Formal Verification of a Security Protocol

By
Shizra Sultan
NUST201260846MSEECS63012F

Supervisor
Dr. Abdul Ghafoor
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer and Communication Security (MS CCS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan

(December 2014)

Approval

It is certified that the contents and form of the thesis entitled **“Formal Verification of a Security Protocol"** submitted by **Shizra Sultan** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Abdul Ghafoor**
Signature:
Date:

Committee Member 1: Dr. Awais Shibli
Signature:
Date:

Committee Member 2: Dr. Osman Hassan
Signature:
Date:

Committee Member 3: Ms. Rahat Masood
Signature:
Date:

Dedication

To my father for his endless love, support and encouragement,

To my husband for his affection, patience and believe in my work

To my daughter Musfira and niece Mehraal for their presence in my life

&

To my mother for everything, all that I have ever accomplished is because of you

Abstract

Acceleration in Smartphones technology is overpowering everything; it has ascended as a prerequisite for every other technology that has come or yet to come, every utility is been revamping itself as a compatibility to smartphones, it is becoming a one-for-all device that handles routine life matters education, health, shopping everything. One of the essential and critical human is daily routine financial transactions involving sellers, buyers, third parties and most notably our money. Many protocols are designed for mobile platforms to deal with the financial transactions which involve hardware tokens like credit cards which are not secure anymore. Growth of smartphones also leads to increased vulnerabilities if not properly tested which raises a big question about the defence capability of smartphones to protect user's data. In this paper we propose a secure payment protocol for smartphones to take care of transactions involved in daily routine without using any hardware token. It involves bank as a transparent entity but seller and buyer customarily rely on a payment gateway to mark a successful transaction. The suggested protocol uses symmetric keys, certificates, and two-factor authentication to make protocol safe and to prove the secrecy and authentication properties the protocol is formally verified by AVISPA.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Shizra Sultan
Signature:

Acknowledgment

For the sake **<u>of Allah, the Most Gracious and the Most Merciful</u>**, all praises to Allah for the qualities and His blessings in concluding this research.

I am extremely thankful to my supervisor ***Dr. Abdul Ghafoor*** for his supervision and consistent backing. His constant encouragement and recommendations thorough out the degree have helped me accomplished so much.

My acknowledgement additionally goes to my GEC members ***Dr. Awais Shibli, Dr. Osman Hassan and Ms. Rahat Masood*** for their help and guidance during my time with them

To wrap things up, my gratitude goes to my colleague ***Naveed Ahmed*** and all other class fellows and ***KTH-AIS lab members*** for their assistance with many things that have benefitted me a lot

Shizra Sultan, 2014

Table of Contents

List of Figures

Chapter 1

Introduction

It is the first chapter of this book and presents a brief introduction of Security protocols and Formal methods and how both of them can be integrated to generate desired results (REFINE)

1.1 Security protocols

Internet was evolved in a world without predators. In the beginning, it was just meant to share information among certain number of people, providing them with fast accessible information; however, it had rapidly progressed from a facility to a vital necessity and it went on to such an extent that presently world seems to be fully dependant on the internet. With such an extreme outreach, internet where attracted brilliance to nurture itself, it also enticed malicious agents who exploited its in-built vulnerabilities, giving rise to arena of Security. Security is difficult to define in a single standard as the term is different for different people. From data breach to communication monitoring, from data corruption to identifying theft, all are major threats and widespread of all these played a motivational role in development of network security.

1.1.1 Fundamentals of Network Security

Network security can be characterized into four different categories i.e. Subjects, Objects, Access Control Policy and Flow Control Policy. Entities that can access objects are known as Subjects while entities that need to be protected are Objects. Access polices define how any subject can access any object whereas Flow Control Policy controls the information stream between subjects and objects. Main security threats that any network can face are classified further based on the following aspects:

I. **Confidentiality:** Protecting the object against unintended assess and granting access only to the authorized parties

II. **Authentication:** Verifying the identity of a user/user device logging into a network or to verify the integrity of data stored/transmitted

III. **Authorization:** Determining the user privileges or access rights to resources related to information security and computer security

IV. **Integrity:** Preventing the information from being modified by unauthorized parties, preserving trustworthiness, origin, completeness, and correctness of information

Considering above notions, different protocols and systems are built to provide required security to users as well as the data and assurance about their functional correctness is not only critical but highly desirable. Therefore, one must cater all the possible threats on any system before launching it. But it is not possible due to unlimited number of threats. In short, there must be a way to ensure whether a system is secure or not.

