Implementation Object Linking and Embedding for Processes Control Unified Architecture Specification on Secure Device

Yuankui Wang (Matr.-Nr.: 6670785)

University of Paderborn wangyk@mail.upb.de

Abstract. Object Linking and Embedding for Process Control Unified Architecture, knows as OPC UA is the most recent released industry standard from OPC Foundation, which compared with his predecessors is equipped with a list of charming new features, with whose help OPC UA is capable of solving imperfections that come along with OPC and offering more functionalities to the end users. And at the same time, the technology of smart card is widely used in information security fields of finance, communication, personal and government identification, payment. Therefor it is meaningful and promising to build OPC UA standard satisfied application on embedded smart card secure device, for the purpose of secure remote control, enterprise resource planning and etc.. The main goal of my master thesis is describing highlighting features of OPC Unified Architecture, especially in security direction, analyzing potential attacks and corresponding countermeasures taken by OPC Unified Architecture, evaluating performance of different possible security polices and extending the transport layer communication sack of OPC UA to support SMS based message exchange for UICC smart card, studying smart card technology and security, at last designing a OPC UA standard based Smart Home to illustrate the implementation of OPC Unified Architecture on secure device.

1 Introduction

OPC from OPC Foundation has already found a great application area in todays industry world, providing a set of standards used to support system interconnectivity and realize a common interface for communications between different products from different vendors. According to [1] there are over 22,000 products applying OPC offered by 3,200 vendors in automation industry.

Even OPC standards are widely accepted, there exit still limitations. I.e. Most of all, OPC is windows platform dependent and based on Microsoft COM/DCOM technology, which is already deemphasized and shows less attraction compared with platform independent Web Services. Moreover although COM/DCOM should help OPC to conquer cross-computer distribution weakness, but it also brings several drawbacks. For instance, developer is not capable of controlling DCOM and has to face frequent DCOM configuration issues[2].Also OPC only supports simple date type information and provides single hierarchy, which apparently

is not able to meet increasing need from user And etc [3]. In order to solve all these imperfections, OPC Unified Architecture comes into the world, which is a radical update of OPC protocols and aimed to achieve simplicity, scalability, outstanding performance, perfect and flexible security, cross platform, always availability, robustness ,supporting complex date types.

In conclusion, OPC Unified Architecture is a platform independent industry policy, supports secure communication based on different network conditions between client and server that are provided by various vendors.

2 OPC Unified Architecture Structure Overview

2.1 OPC UA Specification

The whole OPC Unified Architecture specification can be divided into three main parts, core specification part, which consists of OPC UA concepts, security model, address space model, services, information model, service mapping and profiles, access type specification part including date access, alarm and conditions, programs and historical access, at last utility specification part covering discovery together with aggregates.

In OPC Unified Architecture information that can be visited by clients is defined as address space[4] and there is a set of services[5] provided by OPC UA which are introduced in order to apply operations in the address space. The information in address space is organized as a set of in particular hierarchy structured objects. Clients can accept information provided by OPC Unified Architecture Servers in two major ways, binary structured data and XML documents, depending on the complexity of exchanged date, network quality and so on. In addition three kinds of transport protocol are already defined to support client server communication. They are: OPC UA TCP, HTTP/SOAP and HTTP. Also the hierarchy structure in which objects are organized in address space is also various according to OPC UA standards and not limited to simple single hierarchy.

Another charming feature of OPC UA is Event Notifications. With the help of Event Notification, OPC UA servers are allowed to immediately after some conditions are satisfied publish data, which is subscribed by clients. In this way, clients can for instance discovery failures within client-server-communication quickly and recover as soon as possible, which in return minimizes the lost to the smallest possible amount and also clients are able to observe the subscribed data more precisely and find the pink elephant as fast as possible.

2.2 OPC UA client/ server structure

Figure 1 illustrates a typical OPC UA client server architecture and also describes a combined server-client. The routine communication between client and server consists of requests from client, corresponding responses sent from server and notifications which are generated because of clients early subscription.

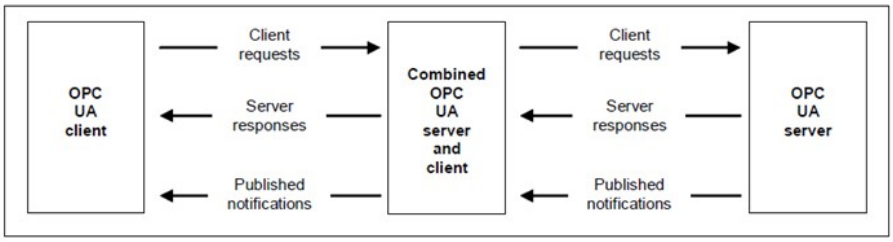

Fig. 1. OPC UA Client Server Structure[6]

