FACTA UNIVERSITATIS
Series:**Electronics and Energetics** Vol. 27, N^{o} 1, March 2014, pp. 137 - 151
DOI: 10.2298/FUEE1401137B

EVALUATING SYSTEM SECURITY USING TRANSACTION LEVEL MODELLING

Aisha Bushager[1], Mark Zwolinski[2]

[1]Department of Information Systems, College of Information Technology, University of Bahrain, Bahrain
[2]Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

Abstract. *The design of secure systems requires the use of security analysis techniques. Security objectives have to be considered during the early stages of system development and design; an executable model will give the designer the advantage of exploring the vulnerabilities early, and therefore enhancing the system security. In this work we create an executable model of a smart card system using SystemC with the Transaction Level Modelling (TLM) extensions. The model includes the security protocols and transactions. The model is used to compare a number of authentication mechanisms with different probabilities of failure. In addition, a number of probable attacks, including theft of a private key and denial of service were modelled to examine the vulnerabilities. The executable model shows that security protocols and transactions can be effectively simulated in order to design improvements to withstand different types of security attacks.*

Key words: *Security Modelling, SystemC, Transaction Level Modelling, Protocols, Smart Cards.*

1. INTRODUCTION

Robust and secure system design requires the selection and implementation of a set of policies, procedures, architectures, technology, and personnel. However, there is no system that is 100% secure; there will always be a way to breach the system. The objective in security analysis is to identify the weak points. This requires modelling and simulation tools.

We have used an executable model of a smart card system as an exemplar, including the security protocols and transactions, to allow examination of the security strengths and weaknesses by executing tests on the model. This paper extends work previously presented [1].

Received January 12, 2014
Corresponding author: Mark Zwolinski
Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
(e-mail: mz@ecs.soton.ac.uk)

2. Related Work

Security protocols are sets of rules designed to ensure particular security goals. However, designing and implementing these protocols is difficult and they may fail against various attacks. To be able to effectively integrate the security protocols at early stages of development, modelling languages and techniques are used to better visualize the entire system. One such modelling tool is Communicating Sequential Processes (CSP), which is a process algebra that is used to describe and analyse security properties and protocols by providing a mathematical framework [2]. However, to be able to use CSP, the designer must have specialized knowledge and training, which limits the usage of this method. GSPML, [3], is a visual security protocol modelling language. Again, this language introduces notations and complex models that are targeted to security specialists.

Stereotypes and tags are used to create and present security requirements and assumptions, constraints may be attached but they should be satisfied by modelling elements with the related stereotype [4]. The Unified Modelling Language (UML) version 2.0 has been widely used to model security protocols [5]. For example, UMLsec [4], [6] is an extension to UML for integrating security related information into UML specifications, by specifying security requirements through stereotypes, tagged values, and constraints[7].

An adversary can be created in UMLsec to model possible threats to a system. UMLsec was used to find possible vulnerabilities in Common Electronic Purse Specifications (CEPS) [4], it was also used to define security permissions that enforce restrictions on the workflows of a system [8].

None of the above modelling languages provides an automatic transition from design to code implementation. A designer would like to have an executable model that allows a better testing of the designed model and therefore links the gap between the design phase and the code implantation phase. In our work, an executable model is produced using SystemC with the TLM extensions [9]. SystemC has been used to produce a methodology to simulate security attacks on smart cards with fault injection [10] and it has also been used to create an environment for design verification of smart cards using security attack simulation [11]. In TLM, communication among computational components is modelled by channels and transaction requests are handled by calling interface functions of these channel models [12].

3. Using UML to Model Smart Card Transactions

As an illustration of our methodology, we use a smart card system. Because smart cards are used to store sensitive data such as PINs, passwords, and keys, they are likely targets for criminal attacks. The main purpose of an attack is to get hold of this data. Attackers might perform various numbers and types of attack on the smart card system.

3.1. Overview of a Smart Card System

Figure 1 is a use case diagram that gives an overview of the basic components and functions of any smart card system. The use case diagram is a behavioural UML diagram that presents the system functionality. In our system, the actors illustrated in the figure represent the main components of the system, which are the User, Smart Card, Smart

Card Reader, Client, Server, and Database. The use cases represent the functions or services that take place while the system is operating. The focus of the analysis in this study will be on the functions of three main components, which are the User, Smart Card, and the Smart Card Reader.

