# 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, known 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 developing a common communication interface for devices which participate in automation system.Meanwhile, the technology of smart card is widely used in information security fields of finance, communication, personal and government identification, payment. Therefore it is meaningful and promising to develop OPC UA standard satisfying application on embedded smart card secure device, for the purpose of secure remote control, enterprise resource planning and etc. Since the storage and compute capacity of chip card is limited, OPC UA product will consist of two essential parts, namely client/server application code, realized as Android or other application, and communication stack, realized as Javacard Applet based on Remote Application Management from GlobalPlatform. The implemented demonstration scenarios and corresponding analysis show the possibility of developing OPC UA standard satisfying application on embedded devices with smart card to benefit customers.

## 1 Introduction and Motivation

According to the *Mobile Economy 2013* from *Global System for Mobile Communications Association*, at the end of year 2013 there are over 3.2 billion mobile subscribers in total, which means one half the population of the earth now enjoy the social and economic convenience brought by mobile technology. Moreover by year 2017 700 million new subscribers are expected to be added. And the number of mobile subscriber will reach 4 billion in 2018. Mobil technology opens nowadays a promising market.

Mobile products play an irreplaceable role at the heart of our daily life. With the help of mobile technology, the user's world in many domains such as, education, financial transactions, health and etc. are inter-connected. Mobile users are enjoying the advantages of mobility. Services, like 24/7 monitored home security, full control about the management of home humidity and temperature, exist not only in science fiction film but also could be realized by today's technology.

At the same time, mobility in industry and business world is also a critical assert, which can not only increase efficiency and productivity but also drive new revenue generation and competitive advantage. The most convicting example here is Machine to Machine communication, that is also referred as M2M technology. In M2M communication, machines which are usually embedded with smart cards exchange gathered date with each other to accomplish common task using wireless or wired networks. M2M technology is widely employed in different industry spheres such as factory automation, remote access control and sensor monitoring. It boosts the efficiency of corresponding processes, offers centralized service support and date management, minimizes system response time.

But in order to enjoy the aforementioned features, two tough issues must be resolved. First, how to achieve a common interface for the devices that participate in the system. And second how to guarantee system security under different communication environments with various date complexity.

### 1.1 Solution Idea

In this master thesis, I am going to address solutions for questions mentioned in section *Introduction and Motivation* and design a smart home system for the purpose of demonstration. In this smart home system, home owner using smart phone is capable of experiencing 24/7 home security service, remotely managing inner home environment parameters and assigning access permissions. This system consists of smart phones with Universal Integrated Circuit Cards (UICC smart card), digital door locks, control devices, environment sensors and if necessary a central control computer. Moreover each device is equipped with smart card, which acts not only as secure token, that saves user credentials, but also is in charge of construction and management of the devices' communication.

In particular, I will introduce the newly released industry automation standards object linking and embedding for processes control unified architecture(OPC UA standards) to build a common communication interface for devices that are mentioned above and design communication stack for OPC UA standards on UICC smart card., whose duties are: creation and management communication between OPC client/server application, message serialization and secure message exchange.

### 1.2 Paper Structure

At first, in the second chapter I will present the fundamental technologies which will be frequently mentioned in this paper. Secondly the state of art, for instances mobile security, home remote control technologies, Remote Application Management from GlobalPlatform will be introduced, which act together as cornerstone for my implementation scenario. In the fourth section, I am going to focus on UICC mobile security and base on UICC framework build a OPC UA standards scarifying communication stack as Javacard Applet with the help of GlobalPlatform specifications. In the next implementation chapter, I will present how my demonstration scenario can be realized. As sixth and seventh chapter,

text results and performance analysis are preformed to show the reliability and security of OPC UA standard based UICC card embedded system.

## 2 Fundamental Technologies

In this section, i am going to give a brief introduction about technologies and terminologies that are applied in this paper.

### 2.1 OPC Unified Architecture Structure Overview

Object Linking and Embedding for Process Control Unified Architecture, known as OPC UA is the most recent released industry standards from OPC Foundation, acts nowadays as the most promising candidate in industry M2M automation world, whose major duty is the build a secure communication interface for machines that participate in aforementioned system. In the following subsections I am going to give a quick ABC guide about the features offered by OPC UA standards.

**OPC UA Specifications** 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.

**OPC UA Client Server Structure** OPC UA standards apply the classic client server architecture, where server is in charge of managing functionalities and data information from a particular machine device, for instance temperature data from a remove allocated sensor. Meanwhile client possesses the ability to query information from server and to let sever perform predefined operations.

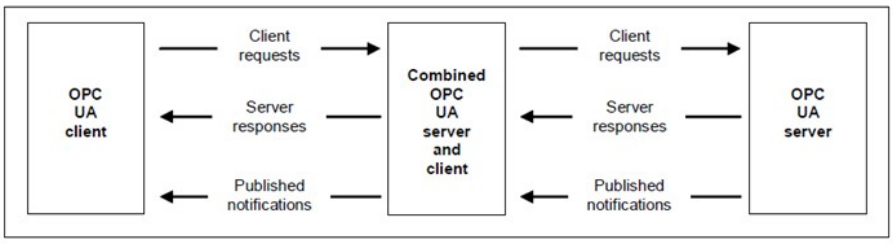


**Fig. 1.** OPC UA Client Server Structure[1]

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

**OPC UA Terminologies** In OPC Unified Architecture on server stored information that can be visited by clients is defined as *address space*[2] and there also exits a set of services[3] which are provided by server and are introduced in order to apply operations on *address space*. Information in address space is organized as a set of in particular hierarchy structured *Objects*. *Object* here could refer to data value gathered by particular sensor or server system parameters and etc. 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 and not only limited to simple single hierarchy.

One of the charming features provided by OPC UA is *Event Notifications*. With the help of *Event Notification*, OPC UA servers are allowed immediately after some conditions, which is normally set by a client, are satisfied to publish data and send this data to that particular client. 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.

**Standard OPC UA Server** In figure 2, the structure of one standard OPC UA Server is described. It includes three main parts, server application, internal API and communication stack. In server application part, functionalists which are offered by OPC UA standards are realized, such as *Event Notification* and other basic server services like, processing request from connected client, data encryption and decryption and subscription process. Moreover, *Real objects* managed by server here are referred as physical field devices or software application that is only maintained internally. *Node* in *Address Space* can be understood as *Real Ojbects* that are related with each other according to one specific hierarchy. *View*, which is pictured as a part of address space, presents objects that can be browsed by particular clients. The main task for communication stack is to establish secure communication session based on secure channel between OPC UA client and server. Typically communication messages which are exchanged frequently among clients and servers are, request-, notification- message from client and corresponding response-, publish message from server. At last, an internal API connection the server application and the communication stack.