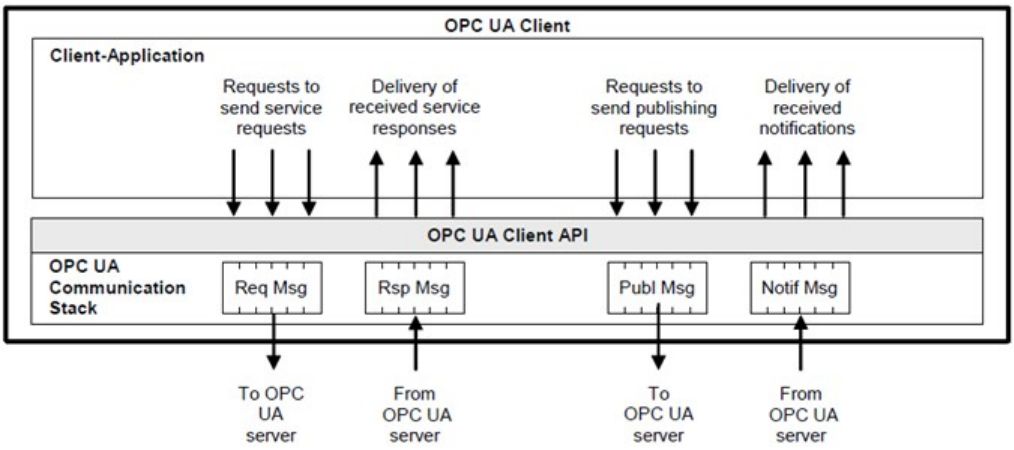

Fig. 2. OPC UA Client Structure[6]

Figure 2 pictures one simple OPC UA client containing client application, an internal API, isolating the application code from communication stack, and a communication stack that converts API calls into messages and delivers them to OPC UA server.

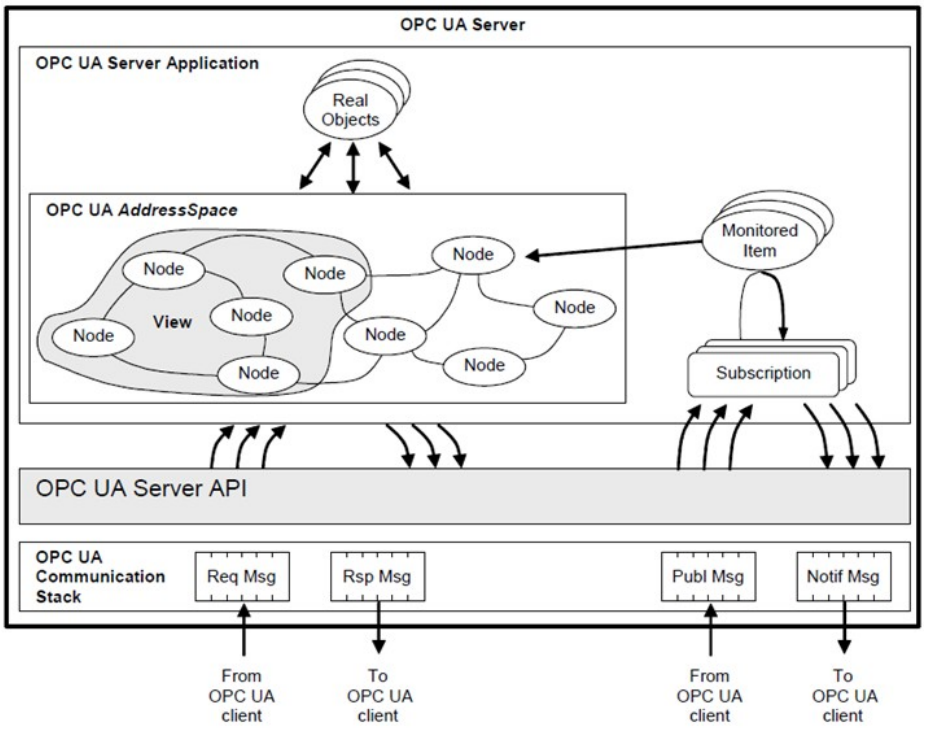

Fig. 3. OPC UA Server Structure[6]

In figure 3, one OPC UA Server structure is explained. As the aforementioned client structure, it also includes three main parts, server application, internal API and communication stack. It is worth mentioning that, real objects here are referred as physical field devices or software application that is only maintained internally. View, which is pictured as a part of address space, presents objects that can be browsed by clients

2.3 Secure Channel and Session

Since some data exchanged between client and server could be extreme precious and should be protected from other malicious third party, OPC UA defines a full set of security model, with which developer of system can configure the security level of the application to meet the need of reality. In the security model, authentication of client and server, authorization, integrity and confidentiality of

client-server- communication, auditability and availability of services are guaranteed. Also OPC UA provides a set of countermeasures against message flooding, eavesdropping, message spoofing message alteration, message reply, server profiling, session hijacking and so on[7].