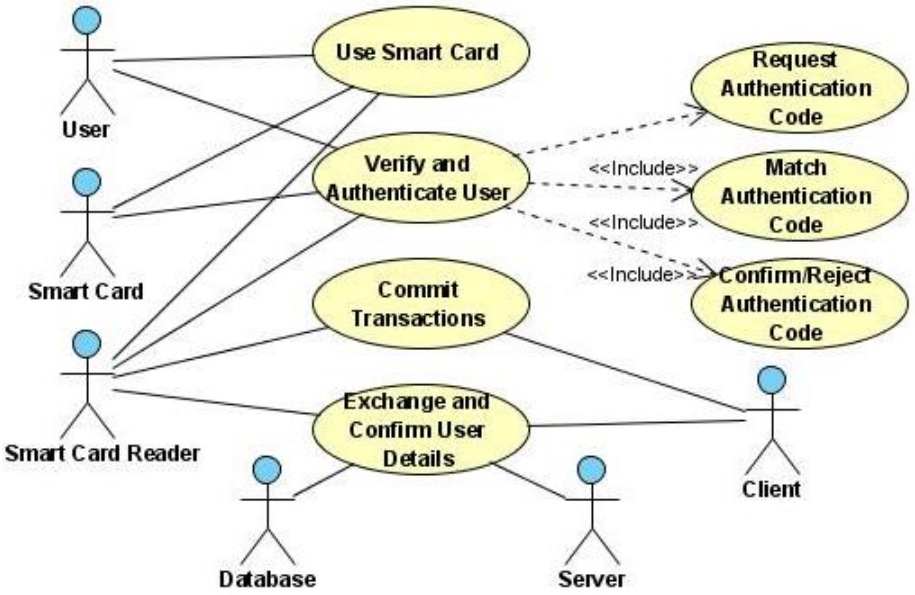

Fig. 1 Overview of a Smart Card System

The system combines three security mechanisms and a smart card ("what the user has"). The mechanisms are: PIN, Biometrics, and PKI. The first two mechanisms are responsible for user identification and verification, a PIN is: "what the user knows", and the biometrics are: "who the user is". PKI verifies the devices in the system.

When the User decides to use the Smart Card, the first step is to insert the Smart Card in the Smart Card Reader. The Smart Card Reader has number of jobs: it has to verify and authenticate the User and Smart Card, commit transactions, and exchange and confirm the User details with the other system components. To be able to demonstrate the transactions of the system, another type of UML diagram has to be used, Figure 2. The following sections describe the registration phase and the verification phase of the smart card system and the potential threats and attacks.

3.2. Smart Card Registration System

To be able to demonstrate the transactions and message sequence between the smart card system objects, a sequence diagram is used, e.g. Figure 2, which is a behavioural diagram that shows the interactions of system processes.

The User provides the required information along with the biometric evidence. The system then saves the User details in the Smart Card and captures the fingerprint, which is the biometric method used in the proposed design, and produces a template that is

stored in the system and the Smart Card. Then, the Registration System requests a PIN from the User to be used in future verification processes.

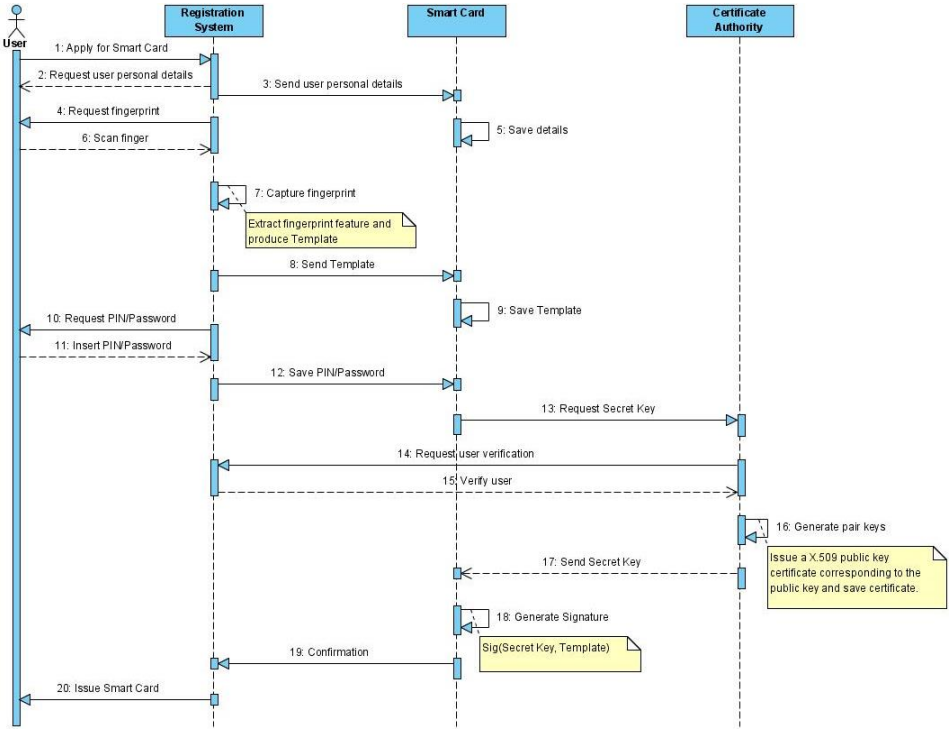

Fig. 2 Registration Phase in PIN, Biometrics (Fingerprint), and PKI Smart Card System

The PIN is stored in the Smart Card for future verification. Finally, the smart card system requests a private key from the Certificate Authority (CA) to generate a digital signature [13]. The CA, on the other hand, requests User verification from the Registration System, generates a pair of keys for the User. The CA also issues a digital certificate corresponding to the public key, and sends the private key to the Smart Card to generate a digital signature that combines the private key and the biometric template of the User.