1.1.2 Attacks on Security Protocols

Security protocols are meant for establishing an authenticated interactive communication. Such communication protocols target their goals despite the hostile activities of interference of attackers with the protocol. With this assertion, we start our thesis exploring how to verify the security primitives of any protocol properly.

Usually attackers access public channels used by protocol actors. So, security protocols aim to conceal information from such unauthorized parties. In other words, secure protocols give guarantee of one actor about the identity of another actor with which it is communicating. Cryptographic security protocols are used to permit actors or subjects to converse securely over an insecure network. Even though security protocols are generally short, they mostly have subtle flaws which make protocols vulnerable and attack prone

1.2 Formal Methods

Formal methods refer to a variety of mathematical modeling techniques that are applicable to computer system design. Such techniques include activities like system specification, specification analysis and proof, transformational development, program verification etc. In formal verification, mathematical techniques are used to ensure that a design conforms to some precisely expressed notion of functional correctness. As a matter of fact, the absence of formal methods for verification could lead to security errors remaining undetected. Thus, these formal verification techniques provide a systematic way of discovering protocol flaws.

During the two last decades, formal methods have demonstrated their usefulness while designing and analyzing the security protocols. In this era, they have become a must-have tool for designers to manage verification complexity. They, indeed, provide rigorous frameworks and techniques that allow discovering new flaws. Up till now, many flaws in security protocols have been revealed while trying to prove the security of protocol in a formal setting. Following the seminal work of Dolev and Yao, many techniques have been developed for analyzing the security of protocols, often automatically. For instance, the AVISPA platform and the ProVerif tool are two efficient and practical tools for automatically proving security properties or finding bugs, if any. The security of protocols is decidable in general; however, checking the secrecy and authentication-like properties is NP-complete, when the number of sessions is fixed.

While the symbolic approaches were successful in finding attacks, the security proofs in these models are questionable because of the level of abstraction that is most of the cryptographic details are ignored. This might create a problem, for instance, a protocol can be proved in a formal, symbolic, model, while there is an attack, which also exploits some finer details of the actual implementation of encryption scheme. In contrast, cryptographic models are more accurate, the security of protocols is based on the security of the underlying primitives, which in turn is proved assuming the hardness of various computational tasks such as factoring or computing discrete logarithms. The messages are bit strings. The proofs in the computational model imply strong guarantees (security holds in the presence of an arbitrary probabilistic polynomial-time adversary). However, security reductions for even moderately-sized protocols become extremely long, difficult, and tedious.

1.3 Security protocols and Formal Methods

1.3.1 Need of Verification in Security Protocols

Security protocols ensure security and integrity of data in communications; therefore, the maturity of protocol verification methods is highly desirable. Despite the simplicity of security protocols, unfortunately, authentication of correct implementation of such protocols is quite difficult and even security experts experience such complications. As a matter of fact, security protocol problems are not just related to the strength of cryptographic algorithms employed, the way cryptography is used, is equally important.

While designing a novel security protocol, it is essential to consider all the possible behaviors of hypothetical attackers including violations of the protocol rules and any possible forgery of messages. The figure of such behaviors is usually large because an attacker can forge and inject a new message at each protocol step in a number of ways that is typically unbounded. This fact adds extra complexity to those concurrent interactions which a communication protocol must normally manage. Therefore, knowingly the existence of best practices and recommendations, manual design of a novel security protocol has been very error-prone and challenging task. Such a situation has been faced while defining Needham-Schroeder public-key protocol and interestingly the protocol was believed secure for 17 years before Lowe discovered a simple flaw that compromised the NS protocol.

Due to inherent complexity, correct development of security protocols demands rigorous, mathematically based methods for reasoning about their authentication. The research on rigorous methods which have been developed so far, for reasoning out security protocols, can be grouped into two approaches. One stream is based on quite abstract and symbolic modeling, originated from research by Dolev and Yao and developed mainly in the formal methods community. The other one was originated from work by Goldwasser and Micali and by Yao. It is based on more detailed computational models, involving complexity and probability theories. Both approaches have made much progress in the last few years, pushed by a growing interest in secure computing.