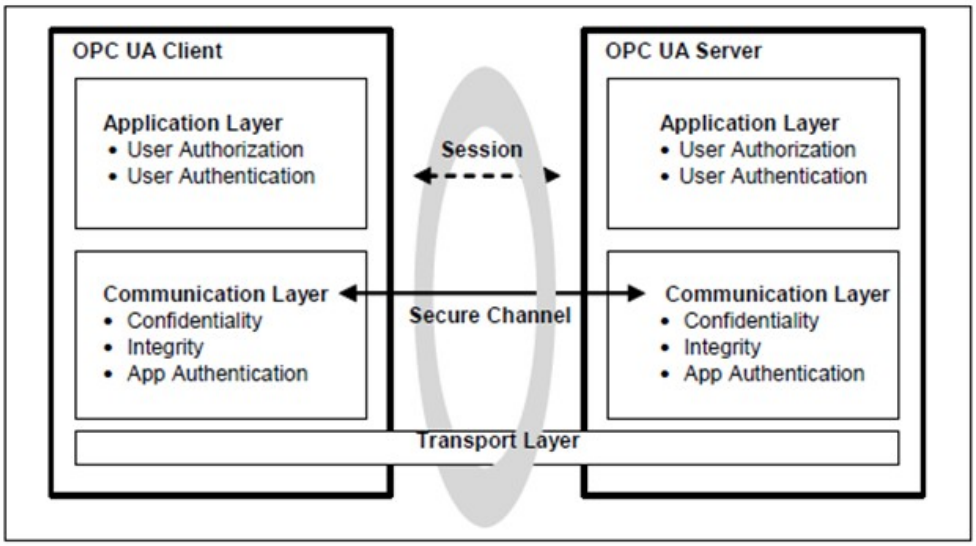

Fig. 4. OPC UA Client Server Communication[7]

Figure 4 pictures the typical security communication architecture of OPC UA. As shown in 4, the communication between OPC UA client and server is established above a secure channel, which is active during the whole application session and in this session, the state information, such as subscriptions from client, user credentials, is maintained. The secure channel is established only after successful validation of both client and server certificates and it provides necessary mechanisms to support confidentiality, message integrity and application authentication. On top of secure channel, is an application level session between OPC UA client and server, whose responsibilities are to transmit data information and commands. This session is also in charge of managing security policies like user authorization and authentication. It should be pointed out that, even a secure channel is out of work for some reasons, the session is still valid and OPC UA client and server involved in aforementioned session can still re-establish the broken secure channel. A secure transport layer is guaranteed by encryption and signatures methods provided by platform that supports OPC UA structure.

Security Handshake Security handshake as above explains with some details about how secure channel and session are established.OPC UA client initiates

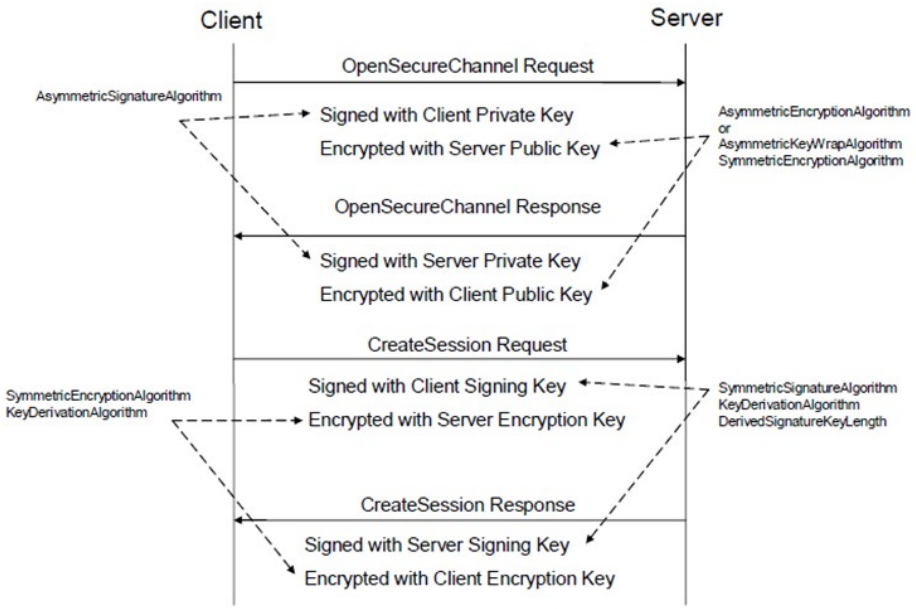

Fig. 5. OPC UA Client Server Security Handshake[7]

the first *OpenSecureChannel* request and waits the response from server. Messages exchanged during the process of construction secure channel between client and server are encrypted using asymmetric encryption and signature algorithms. But some security protocols that could be applied according to OPC UA standard, are not using an asymmetric message encryption algorithm to encrypt to request/response messages. Instead, they apply AsymmetricKeyWrapAlgorithm to encrypt symmetric keys and use symmetric encryption algorithm with encrypted keys to encrypt messages. After a successful construction of secure channel, OPC UA client sends *CreateSession* request and waits for server response. Messages transported during this procedure are encrypted with symmetric encryption algorithms and signed with client/server signing key.