3.3. Smart Card System Verification

Figure 3 shows the transactions that take place when the User uses the Smart Card in a security environment that combines PIN, Biometrics, and PKI security methods.

The Sender first inserts the PIN, the Smart Card Reader extracts the stored PIN from the Smart Card and starts the comparison process. If the match is successful the Smart Card Reader will ask for another proof, which is the Sender's fingerprint, otherwise, the transaction will be aborted after allowing the Sender three attempts to enter the PIN. The Sender scans the finger through the Smart Card Reader scanner; the Reader will extract the Sender's biometric feature and produce a template. The matching process will then take place and the result will decide whether the Sender has permission to access the

system or not. If the match is true, the Smart Card releases the Sender's private key. Next, the Sender starts to send a message to the Receiver; the message is going to be digitally signed with the Sender's private key, and the system will request the Receiver's public key from the CA to encrypt the message. The CA will send the digital certificate and the message will be encrypted using both the Sender's private key and the Receiver's pubic key, therefore, the digital envelope is now ready to be sent securely to the Receiver. Finally, the Receiver will send a request to the CA to get the Sender's public key to decrypt the message. Again, using both the Sender's public key and the Receiver's private key the Receiver will be able to decrypt the message successfully.

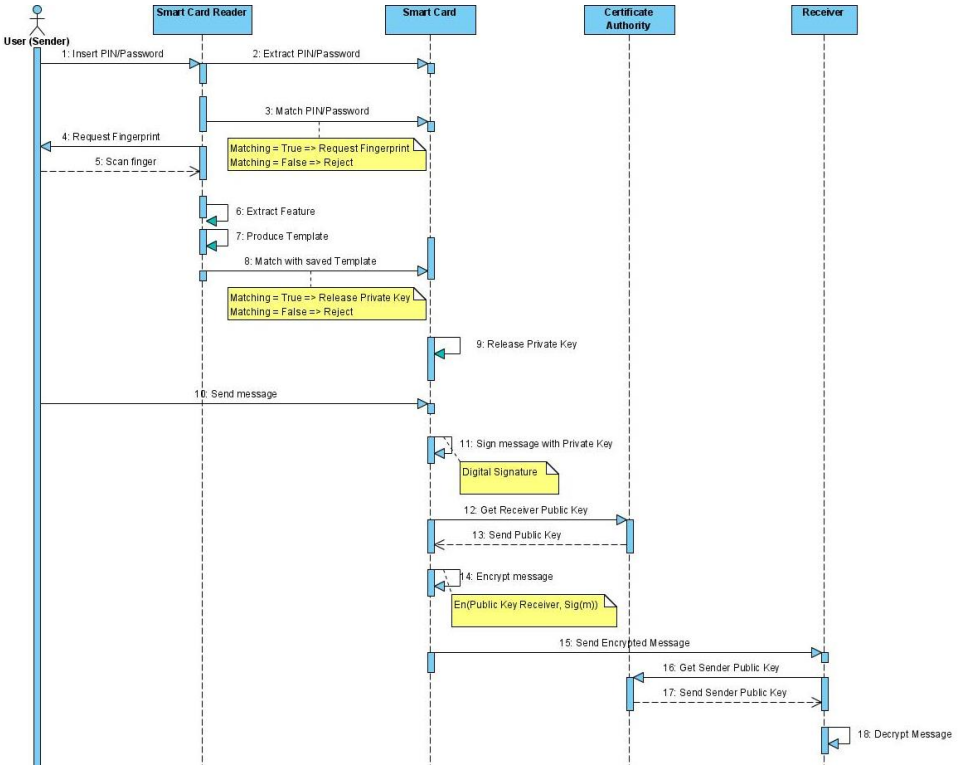

Fig. 3 Verification Processes in PIN, Biometrics (Fingerprint), and PKI Smart Card System

These security methods should achieve the security goals of confidentiality, integrity, authentication, and non-repudiation. However, each mechanism has its pros and cons. For example, fingerprints have disadvantages: How can we know that the biometric provided is not subject to misuse? If the User was clever and powerful enough to fool the system and use a false fingerprint, then the system will be breached and an intruder will have access to the real User's credentials and privileges. The PKI method has its disadvantages as well. If one breach takes place during the transaction the Sender and the Receiver can both suffer security loss.

3.4. Smart Card System Threats

Threats are the possible means by which a security policy may be breached [14]. A threat source can be any person, thing, event, or idea that poses danger to an asset within a system in terms of confidentiality, integrity, availability, or legitimate use. Moreover, threats can be deliberate or accidental [14]. If deliberate, a threat can be categorized as *passive*, such as network sniffing, or *active*, such as negligence, errors, attempt to gain unauthorized access to the system, or changing the value of a particular transaction by malicious persons. Therefore, possible threats on a smart card system include unauthorized system access, hacking and system intrusion, information leakage or theft, integrity violation (errors and omissions by insiders or outsiders), distributed denial of service, illegitimate use (dishonest or disgruntled insiders or outsiders), system penetration and tampering. Threat sources have different motivations that may lead to various attacks on any government or business information system; therefore, the parties involved in the smart card system must be familiar with the human threat environments and their different motivations.