The symbolic approach, being more abstract, enables better automation in developing proofs, but its results are more complex in relation to the real world security goals. On the other hand, computational approach, being closer to reality, gives more realistic security assurance at the expense of increase difficulty in proof automation. After the work of Abadi and Rogaway, researchers have been trying to define a relation between these two approaches, either by proving computational soundness for the symbolic model or by applying reasoning techniques that proved successful in the symbolic model to the computational model. An extensive amount of recent progress has been made in this direction.

Both symbolic and computational approaches offer laborious proofs based on abstract models. However, a large gap still exists between these models and a real-world protocol implementation including its execution. This gap may be responsible for final unsatisfactory security levels, even when proofs for authentication have been developed for a model of protocol.

Notoriously, in OpenSSL, an error condition returned by a cryptographic function was incorrectly interpreted by the function caller, making the application to accept corrupted data. Such a fault couldn't be found if a formal model having no relation with the implementation code is analyzed. This is because, the semantics of model itself defines the (correct) interpretation of cryptographic functions results, and the way of code handling return values is neglected.

1.3.2 Security protocols and their vulnerabilities

A security protocol involves two or more communicating actors (also called principals). Each actor plays a protocol role and is usually associated with an identity (e.g. the identity of a human user). Each protocol role defines the rules which actors have to follow. Communication among actors may occur on various types of communication channels (e.g. a public network, dial-up lines, etc.) by exchanging messages. Multiple sessions of the protocol can be maintained in parallel.

In addition to honest actors, there might be dishonest actors or attackers aiming to subvert the protocol, i.e. preventing protocol sessions from achieving their security goals. Off course, attackers are not constrained to follow the protocol rules. These attackers can be broadly divided into two classes; passive attackers and active attackers. Passive attackers can just intercept record and analyze protocol messages whereas active attackers can interfere with the protocol by altering, deleting, redirecting and reordering protocol messages, as well as by forging and inserting new protocol messages into the conversation.

As discussed in [GS97b], [Car94] and [PPS11], attacks on security protocols can also be categorized depending on the weaknesses they exploit. For completeness, a non-exhaustive list of the most common types of attacks is discussed below in detail.

Attacks based on cryptographic flaws are those that try to break ideal cryptographic properties, hence, exploiting weaknesses of the cryptographic algorithms. Analyzing an encrypted message to extrapolate the secret key is an example of a cryptographic flaw attacks. Such attacks can be refined to take into account collateral information leaked by a cryptographic primitive or by its particular usage within a security protocol. For example, the measure of power consumed or the time taken to encrypt a message with different secret keys can be used to infer some information on the secret keys themselves.

Other attacks exploit the absence of some operation which is crucial to guarantee a security property. This kind of weakness falls within the so-called internal action aw class and may be due to protocol design errors or implementation mistakes. This type of vulnerability was found in Three Pass Protocol definition where absence of a check on message received in third phase of protocol enables the attack.

Similar attacks which are based on the absence of proper message type checking are known as type aw attacks. In this kind, the attacker sends a protocol actor a message of different type than what is expected, and the actor fails to detect the type mismatch, thus misinterpreting the message contents or behaving in an unexpected way. Such vulnerability can be found in the Otway-Rees protocol.

When some actor is not able to distinguish between a fresh message and an old message that is being re-used, then replay attacks (i.e. attacks exploiting freshness flaws) are possible. Needham-Schroeder secret key protocol has this type of vulnerability in which the attacker can force server and client to re-use an old secret key that the attacker may already have conceded in a 1st session. Some attacks, like the one possible in case of Needham-Schroeder public-key protocol, are called man-in-the-middle attacks because the attacker stands in the middle of two honest actors and breaks the protocol while relaying messages from one actor to the other.

Some of the vulnerabilities that are most difficult to spot are those related to possible bad interactions among multiple sessions of the same protocol. For example, an attacker could be able to use a protocol actor as an oracle to get some information that attacker could not generate on its own. Then, this information can be used by attacker to forge new messages that get injected into another parallel protocol session. This is oracle attack; Three Pass Protocol is vulnerable to this type of attack.