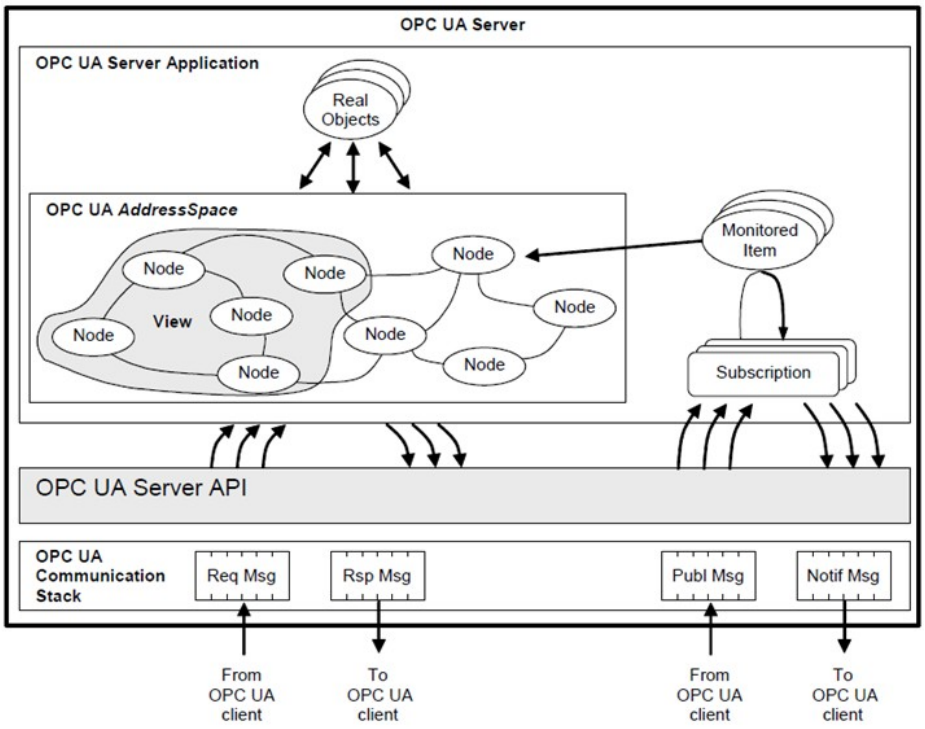


**Fig. 2.** OPC UA Server Structure[1]

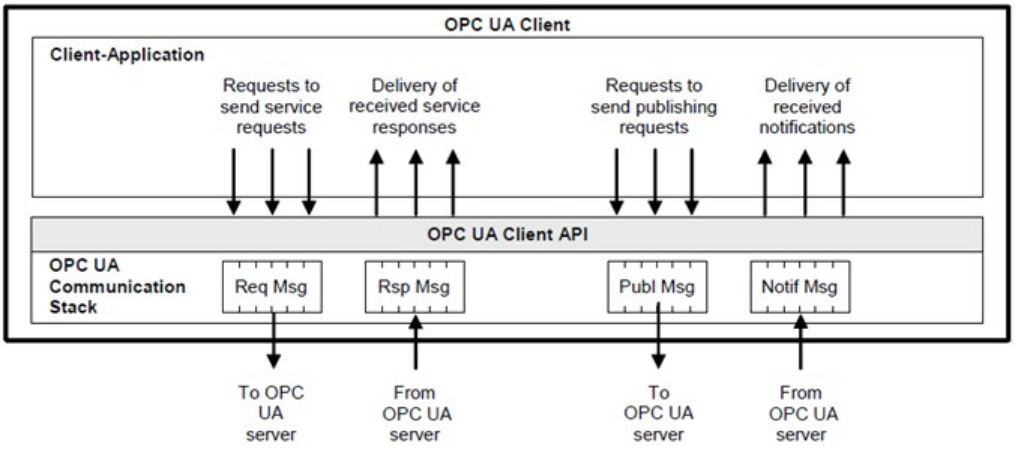


**Fig. 3.** OPC UA Client Structure[1]

**Standard OPC UA Client** Figure 3 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.

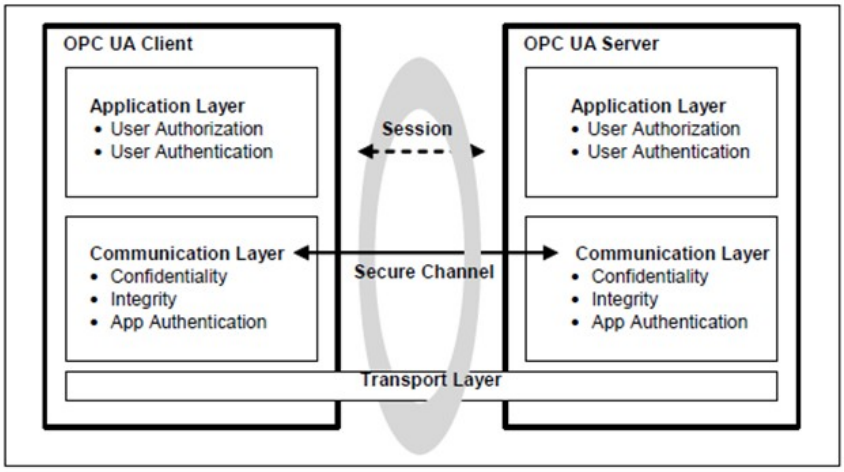


**Fig. 4.** OPC UA Client Server Communication[4]

**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[4].

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.

