Software Architecture of JTAG Security System

SANG-GUUN YOO, KEUN-YOUNG PARK, JUHO KIM
Department of Computer Science and Engineering
Sogang University
Mapo-gu Shinsoo-dong Sogang University, Seoul
REPUBLIC OF KOREA
jhkim@sogang.ac.kr

Abstract. : - The issue of JTAG security has recently become of interest not only to academic researchers but also to industrial entities. As a response to this security need, several security approaches using fuses, key matching, and three-entity authentication approaches have been proposed. However, each of those solutions only provides the idea of the security mechanism or implementation of the hardware part of the security solution without thinking of how the user can access such a solution in an effective manner in terms of ease of use, administration, and practicality. In this paper, we share our experience in developing a real-life complete software solution for a JTAG security system. The proposed software solution provides benefits such as ease of use/administration, complete functionality, scalability, maintainability, and practicality. This work also shows how a user-specific software solution can overcome the limitations of commercial applications and improve the efficiency of special processes.

Key-Words: JTAG, software architecture, RUP, security

1 Introduction

Testability is a very important property of every hardware device allowing the user to verify if the device works correctly. In general, tests can be divided into two groups: tests performed in the factory during device production, and tests performed during normal usage of the device. Tests that belong to the first group permit the manufacturer to select and reject devices that do not comply with the assumed specification. The other group of tests is dedicated to be performed during or along with normal usage of the device. Their assignment is to identify errors in device operation and indicate its failure to the user. As a solution to provide testability with ease of use and effectiveness, JTAG was proposed. JTAG, also known as Boundary Scan, was standardized in IEEE 1149.1 [1]. This standard defines a 5-pin serial protocol for accessing and controlling the signal-levels on the pins of a digital circuit, and has some extensions for testing the internal circuitry on the chip itself. However, because of the open access characteristic of JTAG, this technology has been used many times by unauthorized users to perform different kinds of attacks, such as firmware modifications and logic/circuit reverse engineering [2-5]. For this reason, different groups have decided to include security for JTAG as requirements in their specifications, as occurs in the OMTP Hardware Requirements and Defragmentation [6].

The issue of JTAG security has recently become of interest not only to academic researchers but also to industrial entities and there have been several approaches proposed [7-16]. However, each of those solutions only provides the idea of the security mechanism or the implementation of the hardware part of the security solution without thinking about how the user can access such a solution in an effective manner in terms of ease of use, administration, and practicality. In this paper, we describe our experience in developing a complete software solution for implementing a real-life secure JTAG environment. The proposed software solution provides benefits such as ease of use/administration, complete functionality, scalability, maintainability, and practicality. We have based our software solution based on the hardware and protocol proposed in our previous work, which is detailed in [16].

The rest of the paper is organized as follows. Section 2 overviews the JTAG technology and why its security is important. Section 3 then describes briefly the JTAG security system based on credentials, which is the system upon which we have based our software solution. Later, in Section 4, we show the details of the development process of the proposed application. Finally, Section 5 concludes the paper.

2 Background

JTAG (Joint Test Action Group) is the common name for the standard IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture [1]. JTAG was designed initially to handle some problems of digital systems: faults in design, fabrication, packaging, and PC boards. However, it has been extended in various directions, including features such as in-circuit configuration, debugging, and device programming [19, 24]. As JTAG is widely used these days, the necessary connections for the JTAG interface are available on most currently sold electronic devices, allowing developers and testers to access the internal resources of devices, including memories and processors. However, JTAG also opens possibilities for a malicious user to use this technology for performing attacks to disable or corrupt the system, extract embedded code or crypto keys, embed disallowed functionalities without detection, duplicate system design, and so forth [2-5], creating a serious threat to the electronic device's security.

The case of iPhone hacking is an example how misuse of JTAG can allow functionality modification of devices. On the Internet we can find different applications that permit one to disable the SIM Lock and Software control features of iPhones. An example of the steps that the attacker can follow to create such an application is as follows. (1) The target device is opened to find the JTAG interface. Many devices hide the JTAG pins to protect the device from attackers. However, the attacker uses the datasheets, manuals, hardware tools and probing techniques for wires and pins to find the JTAG port (Fig. 1). (2) Once the JTAG interface is found, logical analysis tasks like device memory reading, extraction of the firmware, dumping of memory, and test data sampling are executed. (3) Using the information collected in the previous step, reverse engineering of the firmware and data stored in the device is executed. Through reverse engineering, the attacker develops a firmware modification. (4) Using the JTAG port, the cracked firmware is uploaded to modify the functionality of the device (e.g. SIM unlocking and Jailbreaking).

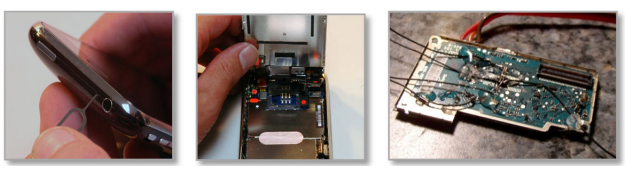

Fig. 1 Finding JTAG interface

There are many other cases apart from iPhone modifications, such as XBOX360, Wii, and PlayStation modifications. In those cases, JTAG was used to understand the illegal copy protection features to then create a hardware component to avoid such protection mechanisms (e.g. mod-chip). JTAG was also used to determinate the Secret Key of cryptographic chips.

As a response in order to reduce the risks produced by JTAG misuse, many approaches such as [7-16] have been proposed. However, each of those solutions only provides the concept of the security solution or implementation of the hardware part of the security mechanism, without proposing how the user can access the solution in an effective manner in terms of ease of use, administration, and practicality. The actual software solutions for JTAG provide only limited features that cannot be adapted for such security solutions. Some advanced solutions, such as TRACE32, offers script languages (e.g. PRACTICE [17]) for the execution of user-specific processes; however, even those facilities only provide very limited capabilities and they are not practical in terms of ease-of-use, scalability, user interactions, and so forth.