3.5. Possible attacks on a Smart Card System

Attacks may occur at every single stage of a product's lifecycle, starting from the development stage, the manufacturing stage, and ending up with actual usage. Attacks that take place at the development stage and the manufacturing stage of a smart card are most likely to be carried out by an insider, [15]. Attacks during the smart card use stage can be physical or logical [15]. Physical attacks may manipulate the semiconductor itself and usually require equipment like microscopes, focused ion beams, etc. [16]. Side-channel attacks consist of observing behaviour while the information is being processed and include timing analysis and power analysis [17].

In contrast, logical attacks or so-called software attacks do not attack the hardware properties directly; they are more focused on the communication and flow of information between the smart card and the terminal [15]. Attackers can write malicious software, that can be employed in a software attack on a smart card, for example, in smart cards that support Java Card it is possible to load and run software. Examples of logical attacks could be bug exploits, illegal bytecode, and attacks during PIN comparison.

Other types of attacks take place during the authentication phase of the smart card system, where the user identity is authenticated using different types of authentication mechanisms like biometrics [18].

3.6. Modelling attacks using UMLsec

After using UML diagrams to express the smart card system protocol and processes, and to represent the transactions that take place while messages are exchanged during the registration and verification processes, in addition to knowing where the areas are that could be vulnerable to attacks, it is also essential to test the model against possible attacks. UMLsec was used to model attacks, using stereotypes such as secrecy and secure information flow along with their tags and constraints. An adversary type in UMLsec can have a function called Threat that allows the adversary to commit delete, read, and insert attacks. Nevertheless, the model is still static and not executable.

4. ANIMATING THE MODEL USING SYSTEMC TLM

SystemC was developed to support the need for a language that improves the overall productivity for designers in the electronic systems field [9]. It supports the development of complex systems by the design and verification of hardware system components at a high level of abstraction. The SystemC library is open source and written in C++. In addition, it contains a lightweight kernel that schedules the processes.

The SystemC library provides concurrent and hierarchical modules, ports, channels, processes, and clocks. Large designs are always broken down hierarchically to be able to manage complexity; structural decomposition of the simulated model in SystemC is specified with modules. The module is the smallest container with state, behaviour, and structure for hierarchical connectivity [9]. Within a module, we use a thread process, which is associated with its own thread of execution. Once the thread starts executing it is in complete control of the simulation until it chooses to return control to the simulator. Hence, the thread process is used to model sequential behaviour [9]. SystemC has two ways to pass control to the simulator again, one way is to exit by (return), in this case the thread is totally stopped, the other way is by having a (wait), therefore, every thread contains an infinite loop and usually has at least one wait function.

The TLM library is built on top of SystemC and allows abstract communications to be modelled in a structured manner. In TLM communication between components is modelled by channels and transaction requests, which are implemented by calling interface functions of the channel models [12]. The initiator port and the target port are distinguished in TLM. An initiator is a module that creates new transactions and passes them on by calling a method of one of the core interfaces. The target is a module that receives the transactions from the initiator. A system component can be an initiator, a target, or an interconnect. The interconnect module accesses a transaction but does not act as an initiator or a target for that transaction, for example routers can be interconnect modules in a system. Another important element in TLM is the generic payload, which allows data abstraction.

4.1. Smart Card System Simulation

The executable model produced in our work shows the sequence of transactions that occur in the smart card system while the smart card is used; they correspond to the transactions in Figure 3.

Hence, in the executable model, the smart card system objects and their related transactions, the lifelines in the UML diagram, are represented as objects – modules in SystemC, and the arrows are represented as TLM transactions. The modules have two types of socket, an initiator socket that is responsible for sending the transactions and a target socket that is responsible for receiving the transactions; both sockets are defined in the module structure. The Sender module communicates with the Smart Card module and the Smart Card Reader module. An initiator socket from the Sender to the Smart Card is created, along with another initiator socket to the Smart Card Reader module, to allow the Sender to send transactions to both modules. The initiator is responsible for calling the transport function to send the payload to the target socket. On the other hand, a target socket is created and then registered in the constructor; the target socket receives the payload from the transfer function for processing and response.

The next step is creating the threads that correspond to the processes taking place in each module, creating the payloads that are transferred from a module to the other, creating functions, and setting events and variables. In the smart card executable model, the authentication methods used are PIN and biometrics. The user, modelled as part of the Sender module, enters the PIN. If the PIN is correct, the user enters the fingerprint. The number of attempts allowed for the Sender is programmable. The executable model counts the number of attempts, and compares the inserted PIN and fingerprint with the saved PIN and fingerprint template in the smart card. Also, there is a time limit for inserting the PIN and fingerprint, otherwise a timeout message will appear. If the number of incorrect attempts exceeded the limit, the system blocks the smart card and saves the smart card ID in the banned smart card list. Errors in entering the correct PIN vary; it could be wrong digits, taking a long time to insert the correct PIN, or an attacker trying to insert the PIN randomly.