As already discussed, the vulnerabilities exploited by attackers may depend on logical flaws or ambiguities in the protocol specification itself or on divergences between the protocol specification and its implementation, caused by programming errors. The code that implements a protocol may also be affected by some other more general errors (buffer overflows, memory leakages, and so on). Such errors may not only affect the security of implemented protocol itself but also disturb the security of that system where protocol is executed. For this reason, verifying that implementation code does not diverge significantly from the intended model and that it is not affected by critical bugs is as important as verifying the security properties of protocol specification.

1.4 Verification of Protocols

Cryptographic protocols are used to certify some security properties. So, verification and authentication of such protocols is highly recommended. The objective of this research thesis is to prove formally that a certain protocol validates a given property. Security properties that can be of interest in any protocol can be the following.

1.4.1 Security Properties

- ***Secrecy-*** concerns a message send or received by the protocol. These messages are generally a nonce or a secret key that should not come to be public at the end of the protocol conversation.
- ***Authentication-*** proves that you are indeed the person or the system you claim to be
- ***Non-Repudiation-*** certifies that the owner of a message cannot later claim not to be the owner.
- ***Fairness-*** confirms that one of the parties cannot end the protocol part-way through and gain some unfair advantages over the other party
- ***Anonymity-*** guarantees that the identity of an agent is protected with respect to the message that he sent.

In real implementations of security protocols, unfortunately, it is difficult to perform formal proofs to give assurance about the application code that implements the protocol logic. For that reason, the perfect cryptography hypothesis is generally used to simplify these proofs. While creating that hypothesis, the encryption schemes are supposed to be perfect to have strong hypothesis. Therefore, it is impossible to deduce a plain-text from its cipher-text without knowing the decryption key. Even with this hypothesis, idealized protocols may contain some logical flaws.

1.5 Smartphone Applications

In recent years, the world of technology has experienced developments more rapidly than it has in any other era before. Because of advancements in information and communications technology, people can now instantly communicate with others thousands of miles away, pursue a career at home, know what is happening on the other side of the world within minutes, and find the answer to any question with just one click or tap. Out of all of the wonderful new tools that technology has brought about, however, none has been more widely impactful than the smartphone.

Smartphones are progressively leading the world of mobile phones. The advent of mobile data has enabled internet on the move and now, as the smartphone becomes ever more affordable, it is rapidly reaching the masses. This has widely been lauded as positive by telecoms operators and even by the users because it is convenient, efficient, and cost-effective. Its portability and versatility have made it one of the most popular electronic devices in the market.

Smartphone applications or "Apps" are developed to make the lives easier in many ways. There are almost 300,000 app out there that can be accessed at any time and from anywhere. However, smartphones are posing one of the biggest risks to business's information security.

1.6 Conclusion

In this chapter, we have presented an introduction of Security Protocols highlighting the need of security protocols, aspects on the basis of which such protocols are defined, implemented and verified. We have also discussed in detail the possible vulnerabilities in security protocols and the need of verification based on different security properties.

We have concluded that security protocols use cryptographic primitives as building blocks to achieve security goals such as authentication, confidentiality, integrity etc. However, these protocols are still error-prone and are needed to be verified. To validate the correctness of security protocols, automated formal methods are used to discover the imperfections in protocols.

In the next chapters, our investigation about the related work done in support of formal verification of security protocols has been documented followed by our research methodology.

Chapter 2

Research Methodology

"When researchers first begin to open up any new line of enquiry there will be no useful theories available from which to deduce propositions for testing. Knowledge has to begin with collecting facts and then trying to find some order in them. This is known as induction. Deduction is the technique by which knowledge develops in more mature fields of enquiry. It involves a sort of logical leap. Going a stage further than the theory, data is then collected to test it. "

(Marshall, 1997:17)

Quality Research is started with identifying a problem worth working for, then setting goals and objectives of study, going through enough literature to have a sufficient state-of-the-art knowledge, choosing the correct research approach and selecting methods that will be effective in answering the research question. All stages are very important in research like choosing the right research method to approach the research problem, if right method is not used then it can take a lot of time to get to the desired results. There are two main approaches of research methodology: Inductive approach and Deductive approach