In the next sections, we share our experience in developing user-specific software in solving the problem of automation of a JTAG Security System. Among the different JTAG security solutions, we have decided to select our previous work [16] because of its balanced benefits in terms of security and usability.

3 JTAG Security System Based on Credentials: Overview

In this section, we explain briefly our previous work, called JTAG Security System Based on Credentials, which we have based our software solution on. As shown in Fig. 2, there are three components that participate in the solution: Secure JTAG, the Host Computer, and the Secure Authentication Server. Secure JTAG is a hardware module that allows blocking and unblocking of the JTAG functionality through user authentication, the Host Computer is the computer of JTAG user where the JTAG tests are performed, and the Authentication Server (or just Server) is the secure component where the user account information of JTAG users is stored.

In this scheme, only those users who own the correct credential issued by the Server and its password are allowed to use the JTAG port. Therefore, users must ask for credentials to the

server after a correct user authentication. Once the credential has been issued, no further communication with the server is necessary for authentication with Secure JTAG. The scheme is composed of two phases: the credential issue phase, and the user authentication phase, as shown in Fig. 2.

The credential issue phase is the stage at which the user receives a credential from the server. In this phase, the server verifies the identity of the user and provides the credential if the user is authenticated and has permission to access the JTAG port. This phase is executed with the following steps. First, the user sends a request for a credential through the host which receives the target JTAG's ID and a challenge from Secure JTAG. Next, the host passes the data received from the device along with the user and host information and the credential's password (selected by the user) through an encrypted channel to the server. The server then checks the validity of the data sent by the user and authenticates the user. As a response to a valid verification, the authenticated user receives the credential and the credential verification data. The user then sends to Secure JTAG the credential along with credential verification data as a response to the challenge. When Secure JTAG receives this information, it checks its validity and after a valid verification it generates the User Verification Data which is used in the user authentication phase.

The user authentication phase is the process of authenticating the JTAG user who owns a valid credential corresponding to a Secure JTAG. In this phase, Secure JTAG authenticates the user based on the submitted credential and its password. The user authentication phase is executed with the following steps. First, the user sends a request for access through the host which triggers Secure JTAG to send a challenge to the host. Then, the host generates a response using the credential and the password sent by the user, and sends them to Secure JTAG. Finally, Secure JTAG verifies the response in order to authenticate the user and check his/her permission, then controls the user's access based on the permission.

The scheme also includes a password change protocol that allows users to change their passwords at any moment.

3.1 Security Analysis

The security of the system was proven by using different analysis in [16]. However, we have included several additional verifications to reconfirm its security in this paper. We recommend to read [16] for the complete security analysis of the system.

Security against insider attack: An insider (legal user) may try to impersonate to be another user. However, this is not possible because the credential is issued independently per user using his/her independent identification/password combination. On the other hand, a legal user with a valid certificate can try to guess secret values using the data that he or she has access. However, this is also not possible because the secret values are not deducible using the values that are reachable by a user i.e. certificates and communication messages.

Impersonation of User: We can assume that the attacker who knows valid user identification can attempt to receive a credential using its own password for the certificate. However, this attack is not possible because the Server decrypts the received information using the random session key generated previously and because the credential is issued only after a valid verification of the challenge-response process. This means that the attacker cannot obtain the random session key and execute a valid challenge-response process without the valid password of the user, implying that the attacker cannot deceive the server with spoofed information.

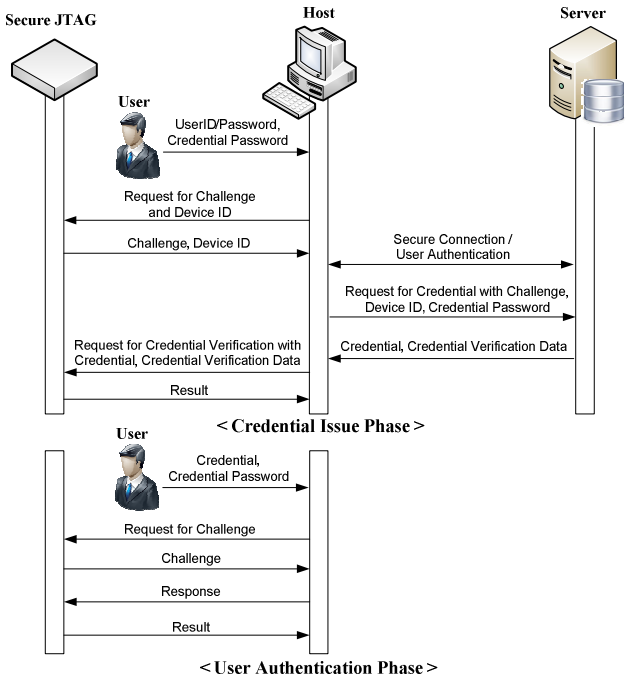

Fig. 2 General view of JTAG Security System Based on Credentials

The proposed JTAG user authentication scheme uses a credential issued by the server; this

information may leak because it is managed by the debug host. Therefore, an attacker may be able to copy the credential and attempt to authenticate against Secure JTAG. However, this attack is not feasible because in order to authenticate against Secure JTAG, it is necessary to generate the correct response for the challenge sent by Secure JTAG; however, even though the attacker has the correct Certificate, he cannot generate the correct Response without the password of the certificate.