The same steps take place when entering a fingerprint. The successful attempts at PIN and fingerprint entry will confirm that the Sender is a legitimate user. Therefore, when the Sender passes the authentication step, the smart card releases the private key. Then the transactions related to signing the message with the private and public keys take place, and finally the system sends the digitally signed message to the Receiver. In reality, the User enters the PIN and scans the fingerprint through an input device like a keypad, biometric scanner, or touch pad. However, our executable model can randomise the PIN and fingerprint entries, and also randomise the correct and incorrect time. A simple pseudo-random number generator is used to randomise the PIN and fingerprint entries along with randomising the correct and incorrect time in seconds. The simple random number generator is fast and provides better randomness properties like adjusting the ratios, changing the range of sample smart cards to be tested, and modifying the probabilities of failure. An arbitrary ratio of successful PIN and fingerprint is used; it can be modified to allow flexibility in testing different probabilities of failure.

The executable module has the smart card system objects and their related transactions. The lifelines in the UML diagram are represented as objects, modules in SystemC, and the arrows are represented as transactions using TLM. The transitions in the output correspond to the transaction number in the UML diagram in Figure 3. Obviously, the designer can observe the attempts to enter the right PIN and Biometric along with the required timing. This allows the testing of the effectiveness of the authentication methods used. By running the simulation on different numbers of smart cards with different probabilities of failure it is possible to evaluate the effectiveness of each authentication method.

4.2. Testing the Authentication Methods

Validation of the authentication methods in the smart card system is based on two proposed models. The first model uses a PIN followed by a biometric authentication method, while the second model reverses the sequence. The main reason behind carrying out these correctness tests is to check that the simulation using the executable model is actually working. The purpose of these tests is to verify:

- The functionality/workability of the smart card simulation tool and the availability of test results;
- The reliability of the smart card simulation tool through simulation;

- The degree of flexibility in assigning thresholds and failure probabilities, which will assist in customising the simulation tool based on the industry and sector in which the smart card system will be used;
- The speed of testing, which allows users of the simulation tool to obtain results and manipulate thresholds with ease and flexibility.

The following tests have been performed:
1. PIN followed by biometrics.
2. Biometrics followed by PIN.

For each of these tests, an arbitrary probability of failure has been assigned to each of the authentication methods. For example, the probability of failure for the PIN is set at 15%, for the biometrics (fingerprint) it is set at 10%, and the time allowed for entering the correct pin and correct fingerprint is set at 10 seconds for each. The reason for assuming that the PIN has a slightly higher probability of failure is that the PIN authentication method is weaker than the biometrics and thus there is a higher probability of successful attacks and user errors and mistakes.

The first test (PIN followed by Biometrics) used 100 to 3,000 smart cards. Table 1 displays the results for the authentication method based on the scenarios of potential failure/error.

Table 1 Results from Testing the PIN followed by Biometrics Authentication Method

Remarks	Number of simulated smart cards						
	100	500	1000	1500	2000	2500	3000
good pin decoded	100	500	998	1490	1976	2464	2950
pin incorrect/re-enter correct pin	16	102	207	302	394	493	587
timeout error (pin)	9	58	125	189	257	299	376
good bio decoded	100	500	998	1490	1976	2464	2950
bio incorrect/re-enter correct bio	13	38	82	126	167	200	234
timeout error (bio)	11	58	124	171	236	299	359

An examination of the results may be interpreted according to the industry and sector of use, which dictate the levels of acceptable thresholds and probabilities of failure. Initially, when examining the relationship between the expected and observed results of failure attempts across all sample sizes we are able to confirm that it is a linear relationship and that observed failure attempts are always below the expected range.

In a sample of 3,000 cards, failure attempts are 963 over 30% of the sample size. This failure percentage alerts us to the vulnerability of the system. This entails a low level of acceptance of usage from both parties due to the increased risks represented by the use of this method. Having such a high degree of risk and vulnerability in the system will expose it to numerous additional threats from different sources.

The results of the expected and observed PIN and Biometric failure attempts are listed in Table 2 and recorded as percentage of the total sample size.

Table 2 Percentages of Expected and Observed PIN followed by Biometrics Failure Attempts

Number of Smart Cards	100	500	1000	1500	2000	2500	3000
Percentage Observed (PIN)	8	11	11	11	11	11	11
Percentage Expected (PIN)	15	15	15	15	15	15	15
Percentage Observed (BIO)	8	6	7	7	7	7	7
Percentage Expected (BIO)	10	10	10	10	10	10	10

When comparing the observed PIN failure attempts to the biometrics failure attempts, it is noted that the percentages are 11% and 7%, respectively. Although the difference is relatively small, it indicates that the PIN authentication method requires additional monitoring, particularly in avoiding risks of external threats that pose potential harm against the users and system confidentiality and privacy. Furthermore, under the simulation of 1,000 smart cards, it is noted that two cards have been banned for reaching the maximum attempts of PIN entry. However, as the sample size increases, the number of banned smart cards grows significantly as illustrated in Figure 4.

