2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

Modelling Smart Card Security Protocols in SystemC TLM

Aisha Bushager and Mark Zwolinski
School of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
Email:{afb05r,mz}@ecs.soton.ac.uk

***Abstract*—Smart cards are an example of advanced chip technology. They allow information transfer between the card holder and the system over secure networks, but they contain sensitive data related to both the card holder and the system, that has to be kept private and confidential. The objective of this work is to create an executable model of a smart card system, including the security protocols and transactions, and to examine the strengths and determine the weaknesses by running tests on the model. The security objectives have to be considered during the early stages of systems development and design; an executable model will give the designer the advantage of exploring the vulnerabilities early, and therefore enhancing the system security. The Unified Modeling Language (UML) 2.0 is used to model the smart card security protocol. The executable model is programmed in SystemC with the Transaction Level Modeling (TLM) extensions. The final model was used to examine the effectiveness of a number of authentication mechanisms with different probabilities of failure. In addition, a number of probable attacks on the current security protocol were modeled to examine the vulnerabilities. The executable model shows that the smart card system security protocols and transactions need further improvement to withstand different types of security attacks.**

Keywords*—*Smart Cards; Modelling Languages; SystemC; Transaction Level Modelling

I. INTRODUCTION

In our digital era, smart cards are a central piece of the wireless revolution. They have entered our wallets as a highly secure key to services that are essential to our daily interaction with the digital world. A smart card allows information transfer between the card holder and a system over secure networks; it contains sensitive data related to both the card holder and the system that has to be kept private and confidential. Therefore, security has to be considered as a key requirement during the early stages of systems development.

The objective of this work is to create an executable model of a smart card system, including the security protocols and transactions, to allow examination of the strengths and weaknesses by executing tests on the model.

The Unified Modelling Language (UML) version 2.0 has been widely used to model smart card security protocols. For example, UMLsec [1] is an extension to UML for integrating security related information into UML specifications by specifying security requirements through stereotypes, tagged values, and constraints.

On the other hand, the models produced by UML are static. In this work we have developed executable models in SystemC, which is a set of C++ classes that provide an event-driven simulation kernel. SystemC is a system level modelling language; it enables design and verification at the system level, independent of any detailed hardware and software implementation. On top of SystemC, Transaction Level Modelling (TLM) is used to model the transactions of the smart card system.

II. 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 development stages, modelling languages and techniques are used to better visualise 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 specialised 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.

UML can model both the static structure and the dynamic behaviour of the system [4]. To support UML for secure systems development, an extension called UMLsec has been proposed, [1], [5]. UMLsec uses a combination of use-case driven processes with a goal directed approach. The three main mechanisms of the extension are stereotypes, tags, and constraints [6]. Stereotypes and tags are used to create and present the security requirements and assumptions, constraints may be attached but they should be satisfied by modelling elements with the related stereotype [5]. 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) [1], it was also used to define security permissions that enforce restrictions on the workflows of a system [7].

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 [8]. SystemC has been used to produce a methodology to simulate security attacks on smart cards with fault injection [9] and it has also been used to create an environment for design verification of smart cards using security attack simulation [10]. In TLM, communication among computation components is modelled by channels and transaction requests go on by calling interface functions of these channel models [11].

III. Smart Card System Security

Because the smart cards are used to store sensitive data such as PINs, passwords, and keys; the main purpose of an attack is to get hold of these data. Attackers might perform various numbers and styles of attacks on the smart card system.

A. *Smart Card System Threats*

Threats are the possible means by which a security policy may be breached [12]. 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 [12]. If deliberate, a threat can be categorised as passive, such as network sniffing, or active, such as negligence, errors, attempt to gain unauthorised access to the system, or changing the value of a particular transaction by malicious persons. Therefore, possible threats on the smart card system include unauthorised 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 carrying out 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.

B. *Possible Attacks on a Smart Card System*

Security is a huge matter; it covers every single stage of a products 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 carried out by an insider, [13] . Attacks during the smart card use stage can be physical or logical [13]. Physical attacks may manipulate the semiconductor itself and usually require equipment like microscopes, focused ion beams, etc. [14]. Side-channel attacks consist of observing behaviour while the information is being processed and include timing analysis and power analysis [15].

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 [13]. 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 [16].

IV. Using UML to Model Smart Card Transactions

A robust and secure smart card system requires an optimal selection of policies, procedures, protocols, architecture, technology, and staff. To have a better idea of the smart card system and its components, operations, applications, data and information, and security mechanisms, we use a number of UML diagrams for illustration.

A. *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.

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. The following sections describe the registration phase and the verification phase of the smart card system.