Impersonation of Secure JTAG: An attacker can try to impersonate a Secure JTAG to capture confidential information from the user during the Certificate Issue or Authentication phases. However, as the messages sent by the user to the Secure JTAG do not contain any confidential information, the attacker cannot obtain any important value.

Impersonation of Server: An attacker can try to impersonate the Server to capture secret data from the user during the Certificate Issue and Password Change phases. However, this is not possible because the user does not send his/her password any moment. Additionally, the authenticity of the server is verified by the user by comparing the response of the server with the value calculated by the user; this is possible because the response of the server sent to the user is calculated using the password of the user stored only in the authentic server.

Replay Attack: Assume that an attacker can eavesdrop on messages transmitted between Secure JTAG and Host computer. In such case, the attacker can copy the challenge-response values and try to reuse them in new authentication trials. However, all such trials will be denied by the Secure JTAG because each authentication request makes Secure JTAG to generate a new random nonce which means it will generate new challenge message, and the response (captured previously) sent by the attacker will be different from the new requested response value.

Brute force attack: An attacker can attempt to authenticate against Secure JTAG repeatedly with random or sequential data. The attacker can send an authentication request to Secure JTAG and generate the response for the received challenge. However, in the same way as the replay attack, each request Secure JTAG generates a new response, making the attack infeasible.

Sniffing Attack: The attacker can attempt to obtain confidential information by listening to the network. However, any confidential information regarding to passwords or keys are not sent directly but sent through a secure symmetric cryptographic algorithm. Therefore, the attacker cannot obtain any valid authentication information by sniffing the network.

Security Analysis per Levels: (1) Server: First of all, the server is assumed to be secure. Therefore, the confidentiality of the secret data stored in the server is not considered in this paper. However, the security against the impersonation of this entity is guaranteed as explained previously in "Impersonation of Server". **(2) Host:** In case of the host, it stores a confidential value, the certificate. However, its leakage does not represent any security problem because it is useless without its password. Additionally, it is not possible to derivate any secret data from its value. **(3) Trace32:** The trace32 only works as a link between the secure JTAG and the client software (it does not execute any calculation during the certificate issue and authentication phases). Therefore, there is not any important threat in this level. **(4) Secure JTAG:** The security of the Secure JTAG is guaranteed by the secure storage of its key (using secure memory technologies [27]). Additionally, the security against the impersonation of Secure JTAG is guaranteed as explained previously in "impersonation of Secure JTAG".

4 Development of a Total Software Solution for the JTAG Security System Based on Credentials

We have decided to use the Rational Unified Process (RUP) [18, 25] in the development of the proposed software. RUP is a software engineering process that provides a disciplined approach to delegating tasks and responsibilities which can be used in the development of different types of software such as learning systems [21], service applications [22], and simulators [23]. RUP is also a guide for how to effectively use the Unified Modeling Language (UML) [26] which is an industry-standard language that allows developers to clearly communicate requirements, architectures, and designs. The goal of RUP is to ensure the construction of high-quality software that meets the needs of users, within an expected timetable and budget. The Rational Unified Process divides the development cycle into four phases: inception, elaboration, construction, and transition phases. Each phase is concluded with a well-defined point in time at which certain critical decisions must be made, and therefore key goals must have been achieved. In this section, we give details of the outcomes of the development of the software solution for the JTAG security system based on credentials separated by each phase of RUP.

4.1 Inception Phase

During the inception phase, the business case and delimitation of the project scope are established. All entities of the system called actors and the interactions with the system at a high-level are identified. This involves identifying all use cases and describing a few significant ones. Below is the resultant output of the inception phase.

We have decided to call the software solution "Secure JTAG Software Suite". Fig. 3 shows the general use case model of the proposed Secure JTAG software suite which has three actors: one representing the role of the JTAG user, which is the person who requires JTAG features to execute the development/testing processes; the second is the Administrator of the system who manages the system; and finally Secure JTAG which is the hardware logic that controls the access to JTAG features. The JTAG user can execute the credential issue process using the "Get Credential" use case. He/she can also change the password using the "Change Password" use case, import/export/delete credentials by using the "Manage Credential" use case, and administer the configuration of the host computer by using the "Manage Host Configuration" use case. On the other hand, the Administrator is allowed to manage the database of the system and the configuration of the server by using the "Manage Database" and "Manage Server Configuration" use cases, respectively. Finally, the Secure JTAG role interacts with the software suite by using the "Authenticate Secure JTAG" use case.

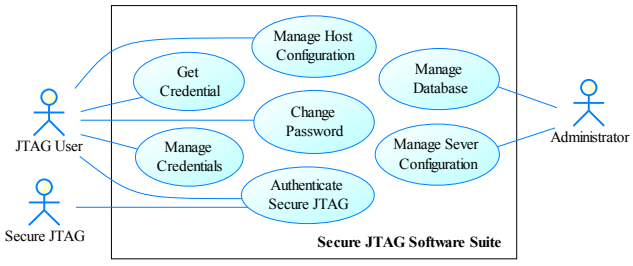

Fig. 3 General Use Case Model

4.2 Elaboration Phase

The purpose of the elaboration phase is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project.

4.2.1 General Architecture

The network connection model of the JTAG security system is shown in Fig. 4. The system involves four hardware levels. The first level is the Server, the infrastructure where the user accounts and user permissions to access JTAG devices are stored. The second one is the Host which is the computer from which the user gets the credentials and/or access to the JTAG port. The third level is the TRACE32 Hardware which is the In-Circuit Emulator selected to access the JTAG resources; we have chosen TRACE32 because it is manufactured by the world's largest producer of hardware assisted debug tools, i.e. Lauterbach, and because it is one of most widely used debugging tools in the high technology industry, and also because (most importantly) it was a requirement of the research grant provider, i.e. Samsung Electronics. Finally, the fourth level is the JTAG device, the device in which Secure JTAG is incorporated.