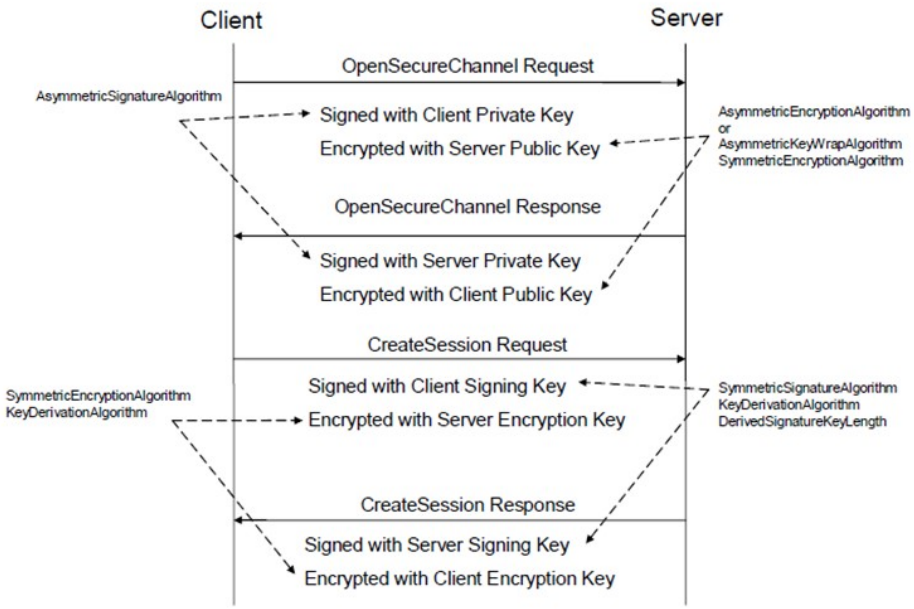


**Fig. 5.** OPC UA Client Server Security Handshake[4]

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

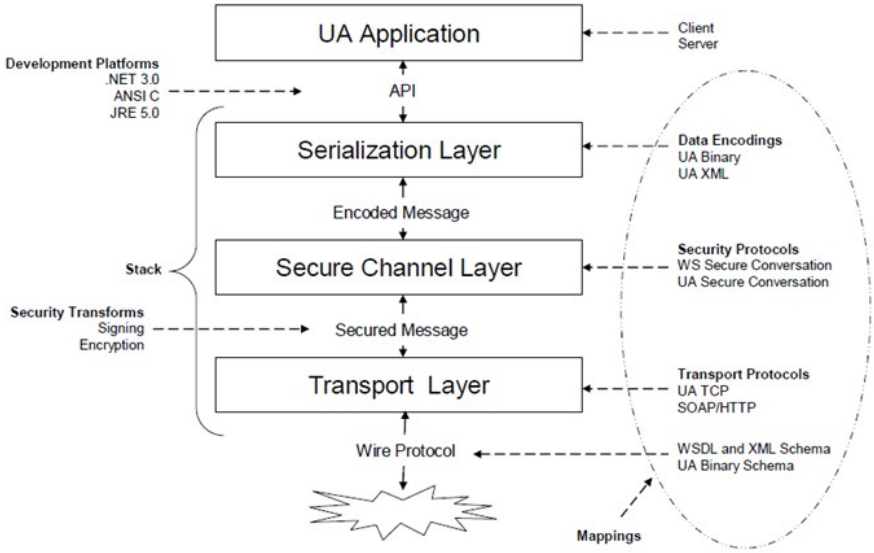


**Fig. 6.** OPC UA Client Server Communication Stack[4]

**OPC UA Communication stack** As described above, the OPC UA communication stack is a three-layer architecture: application layer, communication layer and transport layer. 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. Figure 6 from OPC UA 6th specification[5] gives a precise overview of each layer in OPC UA communication stack model, meanwhile it demonstrates functionalities performed by each layer.

UA Application layer realizes client and server code. Serialization layer together with secure channel layer build the communication layer and their job is dividing long message into pieces referred as message chunk, encrypting each individual message chunk, not entire whole message and forwarding encrypted message chunk to transport layer. 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 and sent to UA Application Code through API. Each secure message chunk applies the following structure described in figure 7.

Knowing the essential parts of OPC UA communication stack, in the following subsections the establishment, re-establishment and close of communication channel are explained.

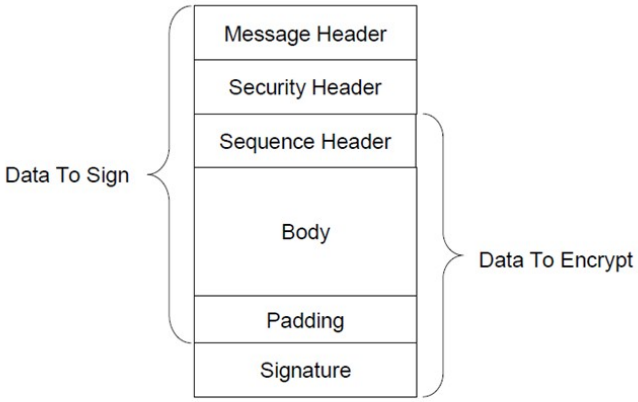


**Fig. 7.** Message Chunk Structure[4]

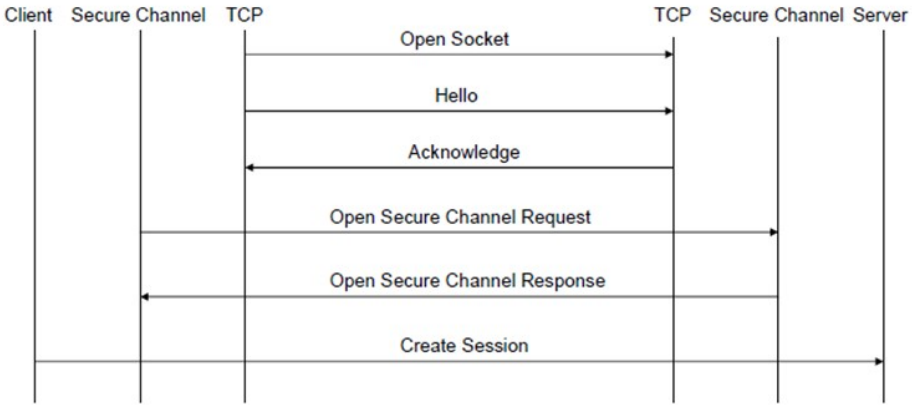


**Fig. 8.** Establish TCP/IP Connection[5]

**Establishment of communication channel** As the first step to create TCP/IP connection, this process is always initialized by OPC UA client. OPC UA client initiates his socket and sends *helloMessage*, that includes supported buffer size which specifies the message chunk size used for future communication, 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 reports negotiated buffer size to his own secure channel layer. Moreover during the creation of communication channel process, the greeting *hallo* and answering *acknowledgment* messages could only be sent once. If OPC UA client or server receives them more than one, error will be reported and corresponding communication socket will be closed. Even thought server application code does not have to work during the negotiation of secure channel process, it should provide the communication stack all his trusted certificates which help communication stack to verify the identity of the other communication partner.

