Siddaganga Institute of Technology, Tumkur, India.

Abstract- The open platform communication (OPC) is becoming the universal standard for platform independent data sharing and data access. The OPC standard defines communication standards to be followed for data integration in the automation industry. The OPC communication facilitates a language independent way of communication and by using this, all devices can communicate with a centralized or a distributed server in the network. The server will provide clients with facilities such as data monitoring, access and to change values in the devices in the network. The OPC UA provides security add-ons such as security in the transport layer which is mainly the Secure Sockets Layer (SSL). But the SSL adds more load on the security as it involves handshake/key-distribution which consists of a costly public key cryptography. This also puts more load on the server to store and maintain multiple keys involving multiple clients. We believe that a if a framework based on shared key is used, which will be persistent for the whole time duration of the relationship, the communication overhead can be reduced in remote access. This helps in reducing the response time of the already time critical OPC UA based architectures which requires devices to work in real time and put as much less load as possible on the server.

Keywords- *OPC UA; security; SSL in transport layer; shared key; persistent security.*

I. INTRODUCTION

OPC UA is the latest standard from the OPC foundation. This standard is the key for a universal device integration standard. The OPC UA[1] standard defines communication standards for device communication. The devices are mainly the sensors or actuators and the integration is the mechanism or means by which we can manage, integrate and overlook the data produced and accessed by these devices. The integration of these devices enables the clients to use any type of device in the field and integrate them and communicate with all the devices using a single universal, centralized server which integrates and handles all the data produced by the devices. This server will also provide a easy to use interface for the clients to communicate to the devices in the network.

The OPC UA defines the standard communication which involves the devices to speak in the same language for example using EDDL (electronic device description language).

The first generation of device integration standard was OPC-DA (data access). It is the most successful and largely implemented of OPC standards. OPC DA is based on Microsoft COM and DCOM and is specific for only Microsoft based operating systems. It offers the lowest delays and fastest data exchanges in all the OPC standards. But it lacks any security features or web services and is also not scalable.

This prompted the OPC foundation to come up with a more scalable and more secure OPC standard which resulted in OPC UA (unified architecture)[2]. This standard defines coming of age standard which is platform independent and secure. The OPC UA provides web service based on HTTP. There are some versions of the OPC UA which use HTTPS or SSL for security for web services.

There are many variants of OPC UA such as cyber opc, mtconnect and others. Most of these variants use SSL encryption and public key cryptography for data exchange. Public key cryptography is a costly affair and this can affect the time delays of operations which are based on OPC UA architecture. The server can thus suffer higher time delay because of this and this affects the performance of the server and the whole framework of the distributed communication itself in turn[3].

The servers which are based on OPC UA handle the distributed control systems network which is real time and has to use techniques which lower the delays as much as possible and faster communication techniques will only further help in their cause.

Since the servers which are based on OPC UA are real time based and have a stringent restriction on delay times, using public key sharing technique will hinder the performance of these servers.

We make use of shared key technique in cutting down the time taken for public key cryptography by using a proxy on the client side and exchanging a single shared key instead. This will cut down the longer time duration taken by the public key cryptography and help in reducing the delay times of data operations of the server based on OPC.

Rest of the paper is organized as follows: section2 discusses client side key generation, section 3 discusses client server key exchange, section 4 discusses communication using shared key and section 5 is conclusion.

II. EXISTING TECHNOLOGY

The OPC UA uses SSL for security in remote communication between client and the server. The SSL technology uses public key cryptography for providing security.

In public key cryptography method, the client and the server both are provided with one public and one private key. The public key is shared with each other and the private key is unique and resides in respective client or server. The private can only be accessed by its owner. Client will encrypt messages by using public key of the server and this can only be decrypted on the server's side by using the server's private key.

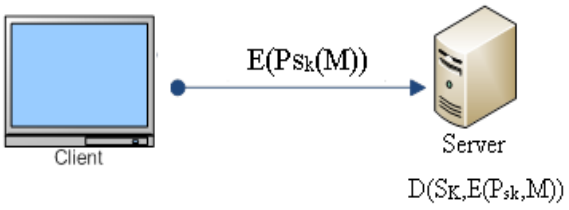

Fig1: Public key cryptography using conventional method

As shown in Fig1, the client encrypts the message using public key of the server. The server needs to decrypt this message and it needs the secret key/private key Sk to do this. Thus after decryption, server gets the intended original message.

III. CLIENT SIDE KEY GENERATION

We propose a client side proxy for key generation. This proxy can be either exist in the client system itself or in another system which the client can trust. There will be a digital signature for authentication between the client and the client proxy which will enable the communication only with the true client proxy.

This proxy will generate shared key on behalf of the client. The client will send server an identity and a shared key. The client will use servers public key to encrypt this and exchange this with the server.