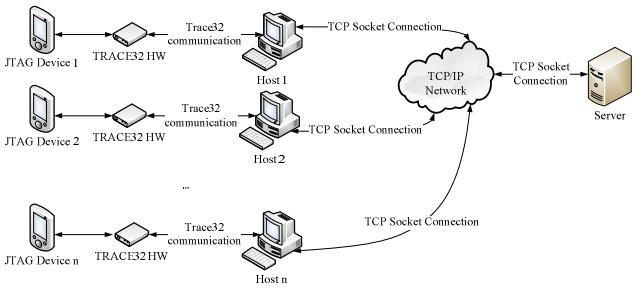

Fig. 4 Network Connection of the JTAG Security System

Based on the network connection model, we have concluded that the development of three different applications is necessary (see the architectural model in Fig. 5).

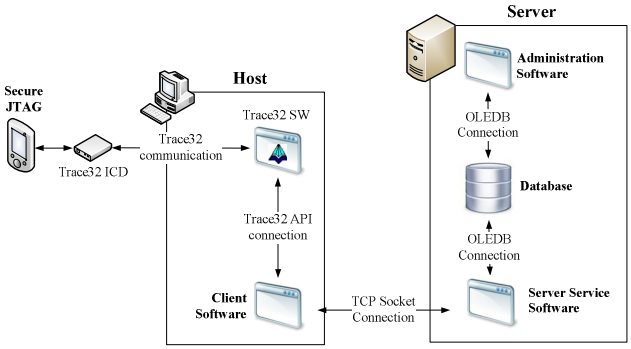

Fig. 5 Architectural Model

1) Client Software: the client software is the application used by the JTAG user to request/receive the credential from the server, authenticate to Secure JTAG to enable access to the JTAG port, and request a password change. The Client Software uses the TRACE32 API features to communicate with Secure JTAG and an encrypted

TCP socket channel to communicate with the Server to execute the Credential issuing and password changing processes.

2) Server Service Software: this is the software that responds to requests from the Client Software. It opens a TCP port for listening to client software requests and is executed as a service in the Server.

3) Administration Software: this application is used by the Server Administrator to manage the database which stores information about the system.

Both applications are executed in the server i.e. the Server service software and Administration software and access a common database. The technology selected for database connection is OLEDB.

4.2.2 Use Case Modeling

The general use case model detailed in Fig. 3 is broken down into three use case models according to the detailed architectural model of Fig. 5. The use case diagram of the Client software is shown in Fig. 6. The Client software has three actors: one representing the role of the JTAG user, the person who requires the features of JTAG to execute the development or testing processes; the second representing the Server service software, which is the software executed in the Server and provides to the Client software the credential issue and password change features; and TRACE32 which is the system that allows the client software to communicate with Secure JTAG. The JTAG user can request a credential issue and password change using the "Get Credential from Server" and "Change Password" use cases which will perform their steps communicating with the Server Service Software. The user can also execute different use cases related to the management of credentials and configuration of the application. Finally, the user can communicate with the TRACE32 system to request the locking or unlocking of the JTAG port by using the "Lock JTAG" and "Unlock JTAG" use cases.

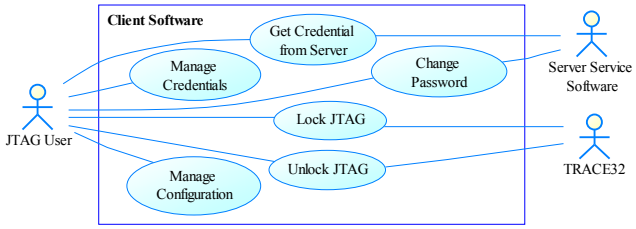

Fig. 6 Client Software Use Case Diagram

The use case diagram of the Server service software is shown in Fig. 7. The Server Service has two actors: one representing the role of the Server administrator, the person who manages the Server service software; and the second representing the Client software. The Server administrator can initiate and stop the functionalities of the software using the "Start Service" and "Stop Services" use cases, respectively. He/she can also modify the configurations of the software using the "Manage configurations" use case. On the other hand, the Client Software can request/get a credential and change the password of a JTAG user by using the "Issue Credential" and "Change Password" use cases, respectively.

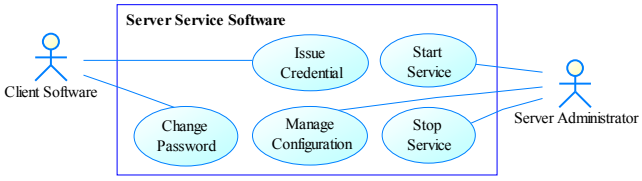

Fig. 7 Server Service Software Use Case Diagram

The use case diagram of the Administration software is shown in Fig. 8. The Administration software has one actor representing the role of the Server administrator who manages the different types of registers of the database.

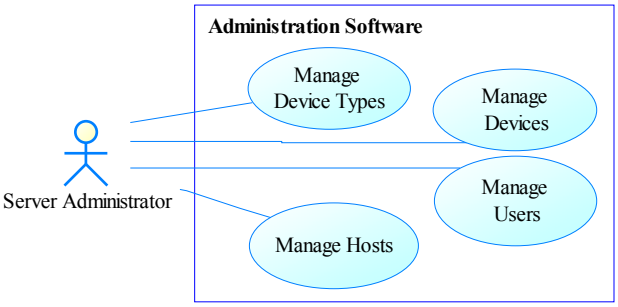

Fig. 8 Administration Software Use Case Diagram

4.2.3 Detailed Architecture

4.2.3.1 Client Software

The client software is comprised of forms, classes, external files (configuration file and credential files), and a special library called TRACE32 API encapsulated in the t32api.dll file. Details are shown in Fig. 9.