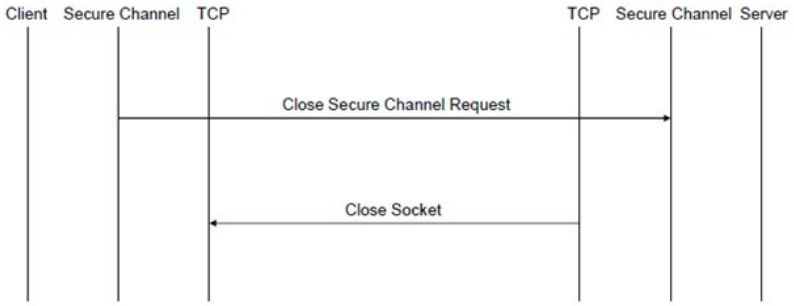


**Fig. 9.** Close TCP/IP Connection[5]

**Close TCP/IP Connection** This process is done when OPC UA server receives *CloseSecureChannel* request from OPC UA client. During this process, server releases all the resource taken by corresponding secure channel and sends none response.

**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.

**Historical Data** Last but not least security feature offered by OPC Unified Architecture is auditing, which supports traceability of any behaviours occur in

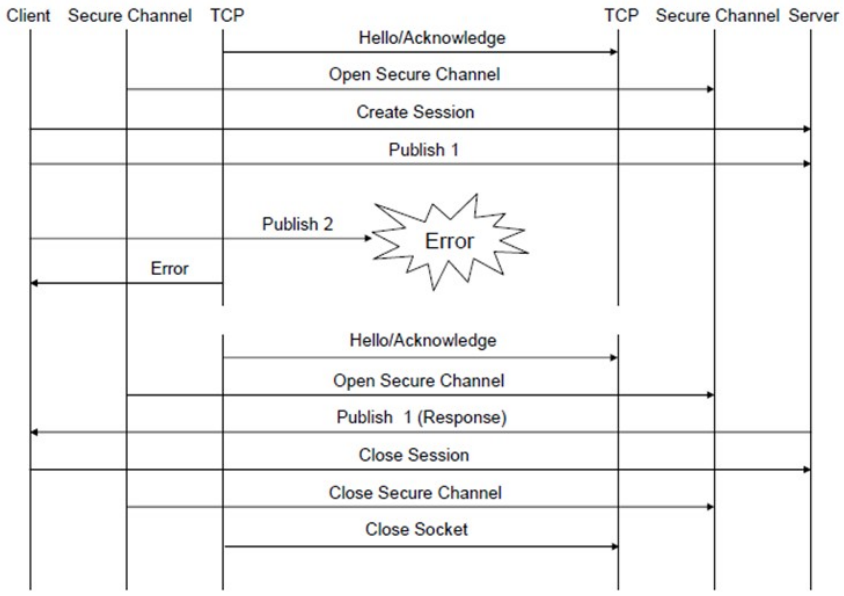


**Fig. 10.** Recover Secure Channel[5]

OPC UA system. That means any security related problem can be recorded and for future use.

**Other Competitor** WebSphere Message Broker Message Queuing Telemetry Transport (MQTT)[6] 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)[7] 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 proto-

col could be used, like TCP, more functions and services other than pure message exchange between client and server, are requested from users.

## 2.2 UICC

The Universal Integrated Circuit Card is the smart card used in mobile terminals in GSM and UMTS networks. It enablse authenticated subscriber to join the network with their mobile terminals and at the same time protects the security as well as ensure the integrity of essential user data. UICC acts also most time as the secure token, which stores and protects subscriber's confidential information. Moreover, as a 32bit processor, UICC is also capable of processing necessary encryption and decryption algorithms[8].

**Smart Card** Smart Card, whose characteristic feature is an integrated circuit that is embedded in a chip card, which is capable of performing data process, information storage and message transmitting[9]. The most charming feature of smart card is that, sensitive user's credential information such as certificates, encryption keys along with other precious user information can only be accessed though a serial interface, which stands under the strict control of the card operation system. This characteristic provides strong protection against unauthorized data access and ensure confidentiality of on card stored information. Therefor smart cards are widely used in applications that require strong protection.

With sophisticated communication protocol using Application Protocol Data Units(APDU), smart card and Card Accepting Device(CAD) are able to process secure message exchange. And 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[9]. More powerful microprocessor on chip card is, the speed performance is better.

In ISO/IEC 7816 standards family, the smart card's fundamental properties and functionalities are defined.

**Smart Card Software Components** As illustrated in figure 11, one typical smart card software includes Card Operation System, native services such as I/O operation and memory management, Java Card Runtime Environment(JCRE) which consists of Java card Virtual Machine and Framework that is in charge of dispatching APDU, applet management, and at last other optional industry specific extensions[10]

**Message Exchange with APDU** Application Protocol Data Unit, or APDU for short, is used to perform data exchange between smart card and CAD. There are two categories of APDU, namely command APDU and response APDU. Command APDU structure and response APDU structure are described in Table 1 and Table 2 respectively[9].