2.1 Inductive approach

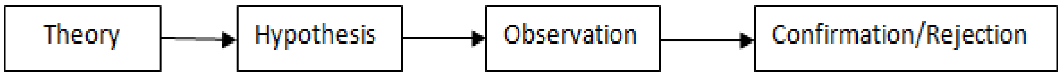
This approach deals with detailed observations and abstract ideas. Begins with a topic, then gradually a researcher develops empirical generalizations, finds initial relationships and then derive a theory from them. Inductive approach is a "bottom-up" approach; scientist uses observations to construct an abstraction or to define a picture of the phenomenon that is being studied. No known theories or patterns need to be tested during the research process.

$$\begin{tikzcd}
\boxed{\text{Observations}} \arrow[r] & \boxed{\text{Pattern}} \arrow[r] & \boxed{\text{Theory}}
\end{tikzcd}$$

2.2 Deductive approach

This approach involves developing a hypothesis built on an existing theory, and then creates a research approach to investigate the hypothesis. In other words, deductive approach is associated with deducting conclusions from propositions. Generally it has following stages

1. ***Inferring*** hypothesis from theory
2. ***Formulating*** hypothesis in functioning terms
3. ***Investigating*** hypothesis with the solicitation of relevant method(s)
4. ***Examining*** the outcome of the investigation, and thus confirming or rejecting the theory.
5. ***Modifying*** theory in instances when hypothesis is not confirmed.



In our research we adopted the deductive approach, started the research process by identifying the research problem, derived a hypothesis from it, conducted different tests to investigate the hypothesis, and concluded different observations conforming our theory

2.2.1 Theory
First we chose a domain, Verification of security protocols for financial applications, identified several issues by doing a thorough literature survey. Survey was in two stages; first we did a survey of different payment protocols, their strength and weaknesses, which works better in which conditions, what are the most needed security parameters for any financial transaction and when designing a payment protocol for smartphones what are the main facts that needed to be catered. Second we did a survey on how security protocols can be formally verified, studied different protocols that are already verified by some kind of formal method, identified different tools and techniques and compared their results to infer what the best way to verify a payment protocol.

2.2.2 Hypothesis
Constructed on our theory, we formulated a following hypothesis

"Formal verification of a security protocol gives a high assurance about its functional correctness and its security parameters"

We have proved this hypothesis during our research. We have designed a secure payment protocol for smartphones and then verified it via formal technique known as Model Checking with two security parameters; secrecy and authentication, at all steps of protocols. End results support our hypothesis

2.2.3 Observations
Different observations were made

- When designing any security protocol for smartphones, low cryptographic computations should be chose for better performance

- It is important to choose the right formal technique for any protocols. There is not one standard to verify all properties via one formal technique,
- First identify which properties we need to verify in our system and then chose the verification technique accordingly

2.2.4 Confirmation

Proposed protocol has been verified by a tool name AVISPA.it is a security protocol analyzer which supports all cryptographic operations and is widely accepted. It verifies the protocol by mode checking explores every possible attack state and confirms the secrecy and authentication properties of a protocol. When we tested the mentioned properties of our protocol, it gave the desired results conforming our hypothesis.

2.3 Explicit Contribution

In our research we have tried to propose a secure protocol whose properties are formally verified and made two research contributions

2.3.1 Conceptual Paper

In our paper titled **"Secure protocol for Financial Transactions using Smartphones- SPFT",** authors S.Sultan, A.Ghafoor, A.Shibli, A.Nasir, published in *Proceedings of the* 11th International Conference on Security and Cryptography (SECRYPT 2014), Aug 2014, Vienna, Austria, we have proposed a **S**ecure **P**rotocol for **F**inancial **T**ransactions **SPFT**- based on smart phones. All transactions are performed by smartphone and a user does not have to carry cash or cards. SPFT is a smartphone based payment system for close network. Now -a- days smart phones have become a daily life necessity, there are all those applications which have made phones a single unit to handle most of the chores like utility bills, online financial transactions e.g., what we propose is to make it more convenient that it eliminates the use of paper money in a close environment (cafeteria bills, buying/selling, pays) like big organizations. An application that can control daily life small or big transactions just with few clicks and does not have to keep trail of paper money on daily basis. The protocol is based on conventional actors' i.e., customer, merchant, payment gateway & a financial intuition. What is different is the kind of access these parties have with each other, bank is always thought to be a close system, in this protocol it's not. All parties will be completely transparent; there will be no transaction trails and anonymity will be preserved. It fulfills all the security parameters required in a payment protocol like secrecy, authentication and conflict resolution and to prove this we have formally tested the code by AVISPA. It is an automation tool to validate security protocols. Protocols that need to be verified against properties like (secrecy, authentication, proof of origin etc.) are written in a specification language HLPSL. AVISPA at back-end works on principles of formal methods like model checking to achieve security goals and exemplify threat models. It covers four back-end practices; OFMC (on the fly model- checker), CL-AtSe (attack searcher), SATMC (SAT model checker) and TA4MC (automata based protocol analyzer). We have tested the proposed protocol with all four techniques