2.4 OPC UA Communication stack

As discussed in subsection 2.3, the OPC UA communication stack is a three-layer architecture. Even the terminologies of those layers are defined as the ones used in ISO model, but layers in OPC UA are not directly equal to layers in ISO model. And figure 6 from OPC UA 6th specification[8] gives a precise overview of each layer in OPC UA communication stack model, meanwhile it demonstrates functionalities performed by each layer. Serialization layer will divide long message into pieces referred as message chunk, and after then secure

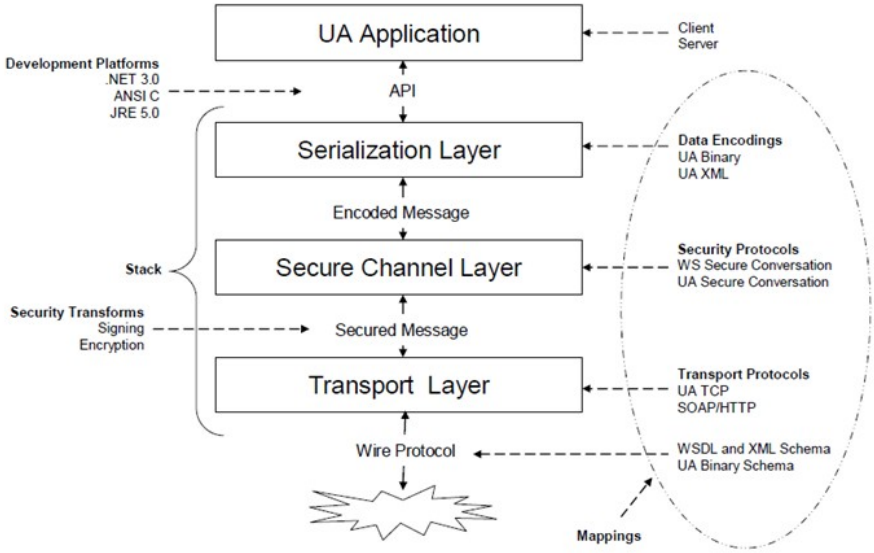

Fig. 6. OPC UA Client Server Communication Stack[7]

channel layer is going to encrypt each individual message chunk, not entire whole message. Likewise when receiving message chunk from others, OPC UA message receiver firstly verifies whether this message piece meets the security standard negotiated between OPC UA client and server. If not, this message receiver will close the secure channel. After a successful verification of all message chunks, the original OPC UA message will be reconstructed. Each secure message chunk applies the following structure.

In order to demonstrate, how OPC UA standards establish communication channel and build secure sessions, close message tunnel and reconstruc communication channel, it is assumed that the following demo OPC UA client and server set uses TCP/IP connection as transport layer protocol.

Establishment of communication channel As the first step to create TCP/IP connection, this process is always initialized by OPC UA client. One OPC UA client initiates his socket and sends *helloMessage*, that includes buffer size which is supported and suggested by message sender, to the target OPC UA server. After receiving greeting message, OPC UA server answers the request for establishing TCP/IP connection with acknowledge message and sends information about proposed buffer size to server's secure channel.

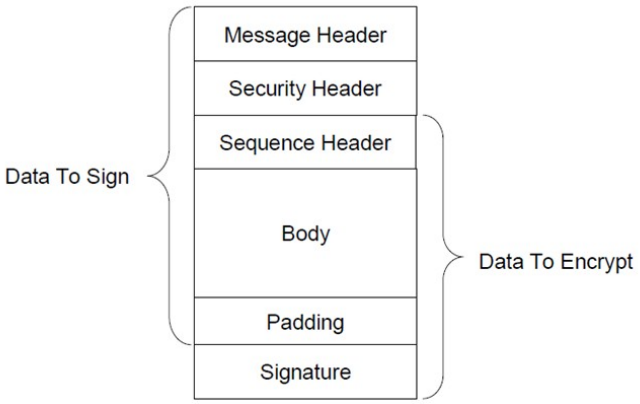

Fig. 7. Message Chunk Structure[7]

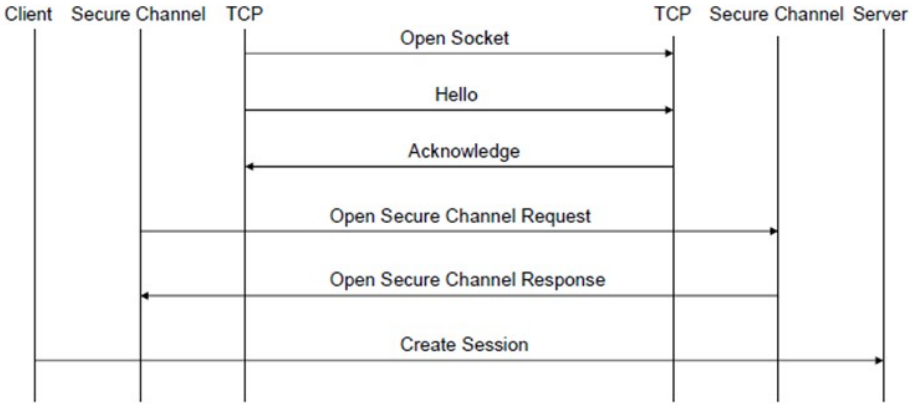

Fig. 8. Establish TCP/IP Connection[8]

Close TCP/IP Connection This process is done when OPC UA server receives *CloseSecureChannel* request from OPC UA client.

Recover Secure Channel Whenever error occurs during TCP/IP connection between OPC UA client and server, client will try to periodically re-establish it until the session is closed or the lifetime of security token goes to an end. Also it should be pointed out that the buffer size defined by corrupt secure channel should not be changed during this error recover process.