B. *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; it is a behavioural diagram that shows the interactions of system processes.

Figure 2 shows the enrolment process that is the main part of the Registration System. The model uses a combination of PINs and biometrics to enhance the verification process. In addition, the Public Key Infrastructure (PKI) is employed to fulfil the system security requirements. PKI is considered to be one of the most comprehensive and secure schemes of passing information from one point to the other. It uses a trusted third party for implementing key life-cycle management processes.

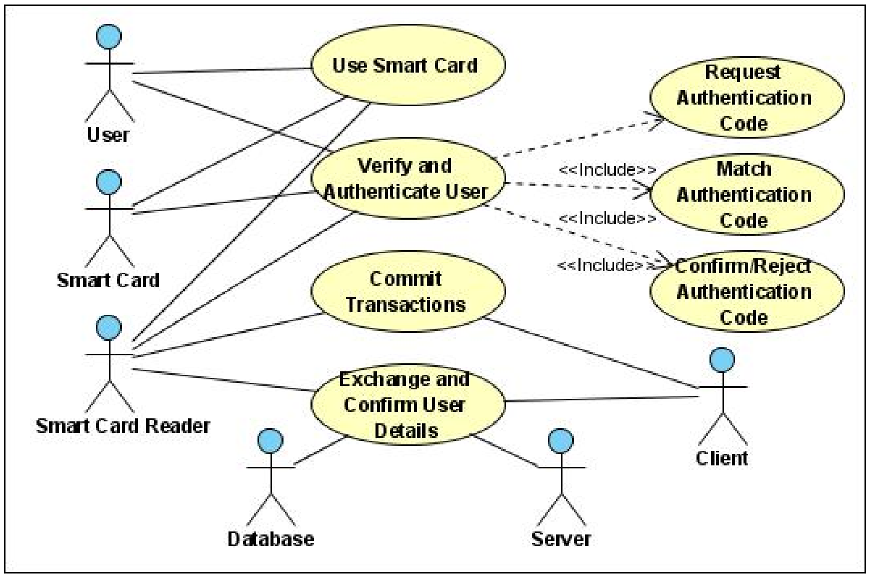
User
Smart Card
Smart Card Reader
Database
Server
Client
Use Smart Card
Verify and Authenticate User
Commit Transactions
Exchange and Confirm User Details
Request Authentication Code
Match Authentication Code
Confirm/Reject Authentication Code
<<Include>>
<<Include>>
<<Include>>

Fig. 1. Overview of a Smart Card System.

This third party is called the Certificate Authority (CA), which validates the identity of the user and issues digital certificates [17].

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.

The PIN is stored in the Smart Card for future verification. Finally, the smart card system requests a private key from the CA to generate a digital signature. 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.

C. 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 User 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 Users fingerprint, otherwise, the transaction will be aborted after allowing the User three attempts to enter the PIN. The User scans the finger through the Smart Card Reader scanner; the Reader will extract the User's biometric feature and produce a template. The matching process will then take place and the result will decide whether the User has the permission to access the system or not. If the match was true, the Smart Card releases the User's private key. Next, the User starts to send a message to the Receiver; the message is going to be digitally signed with the User'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 User'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.

These security methods should achieve the security goals of confidentiality, integrity, authentication, and non-repudiation. However, each mechanism has its pros and cons, the possible attacks that might take place are shown in Figure 3. 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.

D. 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 are the areas 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