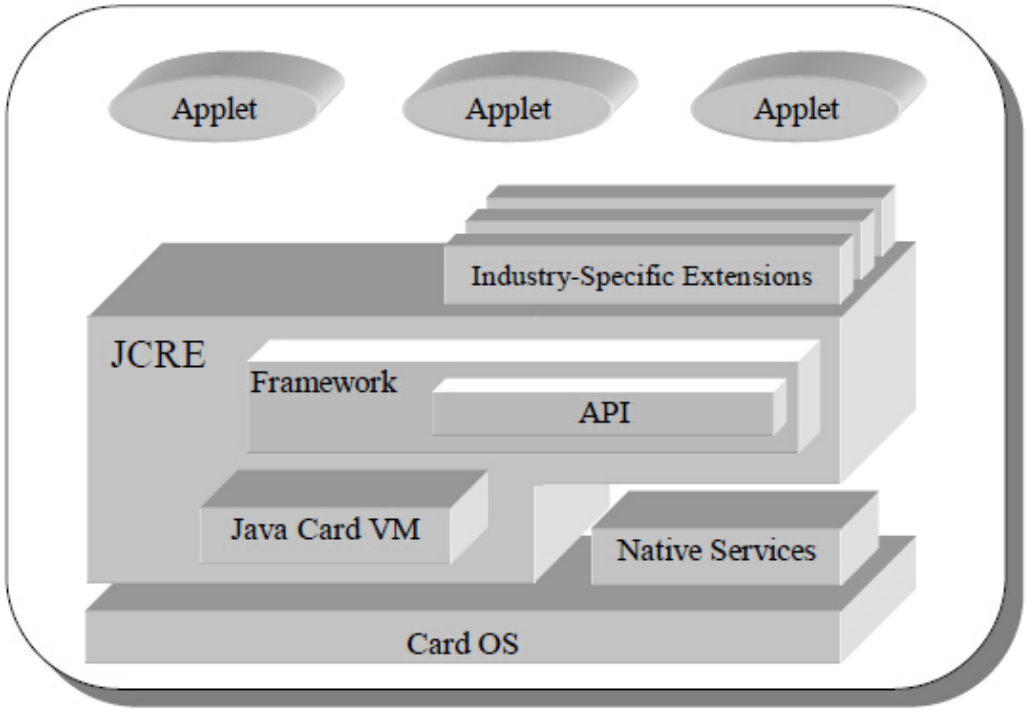


**Fig. 11.** Smart Card Software Components[10]

**Table 1.** Command APDU Structure

| | | |
|---|---|---|
| CLA | class byte identifying application | mandatory |
| INS | instruction byte representing the actual command | mandatory |
| P1 | parameter 1 used to provide more command information | mandatory |
| P2 | parameter 2 used to provide more command information | mandatory |
| Lc field | the length of data received by card | mandatory |
| data field | data sent to card | optional |
| Le field | the length of data sent by card | |

**Table 2.** Command APDU Structure

| | | |
|---|---|---|
| data field length | decided by Le of preceding command APDU | mandatory |
| SW1 | state word 1 also called return code 1 | mandatory |
| SW2 | state word 2 also called return code 2 | mandatory |

**Secure Messaging** Since all communication between smart card and terminal is based on digital electrical pulse performed on card I/O line, attacker can easily record all communication information and and try to recover sensitive use credential data. Therefore secure messaging mechanism is proposed and used to protect against aforementioned message eavesdropping, to grantee authenticity and confidentiality of exchanged information.

In secure message mechanism, both message sender and receiver muss have decided common used cryptographic algorithms and corresponding pre-shared keys. Especially in telecommunication domain in accordance with ISO/IEC 7816-4[9], TLV(Type-lengthl-valuse)-formed data, which encapsulates relative user information, is used to ensure secure messaging.

### 2.3 GlobalPlatform and Remote Application Management

GlobalPlatform is an international non-profit organization that provides standardized specifications for multiple smart card applications. It is now widely accepted and used as industry standard for managing Java Applet based application on Javacard Operation System in several domains, such as in communication industries and payment company[11].

As shown in figure 12, the GlobalPlatform card architecture contains for essential parts. The runtime environment, that provides hardware-neutral API for card application and manages card memory spaces. The on card installed applications, which offers customers various functionalities and services. The security domain, which is usually associated with particular application and knowns as on-card representatives of off-card authorities. And at last, OPEN framework[11].

**OPEN - GlobalPlatform Environment** OPEN provides API to applications, performs APDU dispatching job and also is in charge of application selection as well as logical channel management[11]. Logical channel is designed to enable the data exchange between multi application and terminal. Each opened logical channel will handle message regard of one particular application. Moreover one special logic channel named basic channel is always opened. In order to ensure card security, OPEN is designed to support mechanisms for data integrity, confidential , authentication and resource availability.

**Secure Channel Protocol** In particular, GlobalPlatform has designed a set of secure mechanisms knows secure channel protocol to guarantee aforementioned

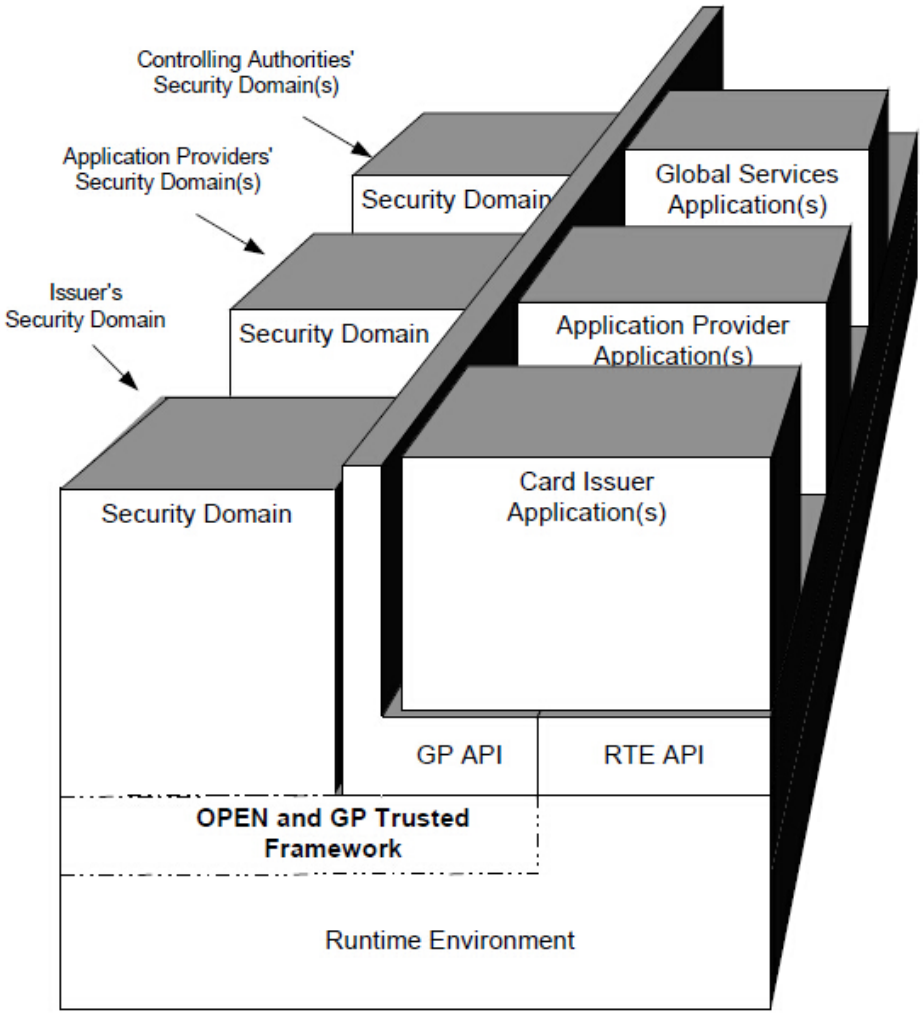


**Fig. 12.** Card Operation System Architecture[11]

secure messaging. It grantees confidentiality of exchange information and offers integrity check and authentication services. Secure Channel Protocol also applies a cryptographic exchange process to let smart card and off-card entities to perform entity authentication with each other.