Forms: Forms provide the visual interface to interact with the user and detect events (e.g. the click of a button, the filling out of a textbox). Events detected by forms are processed by the code incorporated in each event. Event codes create object instances of classes, and these objects are used to communicate with Secure JTAG (through

TRACE32 API), Server Service Software, Configuration File and Credential Files. We have designed two forms in this software: (1) the main form which executes almost all functions of the application, and (2) the configuration form which provides the interface to configure the application.

Classes: The client software manages five different classes. The "cls_user" class manages the data and operations related to JTAG user. The "cls_user" class uses the classes "cls_trace32" and "cls_connectionToServer" to create TRACE32 API and TCP connections to access the Secure JTAG and the Server Service Software, respectively. Additionally, there are two additional classes, "cls_credential" and "cls_configuration", to manage the credentials issued by the server and the configuration data of the application.

External Files: The client software uses two types of external files: credential and configuration files. Credential files are those files containing the credential that allows users to access the JTAG port. This kind of file is created after a successful credential issuing process and its format is shown in Fig. 10. On the other hand, the Configuration File stores the configuration information about the server and its format is shown in Fig. 11.

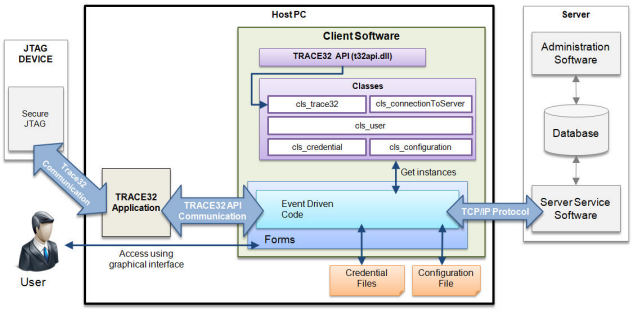

Fig. 9 Detailed View of the Client Software

"CredentialFile"	Prefixed text to identify the Credential file
Issue DateTime	Date and Time when the Credential was issued
Username	Username of the User allowed to use the Credential
Device ID	Device ID of the device for which the Credential was issued
Serial Number	Serial Number of the device for which the Credential was issued
$E_{h(credentialPW)}(C)$	Encrypted value of the Credential (C)

Fig. 10 Format of the Credential File

Server IP	IP address of the Server executing the Server Service Software
Server Port Number	TCP Port Number of the Server Service Software
TRACE32 API Port Number	TRACE32 API Port Number
Secure JTAG Base Address	Base (initial) address of the registers of the Secure JTAG

Fig. 11 Format of the Configuration File of the Client Software

TRACE32 API: This API (Application Programming Interface) provides a software interface that enables control of the TRACE32 software. In our development, we have used the latest library provided by Lauterbach. For additional information, we recommend reading the documents available in [20].

4.2.3.2 Server Service Software

The Server Service Software is comprised of forms, classes, and external files (configuration file and log files), as shown in Fig. 12.

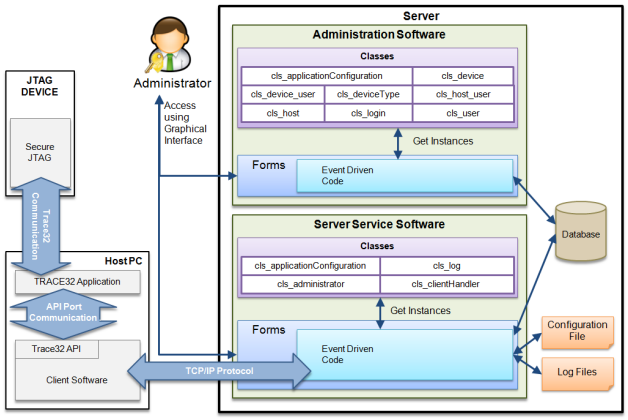

Fig. 12 Detailed View of the Server Service and Administration Software

Forms: forms provide a visual interface to interact with the administrator of the server and detect events (e.g. the click of a button, the filling out of a textbox). Events detected by the forms are processed by the code incorporated in each event. Event codes create object instances of classes, and these objects are used to communicate with the Client Software, Database, Configuration File and Log Files. There are two forms in this software: (1) the main form which executes almost all functions of the application, and (2) the configuration form which provides the interface to configure the application

Classes: The server service software manages four different classes. The "cls_administrator" class manages the data and operations of the administrator. The application uses the "cls_clientHandler" class to manage each TCP connections coming from the client software. Additionally, the Server Service Software uses the "cls_applicationConfiguration" class and "cls_log" class to manage the configuration data of the application and the different logs generated by the processes, respectively.

External Files: The Server Service Software uses two types of files: log and configuration files.

There are two kinds of log files. The first kind, called a Server Log File, stores information about the status change of the Server Service Software (e.g. date/time when the service starts and stops); and the second kind, called a Client Log File, stores the log information about client connections. On the other hand, the configuration file stores the configuration of the Server service software. The format of the configuration file is shown in Fig. 13.

Server IP	IP of the server
Port	TCP Port opened by the Server Service Software
# of connections	Maximum number of concurrent connections
Use IP permission:	Indicates if the Server Service Software will process the IP permission access control. Possible values are "TRUE" or "FALSE"
Use MAC permission:	Indicates if the Server Service Software will process the MAC permission access control. Possible values are "TRUE" or "FALSE"

Fig. 13 Format of the Configuration File of Server Service Software

4.2.3.3 Administration Software

The Administration Software is comprised of forms and classes. Details are shown in Fig. 12.