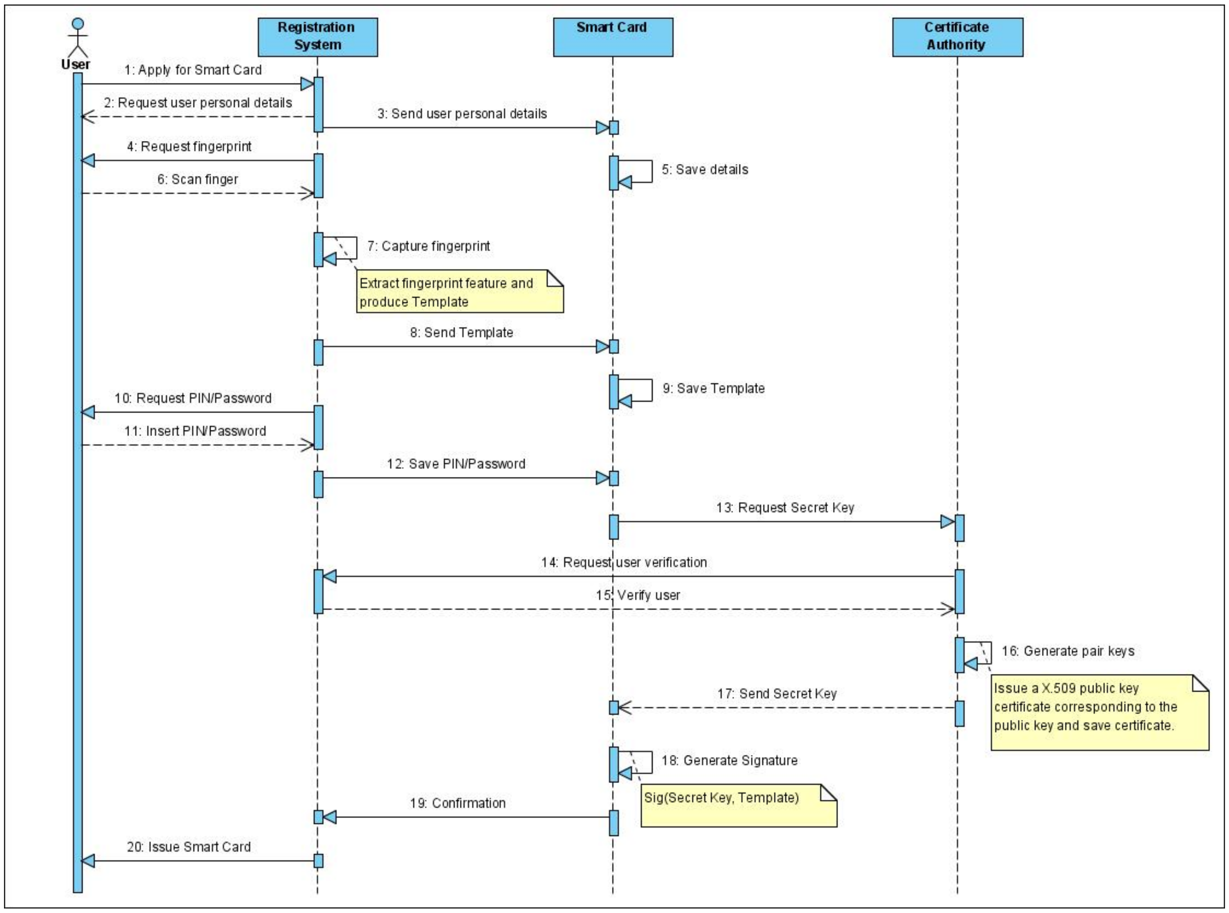

Fig. 2. Registration Phase in PIN, Biometrics (Fingerprint), and PKI Smart Card System.

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. Even by writing these notations down, the model is still static and not executable.

As a result, UMLsec did not automate the model because it is a specification language that has the ability of expressing the system protocols and transactions but not automating them. Therefore, SystemC TLM was used to transform the static model into an executable model.

V. Animating the Model Using SystemC TLM

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, the construct SC_MODULE is used to represent a module [8]. In our work, SC_THREAD is used – a thread process 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 [8]. 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.

In our model, the smart card system objects are programmed as SystemC modules, and the transactions among these modules are modelled using TLM. In TLM, transactions are implemented by function calls.

A. 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. The following is part of the simulation output produced:

```
sender_object:////////////////////////
sender_object:// Card count: 100
sender_object:// Card ID: 611
sender_object:// Pin entered: correct
sender_object:// entry duration: 1 s
sender_object:////////////////////////
```