2.3.2 Journal Paper

We are writing another paper which is an extension to this protocol. Smartcards are acting as client in addition to smartphones while server side is the same for both and then this integrated solution is formally verified.

2.4 Result Validation

Validation is a risky part of the research process. It ensures that careful research is being done and that genuine results are being formed, published and promoted. For computation-based science, this comprises of open and fair analysis of the proposed models by already verified and accepted tools and techniques which actually relates to some truth or physical reality.one of the main objectives achieved by this research is the emphasis on the verification. Lot of protocols and systems are implemented in real world without proper verification which is very unsafe especially if it's for some critical application. We have verified our protocol by a widely accepted tool which validates the results by four different methods

2.5 Conclusion

Different research methods relate to different situations, and henceforth it is essential to know which method is best to practice with a specific hypothesis or question. In fact, if an unfitting research method is used, it could harm the research process. Research methods are used to determine through neutral observation and analysis the functional correctness of any system. Quality research in field of real life applications can benefit everyone in society.

Chapter 3

Literature Survey

3.1 Overview

The progression in the Internet and the World Wide Web has opened the door to a range of data transfer options and security issues. In the beginning, security was not a major concern because most of the enterprise critical software applications ran on mainframes. However, with the rise in enterprise local area networks (LANs) more data became readily available to even more users, and internal network security became a concern. Similarly, major internal and external security risks also emerged in case of financial services with the help of mobile telecommunication devices. It continued till security of financial transactions became the most complicated challenges that need to be addressed. Modern research in this domain reveals that even if a security protocol design is sound, the implementation may be flawed. So, defined-level of confidence is to be provided concerning the security of cryptographic protocols.

For mitigate the issue discussed above, formal methods are considered effective for verifying or falsifying the security of cryptographic protocols. It is actually problematic for those who design or use such protocols. Formally specifying the security protocol avoids interpretation errors, and in principle enables implementations to be derived, thus, reducing the probability of introducing mistakes. Later sections of this chapter will elaborate this discussion further.

3.2 Security Issues of Financial Protocols

3.2.1 Background

Over the years technology has covered every aspect of human life and has transformed into a utility. Aim of technology is to facilitate humans as much as possible so it’s moving towards integrating real life critical areas such as education, health, finance and many others with emerging technologies. There has been so much overlapping among various fields of IT that one cannot clearly demarcate the boundary of any technology. So now when we talk about throughput, efficiency and security of any system we just can’t look at one component, we have to take in account the share of all modules involved in a finished product. We need to utilize different technologies in a way that they combine to give a better product. For example online banking on Smart phones; both mentioned technologies have benefits and problems of their own so we efficiently incorporate both to gain as much throughput as we can. On one hand it benefits the users; and on the other hand there have been evil elements involved which provide a greater harm by exploiting the vulnerabilities of such systems. Financial indiscretion in e-commerce is becoming a major concern for individual users as well as for the organizations worldwide. Cyber criminals are gradually launching well-organized and effective attacks by exploiting the vulnerabilities in existing architectures. Credit card information is being hijacked, financial institutions are being attacked and money is being stolen online by cyber thieves without even entering the bank [8].Taking in account all the above facts, there is a need of a secure e-commerce solution which does not only facilitate users' financial needs but also fulfils the security parameters compulsory in any transaction. To do that we have to accommodate many different problems like mobility and ease of access for users so we suggest a financial solution based on smart phones. Conventional financial solution lack extensibility, openness, privacy, and cost effectiveness. We realize that Smart phones are prone to attacks too so if we want to use them for financial transactions, we need a highly efficient and secure design. Here is a brief literature review for different financial protocols and their strengths and weaknesses