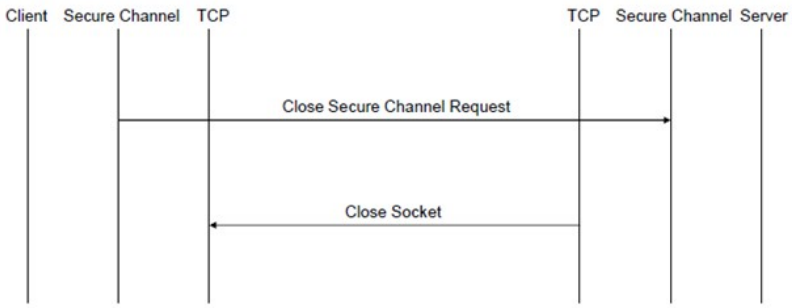

Fig. 9. Close TCP/IP Connection[8]

2.5 Historical Data

Last but not least security feature offered by OPC Unified Architecture is auditing, which supports traceability of any behaviours occur in OPC UA system. That means any security related problem can be recorded and for future use.

2.6 Other Competitor

WebSphere Message Broker Message Queuing Telemetry Transport (MQTT)[9] is another machine to machine (M2M) communication protocol. Compared with OPC UA standard, MQTT also supports UDP protocol in the transport layer. In OPC UA, only unidirectional, client to server, communication is provided, but in MQTT server to client communication is also possible without server implements client code. Moreover the communication overhead of MQTT is in comparison with OPC UA is relative small.

Even thought MQTT protocol supports communication environment with low bandwidth and high latency, OPC UA provides complex object model and supports more features, including historical data record, alarm, notification, complete security policies and this is reason why OPC UA is more suitable for the application scenario that handles sensitive data with complex structure and needs immediate response.

Another member from Internet of Things is Constrained Application Protocol (CoAP)[10] which is designed for the extreme simple electronic devices with less memory and computing power and original CoAP only runs over UDP. Compared with OPC UA, simplicity from CoAP is the advantage, but apparently it should be considered that in the implementation scenario other transport protocol could be used, like TCP, more functions and services other than pure message exchange between client and server, are requested from users.

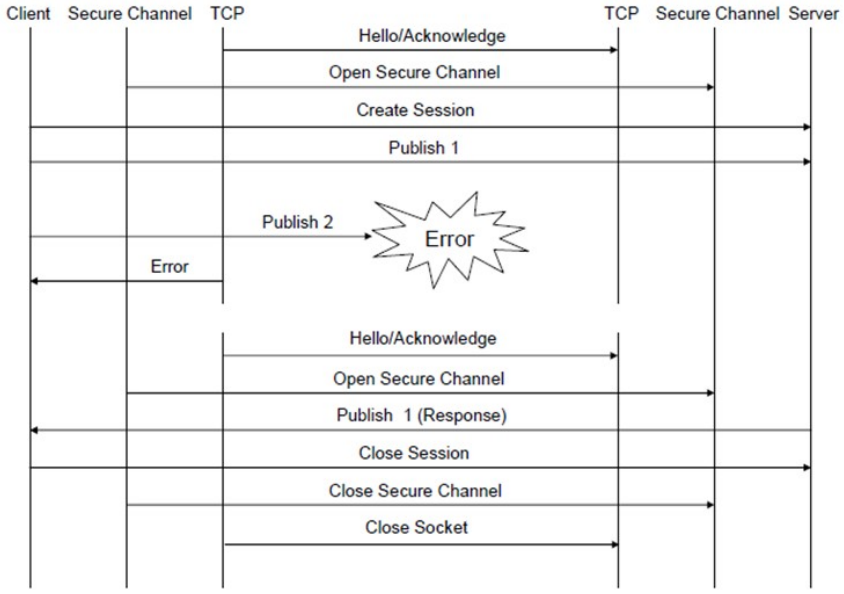

Fig. 10. Recover Secure Channel[8]

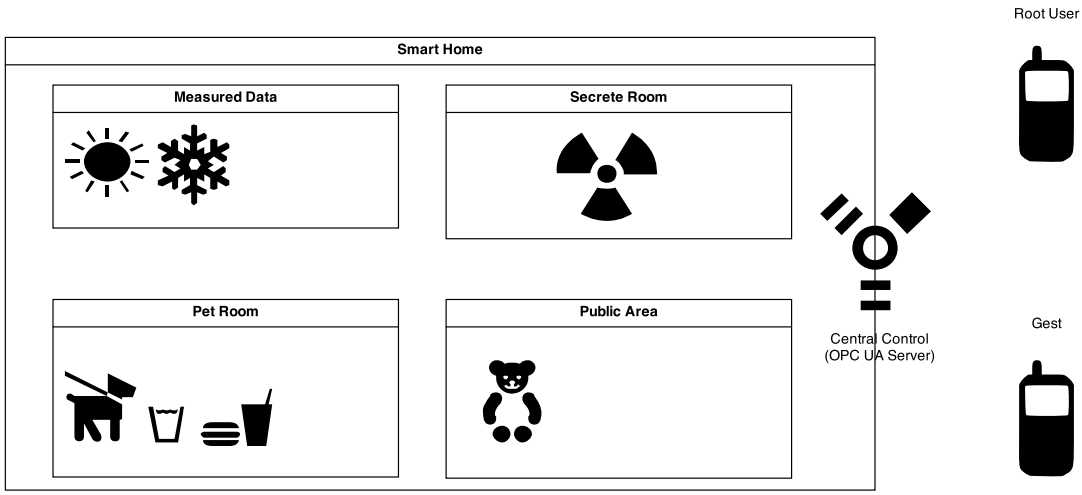

Fig. 11. Smart Home

3 Implementation Scenario

Figure 11 describes the basic structure and functionalities of Smart Home. Central Controller also named as OPC UA Server is in charge of monitoring prede-