### 2.4 Java Card

Java Card technology not only adopts the distinguishing features from Java, such as productivity, security and portability[10], it also makes Java technology available on smart card, where programmer must be faced with more harsh conditions, for instance limited memory resource.

But in contrast with Java VM, Java Card VM is actually divided into two parts, the on-card part, which is in charge of bytecode execution, class as well as object management and secure data exchange, the off-card part, which is the real Java Converter. As illustrated in figure 13, a complied Java applet (.cap

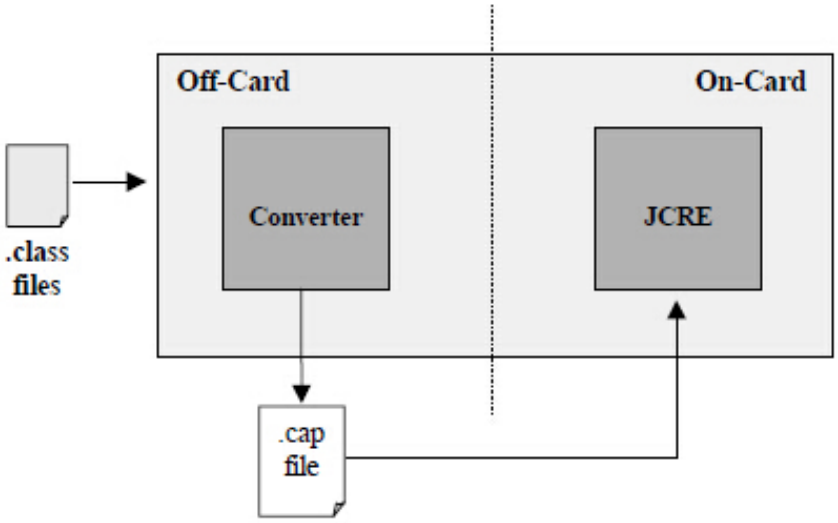


**Fig. 13.** Java Card VM Components[10]

file) is generated by off-card Converter based on corresponding Java class file and executed by on-card Java Card Runtime Environment.

**Language Specification** Apart from above mentioned Java Card VM, compared with original Java language, Java card has many unique features. Since current smart card does not support multitasking, therefore threads are not

backed up in Java Card. Also garbage collection is performed by VM, as a result function *finalize()* is not supported. Moreover as smart card memory space and process ability is limited, programmer can only use three main primitive types, namely byte, short and boolean. Furthermore only one-dimensional array is offered. Nonetheless Java card language supports all features of inheritance and all Java language security, for example, private access modifiers as well as bytecode verification[10].

**Transaction Integrity** One of the most import features of Java Card technology is that Java Card Runtime Environment ensures the integrity of transaction, which means even a unexpected loss of power on smart card, the ongoing transactions' integrity is also protected using following schema[9]:

```
// Transaction Starts
JCSystem.beginTranscation();

doSomething

//Transaction Ends
JCSystem.commitTransaction();
```

Only when JCRE finishes method *JCSystem.commitTranscation()*, this corresponding transaction is finished and submitted. Otherwise JCRE will throw transaction exception and reset data that involved in this broken transaction.

#### Java Applet

### 2.5 Cryptography Foundations

### 2.6 Android

## 3 State of Art

## 4 Implementation Scenario

Figure 14 describes the basic structure and functionalities of Smart Home. Central Controller also named as OPC UA Server is in charge of monitoring predefined environment variables, for instance, home temperature, luminance and how much water the pet has, and taking corresponding behaviors, 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

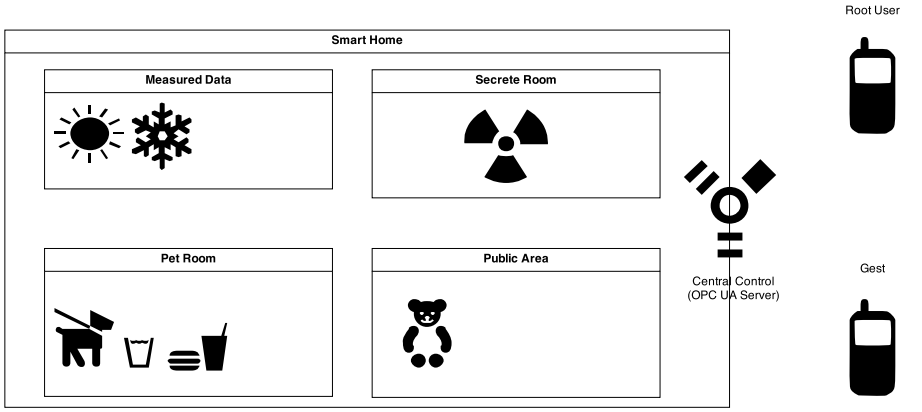


**Fig. 14.** Smart Home

when he/she is taking a vocation and pet cannot get necessary care while he/she is away. At the same time, root user won't 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, OPC UA clients are Universal Integrated Circuit Card (UICC) based phone user and in home allocated senor. OPC UA server is a secure hard device, that is in charge of communication management with phone use and implementation of OPC UA server application code. 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. It is assumed that handy users are in an open environment at mean while senors in home have a relative secure wifi connection with OPC UA server. Smart card that is applied in this scenario also acts as security token for OPC UA client, which contains credential information like encryption keys. certificates and digital signature. Moreover the communication stack is developed and integrated on smart card, which means without corresponding UICC card, OPC UA client and server are not able to appropriately finish their work.

Figure 15, describes possible software structure 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 components from Global Platform[1] is applied or newly released Javacard 3 is used.

[1] Global Platform is a cross industry, non-profit organization that develops and publishes standard in secure chip technology.