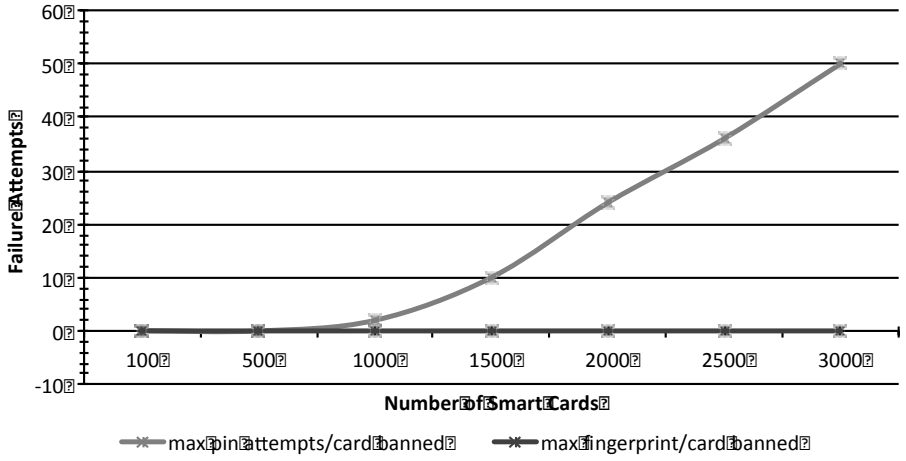

Fig. 4 Smart Cards Banned in PIN and Biometrics Proposed Model

For example when simulating 3000 smart cards, about 50 of them were banned during the PIN authentication step. On the other hand, for the Biometrics authentication method, it is noted that no smart cards have been banned when using this method. This is a clear indication of the level of security that the use of Biometric authentication provides when adopted by smart cards, particularly ones that store and have access to sensitive data.

In the second test, the initial expectation is that the use of a Biometrics authentication first will decrease the possibility of failure attempts and attacks. This mechanism supports the security concept of using something you own (smart card), something you are (Biometrics), and something you know (PIN).

When using the Biometrics authentication method before the PIN, the number of banned smart cards is recorded at 7 and 2 consecutively for a sample size of 3,000 smart cards. This is low compared to when the PIN is used prior to the Biometrics where the number of banned smart cards was 50 and 0 consecutively for a sample size of 3,000. Given the benefits to the user and administrator, as well as the practicality of using the Biometrics and PIN authentication methods across most industries, it is recommended to adopt this method in the given order as it provides better security levels.

In summary, the executable model developed using SystemC TLM allowed the designer to test the proposed models that support a combination of authentication methods; by running simulations on different number of smart cards with different authentication methods and recording the results, the designer can examine the robustness of the proposed models in terms of enhancing security specifically during the phase of authenticating the smart card system users. The simulation tool provided a quick, automated, and flexible environment to test the proposed models, in addition to allowing the designer to observe and modify the transactions whenever changes are required.

Testing the proposed model against physical and logical attacks while the smart card is in use has resulted in giving the attacker the chance to get hold of the users private key, and therefore violating numbers of security properties like authentication, confidentiality, privacy, and integrity. This in essence shows that the system is vulnerable to threats and successful attacks taking place. Yet, to be able to reduce the probability of successful attacks, our approach allows the designer to modify the executable model to test against future attacks.

4.3. Simulating Attacks on Smart Card System

There are different types of attacks that have different probabilities of occurrence and different consequences for the smart card system and its users. Each attack targets different areas of the system and has a specific goal; some attacks violate the smart card system authentication, privacy, and confidentiality like attacks on PIN or attacks on biometrics. Other attacks violate the smart card system integrity, reliability, and even authentication like invasive attacks, side channel attacks, etc. Figure 5 is a UML sequence diagram that demonstrates the types of attacks that may occur in any smart card system, even though safeguards and controls like PIN, Biometrics, and PKI are in place.

The purple callouts represent the types of possible attacks that an attacker can carry out in that area precisely; in addition, the red callouts represent the attacks that are created in the executable model to test the system robustness.

The executable model allows us to simulate an attack on the system. An attack on any part of the system is essentially another transaction inserted into the model. For example, to simulate an attack that allows the attacker to steal the private key released from the smart card object, which is coded as a state machine, an attacker is implemented as a class that can intrude into multiple modules in a thread-safe manner. Thus, a transaction is effectively inserted into the model with one line of code at the appropriate point in the smart card module.