Forms: Each form provides the visual interface to interact with the Administrator, and detects the events. Events detected by forms are processed by the code incorporated in each event. Event codes create object instances of classes, and these objects are used to communicate with the database. There are twelve forms in this software.

- frm_main: this is the main MDI form that contains the main menu of the application.
- frm_authentication: this form is loaded first when the software is executed. This form contains the visual interface to allow the administrator to log in to the application.
- frm_deviceType: this form allows the administrator to manage registers related to device types.
- frm_device: this form allows the administrator to manage registers related to devices.
- frm_host: this form allows the administrator to manage registers related to host computers of users of JTAG devices.
- frm_addIP: this form allows the addition of IP addresses of a host register.
- frm_addMAC: this form allows the addition of MAC addresses of a host register.
- frm_login: this form allows the administrator to manage registers related to administrator users of this software.
- frm_user: this form allows the administrator to manage registers related to JTAG users.
- frm_deviceList: this form allows the addition of devices that a user can access.
- frm_hostList: this form allows the addition of hosts from which a user can access the JTAG device.
- frm_about: this form shows information about the Software.

Classes: The Administration Software manages seven different classes. The classes "cls_deviceType", "cls_device", "cls_device_user", "cls_host", "cls_host_user", "cls_login", and "cls_login_user" are used to manage tuples of different entities of the database. On the other hand the "cls_applicationConfiguration" class is used to manage the configuration data of the application.

4.2.4 Database Modeling

Both applications executed in the Server, i.e. the Server Service Software and Administration Software, make use of a common database which stores all the information about users, devices, hosts, and so forth. The conceptual model of the database is shown in Fig. 14, and below are the descriptions of each entity.

- LOGIN: Contains information about the users authorized to use the Administration Software.
- DEVICETYPE: Contains information about the Types of Devices (JTAG devices classification). For example: Mobile Phone, Embedded Board, etc.
- DEVICE: Contains information about JTAG devices.
- USER: Contains information about JTAG users that are allowed to authenticate to the JTAG Devices listed in the Devices Table.
- HOST: Contains information about Hosts (Computer, Network, Ranges of computers) from which a user can authenticate, change password or perform the certification issue process.
- IP: List of IP addresses of hosts.
- MAC: List of MAC addresses of hosts.

4.3 Construction Phase

During the construction phase, all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. We have decided to develop the JTAG Authentication Suite in C# with the Microsoft .NET Framework 3.5 using Visual Studio as the IDE because of the benefits such as simplicity,

object orientation, and rapid development (see Fig. 15).