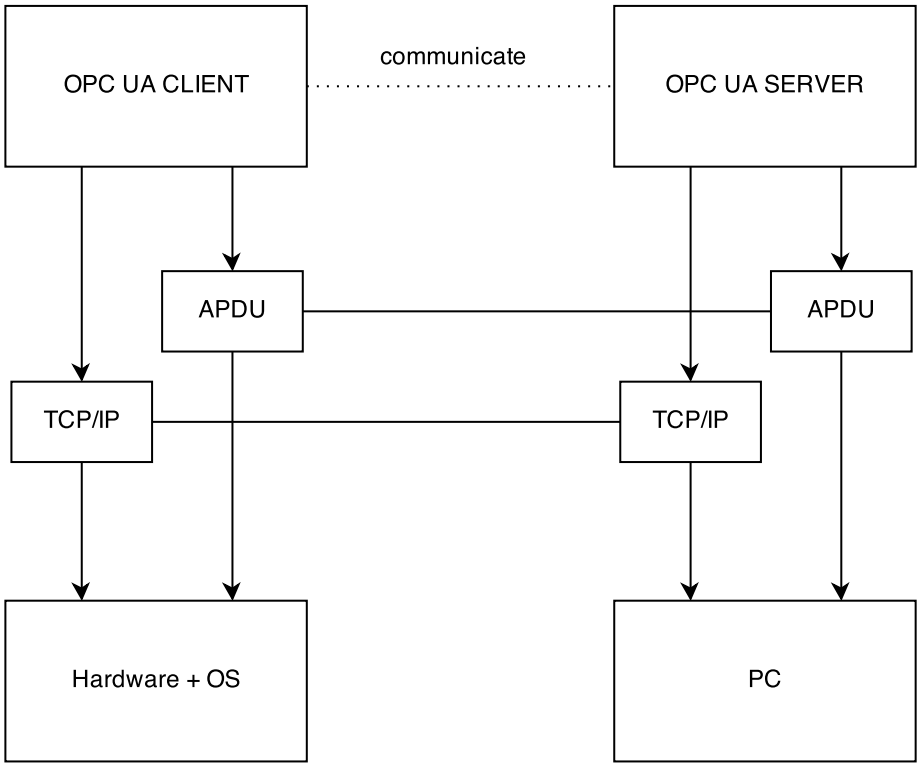


**Fig. 15.** OPC UA Client Server Based On TCP/IP or APDU

### 4.1 Software Structure

According to the above-mentioned application scenario, UA application client and server code will be written using Java, deployed as Handy application, OTA server respectively. Basic functions as following are provided:

- subscription/publishing environment data
- secure message exchange
- authority management
- historical data management
- running client's command

Communication stack is develop on UICC smart card realizing secure channel and session management, transporting data to message chunk receiver using Tcp/IP connections. An internal API translates OPC UA application instructions in to Application Protocol Data Unity (APDU) messagse and forwards them to smart card, which is in charge of user authentication and processing secure messaging between card application and chip card pair.

Moreover thanks to self-containment structure, smart card itself does not dependent on other external resources, which could be extreme vulnerable to

potential secure attack, and therefore provides a better hardware security and OS security.

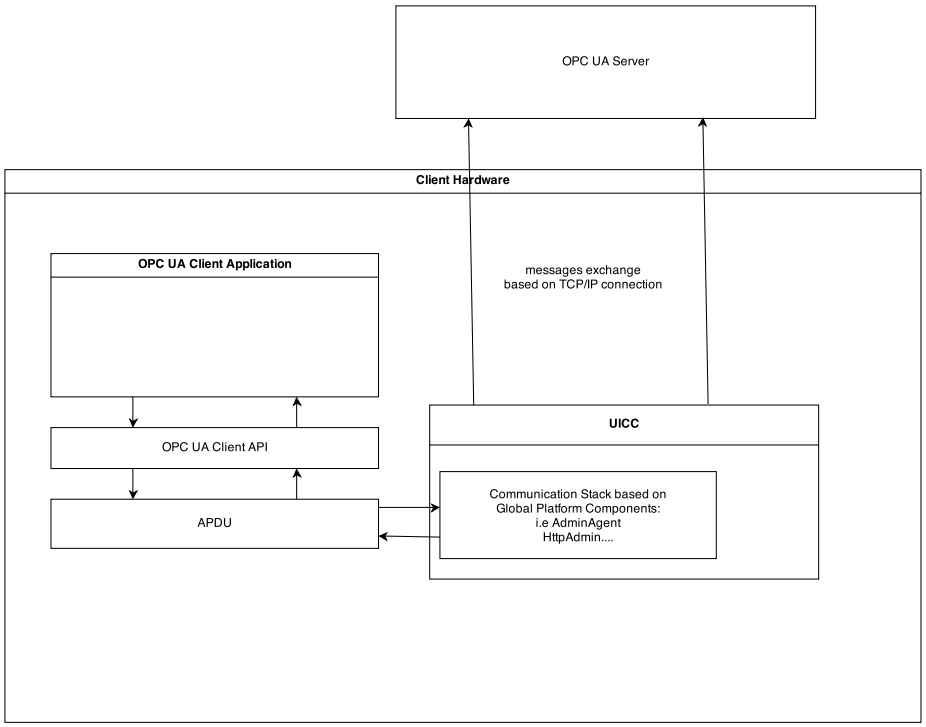


**Fig. 16.** Client Structure

**Client Structure** As described in figure 16, the OPC UA client consists of client application code that realizes client application level functions, OPC UA client API that translates client application instructions into APDU and forwards APDU to UICC smart card. The Communication stack is developed and integrated with UICC card, which is in charge of creation and management TCP/IP connection, secure channel between client and server. This communication stack is based on card OS components provided by Global Platform. Global Platform provides and defines communication flow between an application provider and smart card, allows information exchanged between a remote entity and a card. The on card component, which is responsible for connection creation with the remote entity, is called Security Domain. And the remote entity also is referred as Remote Administration Server. With those concepts, smart card with Security Domain can act as HTTP client and is capable of packing APDU formate information into HTTP POST message. At the same time, the