fined environment variables, for instance, home temperature, luminance and how much water the pet has, and taking corresponding behaviours, such as opening windows, turning off the heating or notifying pet owner that puppy needs water. With the help of such services a more comfortable living condition is created in an automated way. Also OPC UA Server controls access right of entering each room, which means only authenticated users with enough authority can open the door. Moreover the root user, namely the owner of this house, is capable of assigning the permission of accessing particular room to other guests. In case of when he/she is taking a vocation and pet cannot get necessary care while he/she is away. At the same time, root user wont worry about the person, who promises feeding pet, entries the forbidden room. At last, every single action taken by the OPC UA Server is recorded and is available for future use.

In the implementation scenario, both OPC UA clients are Universal Integrated Circuit Card (UICC) based handy user and OPC UA server is a computer with UICC card reader. Environment date is measured periodically by sensors and each room is locked, only the client, who is authenticated by server and holds enough authority can enter. Figure 12, describes possible software struc-

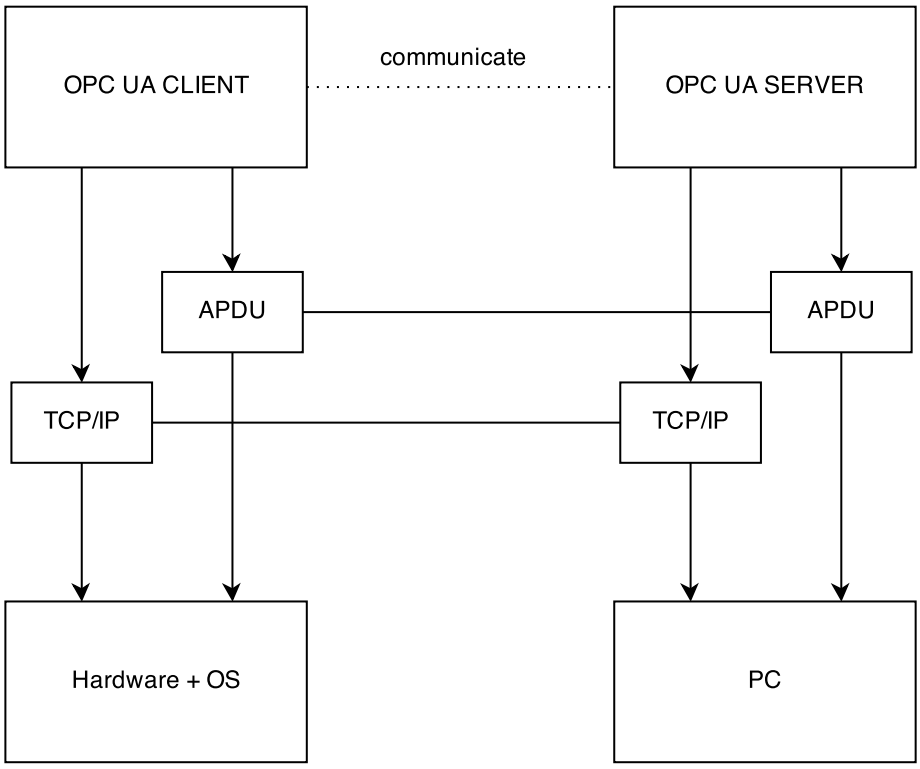

Fig. 12. OPC UA Client Server Based On TCP/IP or APDU

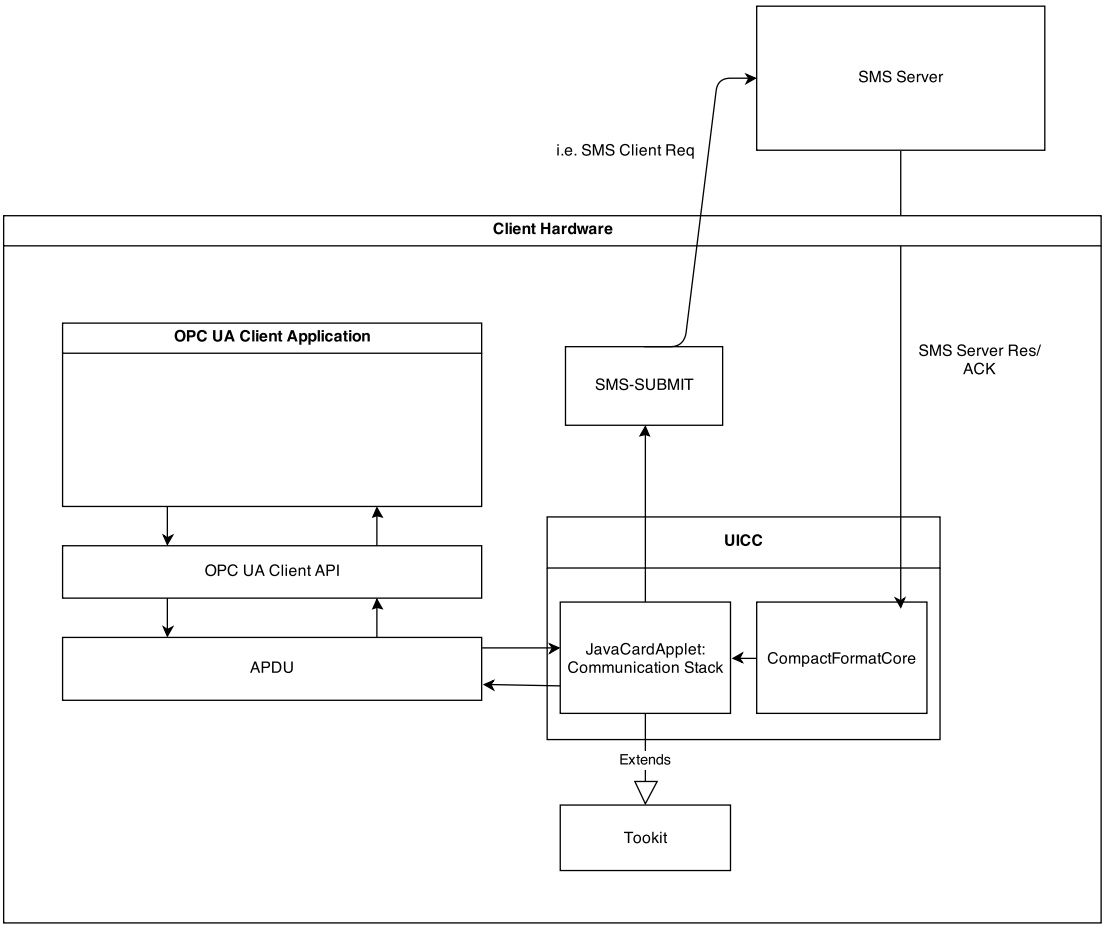

Fig. 13. Client Structure Using APDU

ture of aforementioned OPC UA client server structure. With different chip card, OPC UA client application is able to communicate with server using Application Protocol Data Unit(APDU) and Short Message Service(SMS), which is a more nature and traditional way to exchange date with chip card, or TCP/IP based web service when newly released Javacard 3 is applied.

3.1 Client Server Structure

Based on the abovementioned application scenario, the basic architecture of OPC UA client is pictured as figure 13 when information exchanged between OPC UA client/server application and chip card is in APDU format and messages transmitted between client and server are encoded as SMS. Under aforementioned assumption, OPC UA client is a smart phone with UICC card and this smart