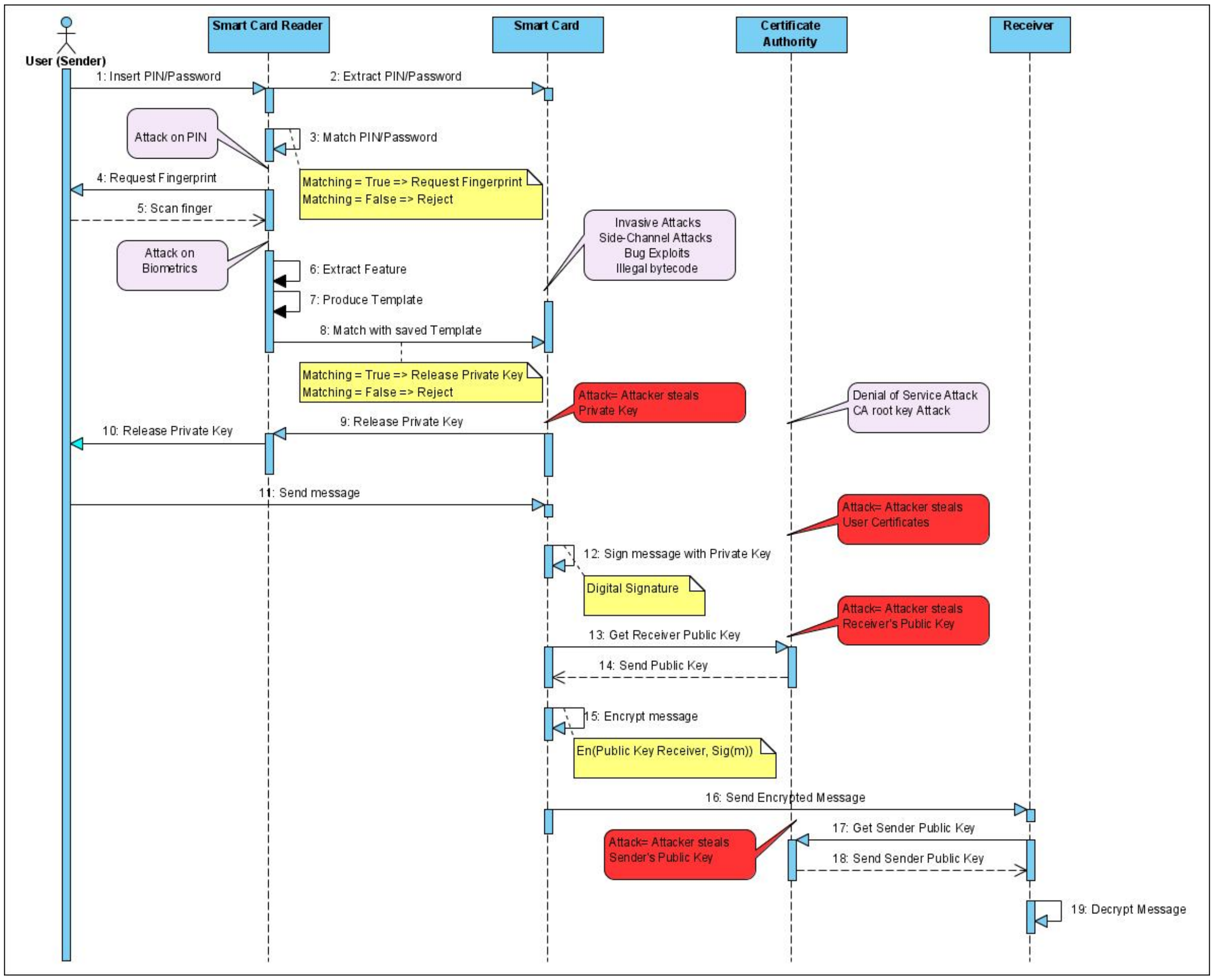
User (Sender)
Smart Card Reader
Smart Card
Certificate Authority
Receiver
1: Insert PIN/Password
2: Extract PIN/Password
Attack on PIN
3: Match PIN/Password
4: Request Fingerprint
Matching = True => Request Fingerprint
Matching = False => Reject
5: Scan finger
Invasive Attacks
Side-Channel Attacks
Bug Exploits
Illegal bytecode
Attack on Biometrics
6: Extract Feature
7: Produce Template
8: Match with saved Template
Matching = True => Release Private Key
Matching = False => Reject
Attack= Attacker steals Private Key
Denial of Service Attack
CA root key Attack
9: Release Private Key
10: Release Private Key
11: Send message
Attack= Attacker steals User Certificates
12: Sign message with Private Key
Digital Signature
Attack= Attacker steals Receiver's Public Key
13: Get Receiver Public Key
14: Send Public Key
15: Encrypt message
En(Public Key Receiver, Sig(m))
16: Send Encrypted Message
17: Get Sender Public Key
Attack= Attacker steals Sender's Public Key
18: Send Sender Public Key
19: Decrypt Message