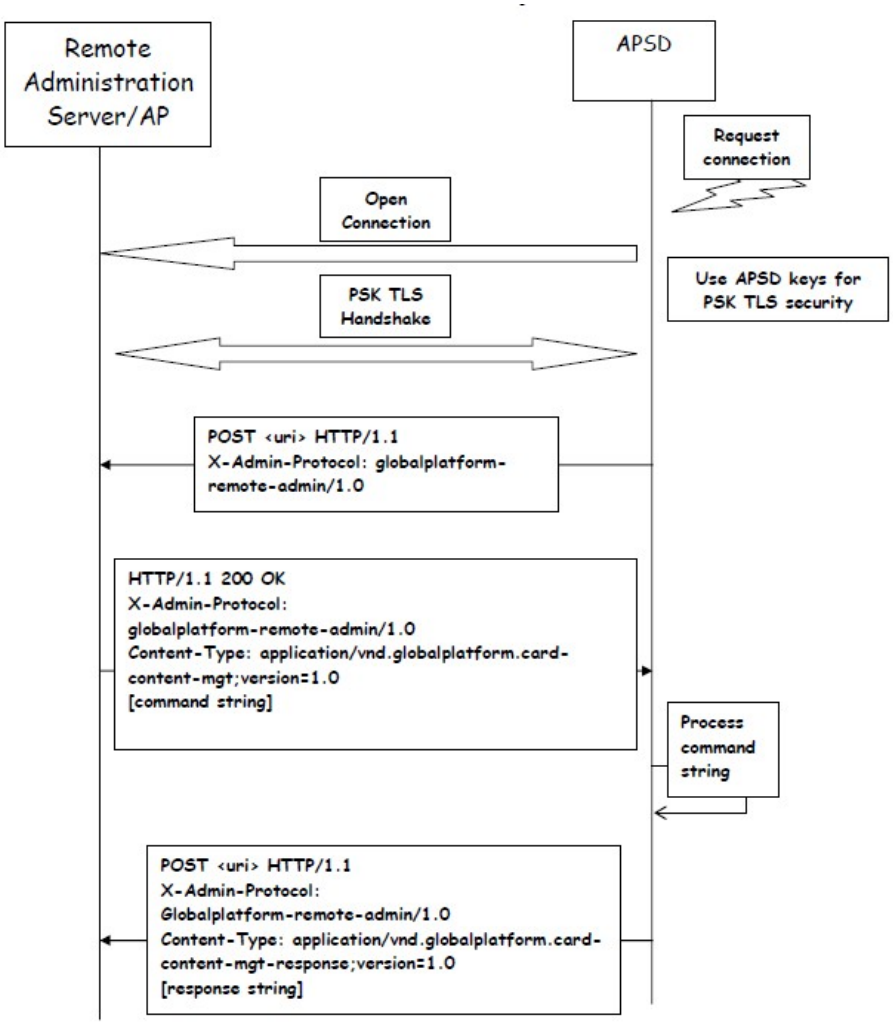


**Fig. 17.** Communication Flow between an AP and corresponding APSD[12]

Remote Administration Server also is a HTTP server, which can send HTTP message including APDU format information to its client.[12]

Figure 17 illustrates a typical communication flow between administration server and corresponding security domain on smart card. As can be seen, the request for open communication is alway initialized by security domain, which is also the phone user. After a successful creation of secure handshake, the remote administration server and security domain is able to using HTTP message to exchange command and response strings, which include APDU instructions from OPC UA client and server.

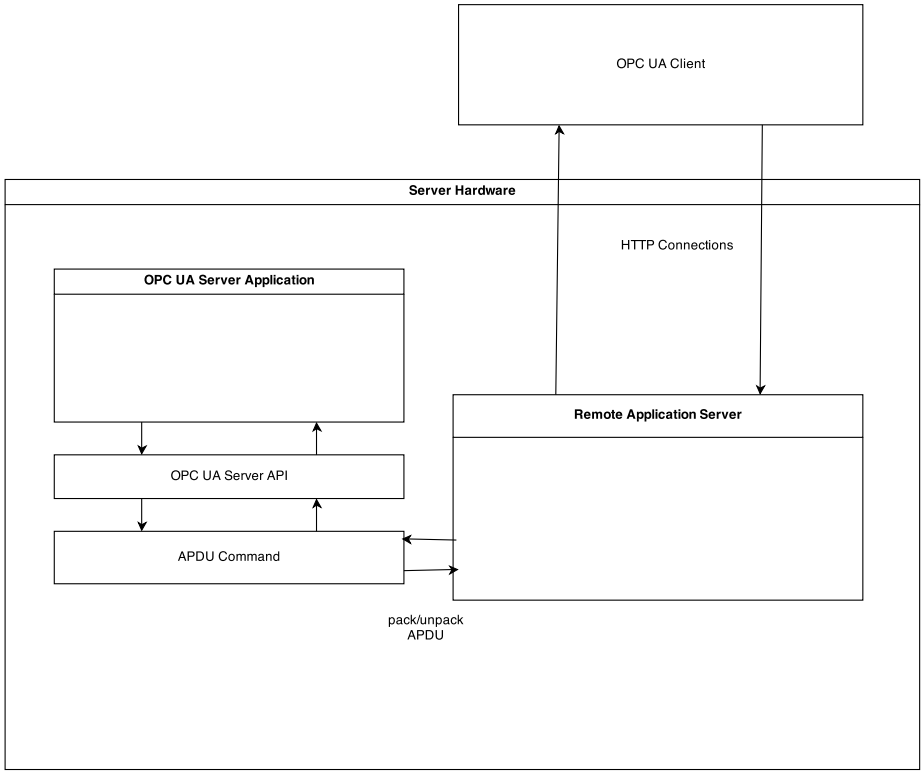


**Fig. 18.** Server Structure

**Server Structure** The server structure for Smart Home is pictured as figure 18 and it consists of an OPC UA server application that implements smart home server services, a remote application server, which realizes server side communication stack, together with on card embedded security domain manages HTTP connections, secure channel construction as well as user authentication.

## References


1. OPC Foundation: Opc unified architecture specification part1 overview and concepts 1.02. (July 10.2012)
2. OPC Foundation: Opc unified architecture specification part3 address space model 1.01. (February 6.2009)
3. OPC Foundation: Opc unified architecture specification part4 services 1.01. (February 6.2009)
4. OPC Foundation: Opc unified architecture specification part2 security model 1.01. (February 6.2009)
5. OPC Foundation: Opc unified architecture specification part6 mappings 1.01. (February 6.2009)
6. 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].
7. Wikipedia contributors: Constrained Application Protocol. `http://en.wikipedia.org/wiki/Constrained_Application_Protocol` [update;Februray 8,2014].
8. Jean-Louis Carrara,Herv Ganem,Jean-Franois Rubon,Jacques Seif: The role of the uicc in long term evolution all ip networks. (January 2009)
9. Wolfgang Rankl und Wolfgang Effing: Handbuch der chipkarten - 5. deutsche auflage. (2008)
10. Sun Microsystems: Java card applet developers' guide. (July 1998)
11. GlobalPlatform: Globalplattform card specification. (January 2011)
12. GlobalPlatform Card: Remote application management over http card specification v2.2 amendment b. (March 2012)