card is in charge of user authentication and processing secure messaging between card application and chip card pair. Thanks to its self-containment structure, smart card itself does not dependent on other external resources, which could be extreme vulnerable to potential secure attack, and therefore provided a set of secure mechanisms, like for instance, secure messaging, hardware security and OS security. OPC UA client Application is a phone app. An OPC UA client API connects application code and OPC UA communication stack, which is a Javacardapplet, that is embedded on UICC chip card. The message exchanged between internal API and communication stack is in form of APDU. After receiving APDU from communication stack, UICC card will generate SMS submit and send this SMS to SMS server. SMS server will then forward this SMS to the OPC UA server. Messages, such as OPC UA Server responds, notification information corresponding to client subscriptions and OPC UA Server Acknowledgements are sent by SMS server to OPC UA client. Upon receiving message from SMS server, OPC UA client firstly processes this SMS message with the help of CompacFormatCore component integrated in UICC card and then forwards this message to OPC UA communication stack. With the help of communication stack, message from server is translated into APDU and sent to OPC UA client API, which eventually translates and passes it to OPC UA client application code. As OPC UA server is a gateway computer with UICC card reader used.

3.2 Other Candidates

Of course SMS based communication is not the only option, according to the newly developed JavaCard 3.0 technology[11], now it is possible to integrate and deploy servlet applications on smart card devices, which means direct TCP/IP connection with chip card and web services could be realized. Moreover according to SIMaliance LTE UICC profile[12], in Long Term Evolution(LTE) standard, which is also marked as 4G LTE, chip card is capable of acting as HTTP client, when there exits an OTA(Over The Air) HTTP Server.

3.3 Smart Card Security

As explained, smart cards are widely used in applications that require strong protection. With sophisticated communication protocol using Application Protocol Data Units, smart card and Card Accepting Device(CAD) are able to process secure message exchange and bidirectional authentication. Moreover sensitive data like certificates, encryption keys are stored on card along with other precious user information and this data is extreme difficult to be altered by third party .Most of time smart card also acts as secure token and can process cryptographic algorithms on hardware. Nowadays, smart card supports symmetric key algorithms like DES, triple DES; standard public key cryptography for instance RSA, hash functions such as commonly SHA-1[13]. More powerful microprocessor on chip card is, the speed performance is better.