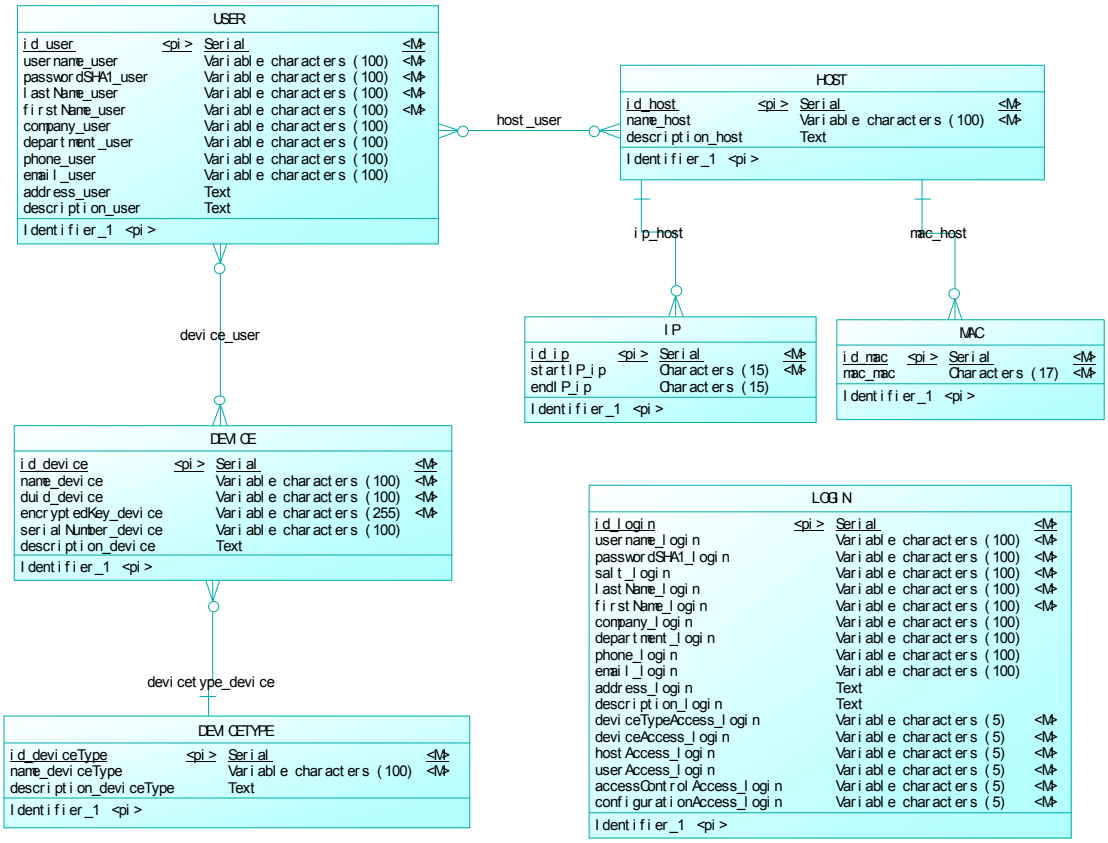

Fig. 14 Conceptual Model of the Database

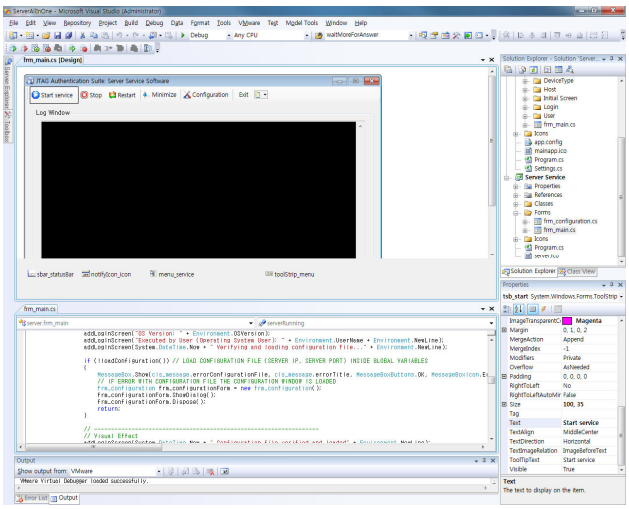

Fig. 15 Development Environment

4.4 Transition Phase

The purpose of the transition phase is to transition the software product to the user. Once the product has been given to the end user, issues usually arise that require you to develop new releases, correct some problems, or finish the features that were postponed. The simulation environment was constructed as shown in Fig. 16. The Secure JTAG logic was implemented in CT1156T2F-S Realview Emulation Board using the logic as explained in [16]. The Host computer and the embedded board were connected using the In-circuit emulation tool TRACE32 ICD. The Client Software was installed in the Host computer; additionally, the TRACE32 software was loaded in the Host computer to establish communication between the Client Software and TRACE32 ICD. The Server Service Software and the Administration software were installed on a server. Finally, the database was implemented in Microsoft SQL Server 2007. Fig. 17 and Fig. 18 show the software screens of the Client Software, Server Service Software, and Administration Software.

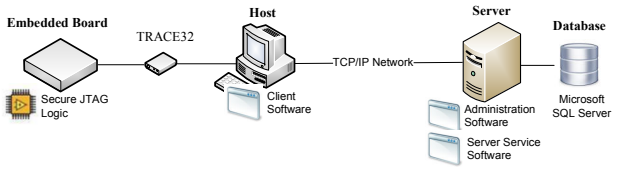

Fig. 16 Simulation Environment

We have executed different simulations in terms of functionality and security, and all simulations were executed without problems and neither

notorious delays nor security holes were present, demonstrating that the proposed software is usable in a real JTAG authentication environment.

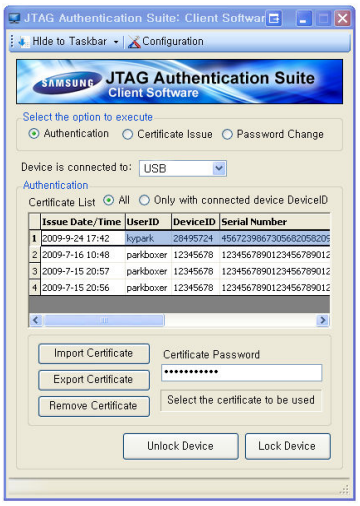

Fig. 17 Client-side user authentication software

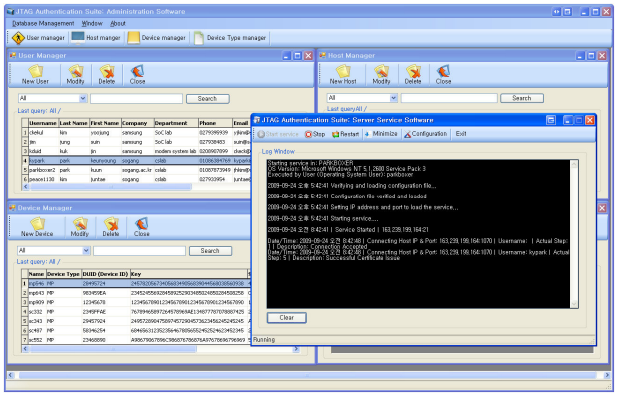

Fig. 18 The Server service and Administration software

With the simulation, we have shown how the proposed software suite has improved the JTAG security solution in terms of security and usability. In the case of security enhancement, we can say that before the Software Suite, the JTAG authentication process was executed using the PRACTICE Script Language which obligated users to store the credential password in plaintext (or simple codification) inside the script file, creating a high probability of password leakage. Additionally, the software improves security by creating an encrypted channel between the host computer and the server offering protection against sniffing, spoofing, replay, and other network based attacks. In the case of usability, the user does not need to interact manually with the TRACE32 software anymore to execute different script files containing the commands of different steps of the authentication protocol, because the software suite provides a friendly user interface to efficiently access the different features of the system. Furthermore, the proposed software does not modify the working environment (in this case, the TRACE32 environment), allowing developers/testers to work without any additional effort.

5 Conclusion

In this paper, we have presented a case study in which we have developed the automation of a JTAG security system. The RUP methodology has been applied allowing the participant of the software development to clearly communicate requirements, architectures, and designs. In the project, after developing the models for the software, we validated the models by directly working and interacting with the engineers who were responsible for implementing the different parts of the software suite. Our case study has illustrated how a total solution for JTAG security can be developed and also has shown how user-specific software can improve the efficiency of special processes by complementing the features of commercial legacy software without modifying the actual working environment.

Acknowledgments

Part of this research was funded by the Industrial-Academic Projects of Samsung Electronics. We would like to thank the modem R&D team for research fund support.

References:

[1] IEEE, IEEE Std 1149.1-2001 - IEEE Standard Test Access Port and Boundary Scan Architecture, 2001.