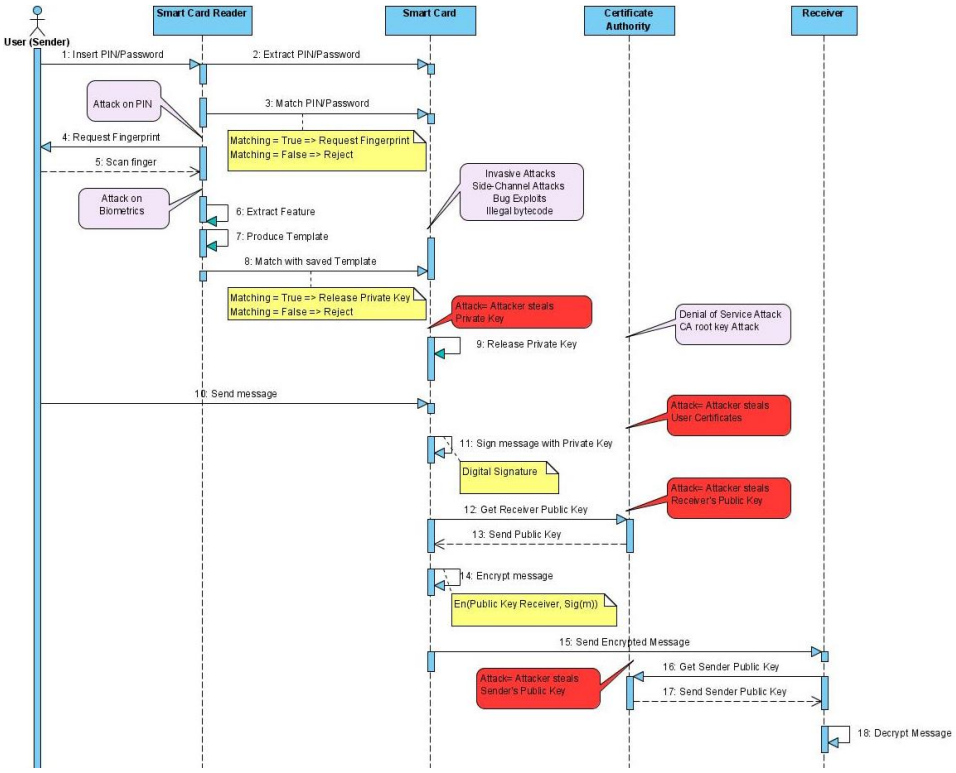

Fig. 5 Possible Attacks on PIN, Biometrics (Fingerprint), and PKI Smart Card System

Now, the model waits for transitions 1 to 8 to occur, and then the attacker interferes and attacks the system after transition 8 where the private key is released, Figure 6.

```
smartcard_reader_object: begin transition 8
smartcard_reader_object: end transition 8
sender_object: end transition 5 Attacker initialized,
@104 s Attacker stole the private key, @104 s
smartcard_object: begin transition 9
smartcard_object: end transition 9
```

Fig. 6 Simulated Private Key Theft

In this example, the attacker has to conduct a physical or logical attack to be able to get hold of the private key. For example, the attacker can practise a successful side channel attack, invasive attack, attacks during PIN comparison, or attacks on Biometrics. The executable model in this study does not simulate the physical or logical attack; it only assumes that a physical or logical attack has taken place. For that reason, it simulates an attack and creates an attacker class with features that allow the attacker to modify the transitions and as a result gain access to the user's secret information, specifically the private key.

Another example of utilising the executable module in attacks simulation is by modelling another sort of an attack, which is carried out on the key exchange operation. This time the attacker monitors the public keys exchanged between the users and the CA, and gets hold of the users' public keys. Being able to interfere with the key exchange protocol opens a door for the attacker to practice attacks that result in network disruption and loss of user trust like for example carrying out a man-in-the-middle attack [19], or a multi-protocol attack [20]. This example focuses on modelling an attack that allows the attacker to interfere through the transactions exchanged between the user and the receiver and gets hold of the data exchanged without both of the users knowing, by being able to model the attack, it is possible to point out a gap in the protocol that allows an attacker to monitor the flow of data, interfere within the transactions, and get hold of the public keys exchanged, Figure 7.

```
smartcard_object: begin transition 13
certificate_authority_object: begin transition 14
certificate_authority_object: end transition 14
Attacker stole the receiver public key, @203 s
smartcard_object: end transition 13
smartcard_object: begin transition 15
smartcard_object: end transition 15
smartcard_object: begin transition 16
smartcard_object: end transition 16
receiver_object: begin transition 17
certificate_authority_object: begin transition 18
certificate_authority_object: end transition 18
Attacker stole the sender public key, @206 s
receiver_object: end transition 17
```

Fig. 7 Simulated Public Key Theft

A Denial of Service (DOS) attack is simulated using the same model. The attack aims at violating the availability property of the system security.

The DOS attack will take place against the Certificate Authority server; the attacker attempts to exhaust the server, which will result in the server being unable to provide the services for legitimate users. The following is part of the DOS attack simulation output:

As the output shows, the transactions of the smart card system are running normally, however, when the DOS attack successfully takes place, the service is denied and the attacker gets hold of the users public keys exchanged among the system objects. In addition, the subsequent transactions failed to occur because the Certificate Authority server is unavailable. This attack shows that the availability property has been violated and the system users will not be able to use their smart cards until the Certificate Authority server recovers from the attack.

DOS attacks are indistinguishable from legitimate sign-in requests. The only differentiation is in the frequency of sign-in attempts and their origin. A large number of sign-in attempts in rapid succession can be indicative of a DOS attack. Hence, smart card systems can be protected from DOS attacks by identifying high frequency of login attempts from a source and denying service to the source of such attack. Another effective way is to limit the number of login attempts a user is allowed at a time.