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

```
sender_object: begin transition 1
smartcard_reader_object: begin transition 2
smartcard_reader_object: end transition 2
smartcard_reader_object: begin transition 3
smartcard_reader_object: Good pin
smartcard_reader_object: end transition 3
sender_object: end transition 1
sender_object: *** good pin decoded ***
smartcard_reader_object: begin transition 4
smartcard_reader_object: end transition 4
sender_object://///////////////////////
sender_object:// Card count: 100
sender_object:// Card ID: 611
sender_object:// Fingerprint entered: correct
sender_object:// Entry duration: 5 s
sender_object://///////////////////////
sender_object: begin transition 5
smartcard_reader_object: begin transition 6
smartcard_reader_object: end transition 6
smartcard_reader_object: begin transition 7
smartcard_reader_object: end transition 7
smartcard_reader_object: begin transition 8
smartcard_reader_object: end transition 8
sender_object: end transition 5
smartcard_object: begin transition 9
smartcard_reader_object: begin transition 10
smartcard_reader_object: end transition 10
smartcard_object: end transition 9
sender_object: begin transition 11
sender_object: end transition 11
smartcard_object: begin transition 12
smartcard_object: end transition 12
smartcard_object: begin transition 13
smartcard_object: end transition 13
cert_authority_object: begin transition 14
cert_authority_object: end transition 14
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
receiver_object: end transition 17
cert_authority_object: begin transition 18
cert_authority_object: end transition 18
receiver_object: begin transition 19
receiver_object: end transition 19
```

The executable module shows the smart card system objects and their related transactions. The lifelines in the UML diagram are represented as objects, called 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. 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.

B. Simulating Attacks on Smart Card System

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 by inserting the following line of code at the appropriate point in the smart card module:

```
attack::
getInstance().set_private_key(private_key);
```

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.

```
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
```

The simulation shows that the attacker gets hold of the private key by practising a successful attack on the smart card object. Attacking the key exchange operation violates the privacy, authentication, and integrity properties of the system. Also, it compromises the security of the User, which may result in identity theft, information leackage, or message alteration. Being able to steal the private key points out a vulnerability within the security protocol employed in the system.

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:

```
iteration: 4 at time: 2 s 194
insert_pin_password 201
extract_pin_password 294
request_fingerprint 305
match_with_saved_templates 324
release_private_key 428
release_private_key 335
send message 217
get_receiver_public_key 438
send_encrypted_message 451
Attacker stole the sender public key,
SERVICE DENIED, @2 s 561
```

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.

In summary, the executable model developed using SystemC TLM allowed the designer to discover the weak points of the system, the successful attacks indicate that there are weaknesses in the security protocol. A number of security properties like authentication, privacy, integrity, and availability have been violated, which shows that the system is vulnerable to attacks. 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.

VI. 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] J. Jürjens, "Modelling audit security for smart-card payment schemes with UMLsec," in *Trusted Information: The New Decade Challenge*, M. Dupuy and P. Paradinas, Eds., Paris, 11–13 June 2001, pp. 93–108, proceedings of SEC 2001 – 16th International Conference on Information Security.

[2] S. Schneider, "Security properties and CSP," in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, 6-8 1996, pp. 174 –187.

[3] J. McDermott, "Visual security protocol modeling," in *NSPW '05: Proceedings of the 2005 workshop on New security paradigms*. New York, NY, USA: ACM, 2005, pp. 97–109.

[4] Object Management Group, "Introduction to OMG's Unified Modeling Language^{TM}(UML®)." [Online]. Available: http://www.omg.org/gettingstarted/what_is_uml.htm

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

[6] ——, "Using UMLsec and Goal-Trees for Secure Systems Development," in *Proceedings of the 2002 ACM symposium on Applied computing*, pp. 1026–1031.

[7] 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, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, 2008, pp. 292–295.

[8] "IEEE Standard System C Language Reference Manual," *IEEE Std 1666-2005*, pp. 0_1 –423, 2006.

[9] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger, and A. Muehlberger, "High level fault injection for attack simulation in smart cards," *Asian Test Symposium*, vol. 0, pp. 118–121, 2004.

[10] ——, "Extended abstract: an environment for design verification of smart card systems using attack simulation in SystemC," *Formal Methods and Models for Co-Design, ACM/IEEE International Conference on*, vol. 0, pp. 253–254, 2005.

[11] 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*, ser. CODES+ISSS '03. New York, NY, USA: ACM, 2003, pp. 19–24.

[12] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. Wiley Publishing, 2008.

[13] W. Rankl, "Overview about attacks on smart cards," *Information Security Technical Report*, vol. 8, no. 1, pp. 67 – 84, 2003.

[14] K. Markantonakis, M. Tunstall, G. Hancke, I. Askoxylakis, and K. Mayes, "Attacking smart card systems: Theory and practice," *Information Security Technical Report*, vol. 14, no. 2, pp. 46 – 56, 2009, smart Card Applications and Security.

[15] 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 held jointly with 6th International Conference*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 854–862.

[16] X. Leng, "Smart card applications and security," *Information Security Technical Report*, vol. 14, no. 2, pp. 36 – 45, 2009, smart Card Applications and Security.

[17] 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*, 17-20 2005, pp. 2172 – 2175 Vol. 4.