The client side proxy will handle the shared key generation and this is done by considering clients identity (example : email id), servers identity (domain name in servers url), a secret message from client. This key generation is done transparently by the client proxy on client's behalf. The proxy can be located in the client machine itself or alternatively in some other location with which the client has a trusted communication. Upon contacting the client proxy, the client proxy provides a single secret key, which the client will use for the rest of the communication until the session ends with the server. The key will be valid for one single session of the server and a new session will mean a new shared key from the client side.

The key generation uses a function f which uses the client identity, server identity and secret message to generate the key.

$$f(id_c, id_v, s_k) = h(F(id_v)) \,||\, F(id_c) \otimes id_c$$

F is the pseudo - random function (which emulates a random oracle with some criteria)[4]. h is the collision resistant hash function.

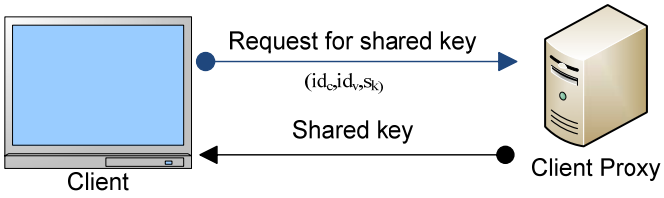

Fig1: Client side key generation

As shown in Fig1, the request for the shared key is placed by the client side and this request gets a corresponding shared key which is generated by the client proxy. The client side key generation will reduce the load on the server. The only job of the server is to accept the shared key from the client and store it along with the client records. The client will identify itself to the proxy using proxy authentication by using recognizable signature.

IV. CLIENT SERVER KEY EXCHANGE

After key generation on the client side, the client will have a secret key which acts as the shared key between client and the server. The client will send the generated key to the server in one of its interactions with the server using public key encryption. The shared key is encrypted using the server's public key.

The server will receive the shared key and decrypt the message using its private key. The server stores this shared key and this is used for the rest of the session with the client. Only if the current session ends will there be a need to generate another shared key. The server stores the shared key along with the clients records

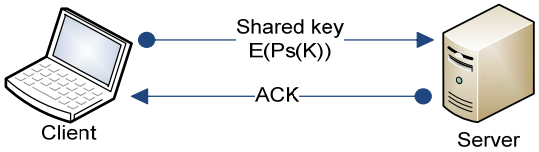

Fig2: Client Server Key Exchange

The server receives the shared key, decrypts it and uses it for the rest of the communication. After receiving the shared key, the server responds to the client with an acknowledgement.

The shared key is used for the entire duration of a single session and this helps in reducing load on the server and the client to use a different key and also provides a better secured communication since it uses shared key.

V. COMMUNICATION USING SHARED KEY

The communication between the client and the server takes place entirely using the shared key received by the client from the client proxy. This will act as the single secret key between the client and the server.

Shared key generated by the client proxy which is used by the client and the server is used for the communication between the client and the server. The key is valid for the entire duration of session of client and server communication.

This shared key is valid for only the duration of the session that it is generated in. If there is some break down of communication between the client and the server, and re establishment of connection, the client will have to get a new shared key generated from the client proxy.

Public key cryptography is used only in the first communication message between the client and the server and this too to only exchange the shared key. The future communication does not need public key cryptography and this results a lot of time and processing power in terms of computational costs.

VI. CONCLUSION

OPC UA is one of the most important transformations ever to take place in the automation industry. The transformation will have to be beneficial to the industry and provide improvements in every regard against the existing standards. And in this regard time management takes the central place as the entire distributed system is real time based. For efficient time management, it is of utmost importance to reduce any delays no matter how minor they might seem and a novel way is proposed here to reduce the delay of public key cryptography.

The proposed scheme considers a shared key scheme to reduce the delay in using public key cryptography. The proposed scheme will avoid generation of several keys. A single shared key is created by using client proxy and this reduces the burden on the client and the server for generating and storing multiple keys. A shared key is generated by the client proxy for the receiving client based on some unique information. This shared key will exist for the whole duration of the client server session. This will avoid macro management of many keys and their generation, in turn reducing the valuable time during communication. The scheme can further be improved by using elliptical curve cryptography.

REFERENCES

[1] Stefan-Helmut Leitner, Wolfgang Mahnke – OPC UA - Service-oriented Architecture for Industrial Applications (2006)

[2] B. Farnham, R. Barillère – Migration from OPC-DA to OPC-UA (2011)

[3] Li Zhao, Ravi Iyer, Srihari Makineni, Laxmi Bhuyan – Anatomy and Performance of SSL Processing

[4] National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions", NIST Special Publication 800-108, November 2008.