In summary, the executable model developed using SystemC TLM allowed the designer to test the proposed models that support a combination of authentication

methods; by running simulations on different number of smart cards with different authentication methods and recording the results, the designer can examine the robustness of the proposed models in terms of enhancing security specifically during the phase of authenticating the smart card system users. The simulation tool provided a quick, automated, and flexible environment to test the proposed models, in addition to allowing the designer to observe and modify the transactions whenever changes are required. In addition, the SystemC TLM executable model also allowed the designer to discover the weak points of the system and point out vulnerabilities; the successful attacks indicate that there are weaknesses in the security protocol. To be able to reduce the probability of successful attacks, the designer can modify the executable model to test against future attacks.

In contrast with the UML diagram, the animation makes it possible to see the attack actually happening. Moreover, it is possible to make changes easily within the model and to try a number of attacks to test the system's robustness by simply inserting transactions into the UML diagram, and transforming them into transactions within the SystemC TLM executable model.

5. CONCLUSION

UML diagrams are an excellent way of modelling systems, along with their extensions; they have features that show the designer how things should work. However, UML does not allow the designer to see what happens if something goes wrong with the system. Therefore, to be able to see things happening and give reasons about the system, simulation has to take place. SystemC TLM was used to transform a static UML model into an executable model. The executable model providing the opportunity to see the transaction flow within the system objects in an animated manner. In addition, it allowed the simulation of attacks in different parts of the system. The model gives a clear view of the weaknesses in the security requirements, methods, and protocols used in the smart card system.

REFERENCES

[1] A. Bushager and M. Zwolinski, "Modelling smart card security protocols in SystemC TLM", In: Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on. 2010, pp. 637–643.

[2] S. Schneider, "Security properties and CSP", In: Proceedings IEEE Symposium on Security and Privacy, 1996, pp. 174 –187.

[3] J. McDermott, "Visual security protocol modeling", In: Proceedings of the 2005 workshop on New security paradigms, NSPW '05:. New York, NY, USA: ACM. ISBN 1-59593-317-4; 2005, pp. 97–109.

[4] J. Jürjens, "UMLsec: Extending UML for secure systems development", In: UML 2002 – The Unified Modeling Language. 2002, pp. 412–425.

[5] Object Management Group, Introduction to OMG's Unified Modeling Language™ (UML ®) 2005;URL http://www.omg.org/gettingstarted/what is uml.htm.

[6] J. Jürjens, "Modelling audit security for smart-card payment schemes with UMLsec", In: Proceedings of SEC 2001 – 16th International Conference on Information Security, 2001, pp. 93–108.

[7] J. Jürjens, "Using UMLsec and Goal-Trees for Secure Systems Development", In: Proceedings of the 2002 ACM Symposium on Applied computing. 2002, pp. 1026–1031.

[8] J. Jürjens, J. Schreck, and Y. Yu, "Automated analysis of permission-based security using UMLsec", In: Fundamental Approaches to Software Engineering, 11th International Conference, FASE 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings. 2008, pp. 292–295.

[9] IEEE Standard System C Language Reference Manual. IEEE Std 1666 2005
[10] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Riegerand A. Muehlberger, "High level fault injection for attack simulation in smart cards", In: Proceedings of Asian Test Symposium 2004, pp. 118–121.
[11] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger and A. Muehlberger, "Extended abstract: an environment for design verification of smart card systems using attack simulation in SystemC", In: ACM/IEEE International Conference on Formal Methods and Models for Co-Design, 2005, pp.253–254.
[12] L. Cai, and D. Gajski, "Transaction level modeling: an overview", In: Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. CODES+ISSS '03; New York, NY, USA: ACM. ISBN 1-58113-742-7; 2003, pp. 19–24.
[13] C. Williams, "Configuring enterprise public key infrastructures to permit integrated deployment of signature, encryption and access control systems", In: Military Communications Conference, 2005. MILCOM 2005. IEEE. 2005, pp. 2172 – 2175 Vol. 4.
[14] R.J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley Publishing; 2 ed.; 2008. ISBN 9780470068526.
[15] W. Rankl, "Overview about attacks on smart cards",*Information Security Technical Report* 2003, Vol. 8, pp.67 – 84.
[16] K. Markantonakis, M. Tunstall, G. Hancke, I. Askoxylakis, and K. Mayes, "Attacking smart card systems: Theory and practice",*Information Security Technical Report* 2009,Vol. 14, pp.46 – 56.
[17] K. Baddam, and M. Zwolinski, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure", In: VLSID '07: Proceedings of the 20th International Conference on VLSI Design. Washington, DC, USA: IEEE Computer Society. ISBN 0-7695-2762-0; 2007, pp. 854–862.
[18] X. Leng, "Smart card applications and security. *Information Security Technical Report* 2009, Vol. 14, pp. 36 – 45.
[19] C. Y. Yang, C.C. Leeand S.Y. Hsiao, "Man-in-the-middle attack on the authentication of the user from the remote autonomous object". *International Journal of Network Security*, 2005, pp.81–83.
[20] A. M. Johnston and P.S. Gemmell, "Authenticated key exchange provably secure against the man-in-the-middle attack". *Journal of Cryptology*, 2002, pp.139–148.