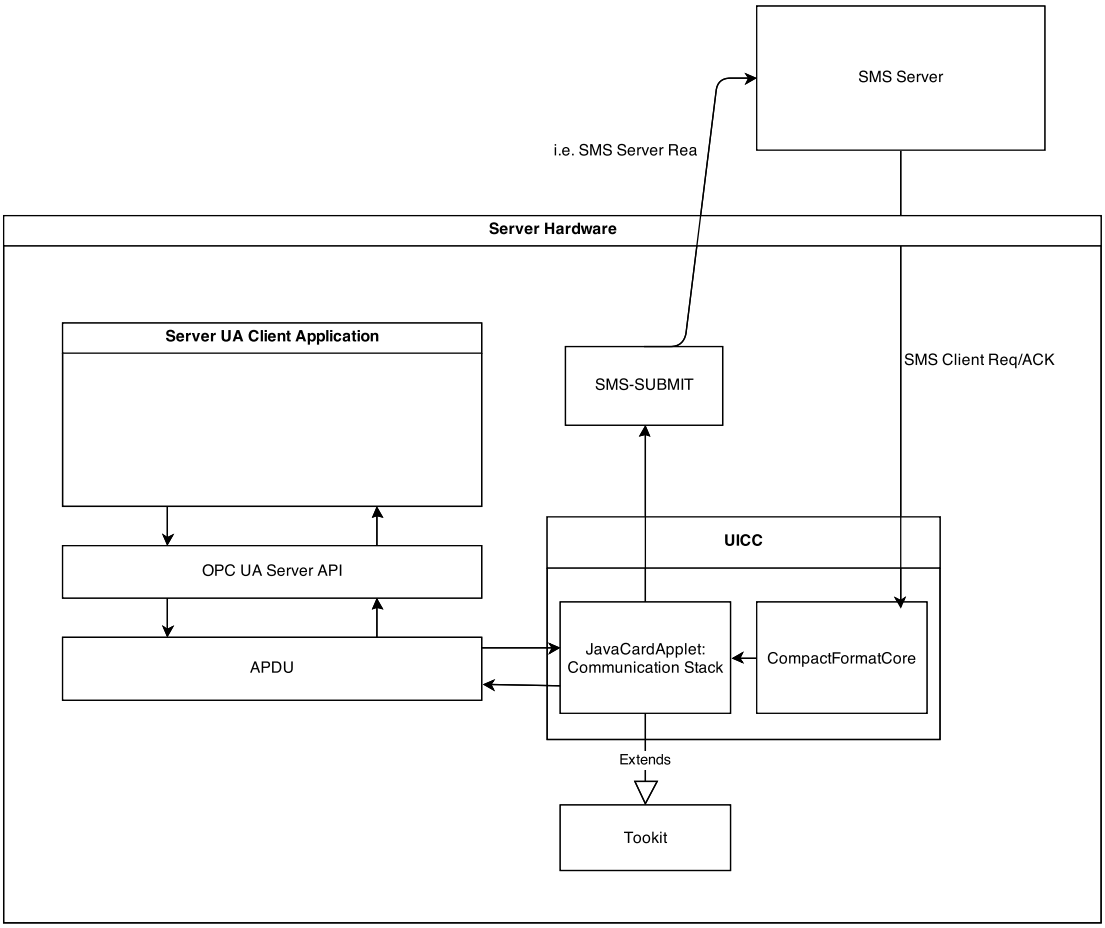

Fig. 14. Server Structure Using APDU

4 Time Lines

- read paper, documentation, reference
- analyse and design communication stack that fits UICC card and meets OPC UA standard
- design proto client/server structure
- design models meeting OPC UA standards
- analyse and design secure protocols.
- design prototype
- programming
- debugging/performance analyse

References

1. Stefan-Helmut Leitner and Wolfgang Mahnk: Opc ua-service-oriented architecture for industrial applications
2. Wikipedia contributors: OPC Unified Architecture. `http://en.wikipedia.org/wiki/OPC_Unified_Architecture` [update;Februray 5,2014].
3. Mariusz Postol (CAS), Voytek J. Janisz (ABB), Claude M Lafond (ABB), Jim Luth (OPC Foundation): Implementation of the opc ua information model for analyzers
4. OPC Foundation: Opc unified architecture specification part3 address space model 1.01. (February 6.2009)
5. OPC Foundation: Opc unified architecture specification part4 services 1.01. (February 6.2009)
6. OPC Foundation: Opc unified architecture specification part1 overview and concepts 1.02. (July 10.2012)
7. OPC Foundation: Opc unified architecture specification part2 security model 1.01. (February 6.2009)
8. OPC Foundation: Opc unified architecture specification part6 mappings 1.01. (February 6.2009)
9. SID: Mqtt vs opc-ua: Das http der industrie 4.0? `http://dennisseidel.de/the-http-for-industrie-4-0-mqtt-vs-opc-ua-german` [update;2013].
10. Wikipedia contributors: Constrained Application Protocol. `http://en.wikipedia.org/wiki/Constrained_Application_Protocol` [update;Februray 8,2014].
11. Bruce Hopkins: Deploying Servlets on Smart Cards: Portable Web Servers with Java Card 3.0. `http://www.oracle.com/technetwork/articles/javase/javacard-servlets-136657.html` [update;January 2010].
12. SIMalliance: Simalliance lte uicc profile v1.0
13. Wolfgang Rankl und Wolfgang Effing: Handbuch der chipkarten - 5. deutsche auflage. (2008)