[2] B. Yang, K. Wu, R. Karri, Secure scan: a design-for-test architecture for crypto chips, *IEEE Trans Comput Aided Des Integr Circuits Syst*, Vol. 25(10), 2005, pp. 2287–2293. DOI: 10.1109/TCAD.2005.862745.

[3] M. Breeuwsma, Forensic imaging of embedded systems using JTAG (boundary-scan), *Int J Digit Forensics Incident Response*, Vol. 3(1), 2006, pp. 32–42. DOI: 10.1016/j.diin.2006.01.003.

[4] B. Jack, Exploiting embedded systems, *Black Hat 2006*, Las Vegas, USA. http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Jack.pdf, Accessed 15 Jul 2011.

[5] A. Becher, Z. Benenson, M. Dornseif, Tampering with Motes: Real-World Physical

Attacks on Wireless Sensor Networks, *LNCS 3934*, 2006, pp. 104-118.

[6] OMTP Hardware Working Group, OMTP hardware requirements and defragmentation, *Trusted Environment OMTP TR0*, Open Mobile Terminal Platform, 2006.

[7] A. Ashkenazi, D. Akselrod, Platform independent overall security architecture in multi-processor system-on-chip integrated circuits for use in mobile phones and handheld devices, *Comput Electr Eng*, Vol. 33(5-6), 2007, pp. 407–424. DOI: 10.1016/j.compeleceng.2007.05.003.

[8] D. Hely, F. Bancel, M. Flottes, B. Rouzeyre, Securing scan control in crypto chips, *J Electron Test: Theory Appl*, Vol. 23(5), 2007, pp. 457-464. DOI:10.1007/s10836-007-5000-z.

[9] W. Moyer, M. Fitzsimmons, Integrated circuit security and method therefor, *United States Patent*, Patent No. US7266848B2, 2007.

[10] F. Novak, A. Biasizzo, Security extension for IEEE std 1149.1, *J Electron Test: Theory Appl* Vol. 22(3), 2006, pp. 301–303. DOI: 10.1007/s10836-006-7720-x.

[11] M. Comulkiewicz, M. Nikodem, T. Tomczak, Low-cost and universal secure scan a design-for-test architecture for crypto chips, *International Conference on Dependability of Computer Systems (DEPCOS-RELCOMEX)*, 2006, pp 282–288. DOI: 10.1109/DEPCOS-RELCOMEX.2006.36.

[12] R. Kapur, Security vs. test quality: are they mutually exclusive?, *International Test Conference (ITC)*, 2004, pp. 1414. DOI: 10.1109/TEST.2004.1387422.

[13] R. Kurt, K. Ramesh, Attacks and defenses for JTAG, *IEEE Des Test Comput*, Vol. 17(1), 2010, pp. 36–47. DOI: 10.1109/MDT.2010.9.

[14] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, Securing scan design using lock & key technique, *International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, 2005, pp 51–62. DOI:10.1109/DFTVS.2005.58.

[15] R. Buskey, B. Frosik, Protected JTAG, *International Conference Workshops on parallel Processing (ICPP)*, 2006, pp 405–414. DOI: 10.1109/ICPPW.2006.65.

[16] K. Park, S. Yoo, T. Kim, J. Kim, JTAG Security System Based on Credentials, *J Electron Test*, Vol. 26, Number 5, 2010, pp. 549-557. DOI: 10.1007/s10836-010-5170-y.

[17] Y. Ki, J. Seo, B. Choi, K. La, Tool support for new test criteria on embedded systems: Justitia, *Proceeding of the 2nd international conference on Ubiquitous information management and communication*, 2000. DOI: 10.1145/1352793.1352869.

[18] K. Fertalj, N. Hlupic, D. Kalpic, RUP and XP - A Modern Perspective, *WSEAS Transactions on Information Science & Applications*, Issue 8, Vol. 3, 2006, pp. 1573-1581.

[19] W. Yin, R. Sun, Z. Wan, Realization of Distributed Remote Laboratory and Remote Debug Software for Embedded System, *WSEAS Transactions on Systems*, Issue 12, Vol. 7, 2008, pp. 1433-1442.

[20] Lauterbach, Lauterbach Development Tools, http://www.lauterbach.com, Accessed 15 October 19, 2011.

[21] F. Neri, A Comparative Study of a Financial Agent Based Simulator Across Learning Scenarios, *Lecture Notes in Computer Science 7103*, 2012, pp. 86-97.

[22] S. Yoo, K. Park, J. Kim. Confidential information protection system for mobile devices, *Security and Communication Networks*, 2012. DOI: 10.1002/sec.516.

[23] S. Yoo, S. Kang, J. Kim, SERA: a secure energy reliability aware data gathering for sensor networks, *Multimed Tools Appl*, 2011. DOI: 10.1007/s11042-011-0735-z.

[24] C. Kao, I. Huang, H. Chen, Hardware-Software Approaches to In-Circuit Emulation for Embedded Processors, *IEEE Design & Test of Computers*, Vol. 25 Issue 5, 2008. DOI: 10.1109/MDT.2008.142.

[25] Rational, Rational Unified Process – Best Practices for Software Development Teams, *Rational Software White Paper TP026B*, Rev 11/01, 2001.

[26] A. Teilans, A. Kleins, Design of UML models and their simulation using ARENA, *WSEAS Transactions on Computer Research*, Issue 1, Vol. 3, 2008, pp. 67-73.

[27] S. Chhabra, Y. Solihin, i-VNMM: a secure non-volatile main memory system with incremental encryption, *ACM SIGARCH Computer Architecture News – ISCA'11*, Vol. 39, Issue 3, 2001, pp. 177-188. DOI: 10.1145/2024723.2000086.