3.2.2 Related Work

Recent years have seen a phenomenal level of growth in the development of wireless communication technologies; however, it has also raised questions about the performance and security of online payment systems. This is because wireless systems have less bandwidth, low reliability and higher latencies. Moreover, mobile devices have some limitations too that is less computational power and less storage space. In such a scenario, high computational operations cannot be performed like PKI encryptions/decryptions. Most payment protocols are projected for fixed networks. PKI- storage to keep certificates, CA to verify certs so additional communication. It is also noted that some protocols, for instance, SET & iKP don't show a very sound compatibility with wireless systems. Therefore, a reliable payment protocol is needed which uses less computational resources & storage, ensures message confidentiality & integrity, party authentication, non-repudiations etc. and fulfils all transaction security properties of payment system.

[Kungpisdan & Srinivasan, 2004] A protocol was proposed to cater the above limitations in which they used symmetric keys and MAC. AKE protocol was used for key generation & distribution. The protocol was composed of two sub-protocols; *merchant registration* protocol and *payment* protocol. According to it, in the first phase, merchant & client know each other; client gets itself registered to merchant and its issuer while merchant registers itself to payment gateway. Client sends a *payment request* to merchant which contains order information and *Value subtraction request* which is to be forwarded to Issuer. Merchant then sends a *Value-claim request* which contains the *Value subtraction request* and the transaction is preceded based on further approval/rejection by the payment gateway. In this protocol, party authentication was ensured by symmetric encryption and the secret Y (secret shared between client & issuer i.e. credit card info), transaction privacy was confirmed by symmetric encryption, transaction integrity was guaranteed by MAC and non-repudiation of transactions was ensured by Y. In fact, merchant was able to provide non-repudiable evidence proving to other parties that the client had originated the message. Authors had proved that above protocol caters non-repudiation via KSL logic.

Considering the security of mobile payment systems, 3P trust issues regarding privacy & non-repudiation cannot be overlooked. Customer has to bind its identity to the purchase request, revealing its identity to merchant. On the other hand, transaction data casts its trail from the customer on the way to merchant. Generally, conventional protocols lack extensibility, openness & cost effectiveness. To benefit the existing financial infrastructure, all the entities must be taken as atomic elements in the system. Trust reliance on financial service providers must be reduced; certificate authorities & time stamping provision should be introduced in transaction process. However, an evidence and mechanism must be provided to resolve the dispute.

[Liu & Liao, 2005] A protocol has been recommended as an extension of SEMOPS which has six elements; customer, merchant, bank, 3P, time stamping server & data center. User shops and merchant sends him the product information which user combines with his account information, signs it and forwards to customer payment processor (CPP). CPP validates the signature and forwards to merchant via merchant bank processor for verification and then routes the confirmation notification to CPP via data center reserving the time stamps and related info to avert the disowning of receipt. Authors have taken care of the privacy and non-repudiation of the proposed protocol. Former property is reasoned with the user information reserved at user payment processor and is not travelled with the transaction messages while later property is catered with the introduction of certificate authorities in a system.

In today's Internet focused world, most of the protocols "assume" that merchant has internet access all the time. In comparison to it, other modes of communication (like SMS, call) are less used mainly due to their high cost. Another fact is that the traditional asymmetric signatures chosen for authentication are computationally expensive for mobile devices. For a network connection, there can be several mobility & connectivity scenarios. For instance, full connectivity, disconnected interaction, server centric case, client centric case, Kiosk centric case (payment gateway) etc. Since protocols work on the principle that all the entities in a system have internet access all the time, a new protocol is needed to be designed which caters the problem, when merchant cannot directly connect with the client. To use less computational resources, asymmetric signing is avoided and payment gateways are trusted.

[Isaac and Camara, year] Resolving the problem discussed, a protocol is proposed which is sub-divided into two; merchant registration protocol & payment protocol. Client and merchant shares a session key via AKE protocol and then client registers to merchant by downloading a wallet software which contains key generation and payment software. Client sends a payment request to merchant including his nick name, order information and value subtraction request encrypted with a session key already shared. Merchant decrypts the request and sends it to payment gateway which then sends the *value claim request* to client. On confirmation and verification by all the three parties, transfers the amount from client's to merchant's account. In the analysis of this protocol, authors have proved *entity authentication and transaction privacy* with the use of symmetric key and *anonymity* is ensured by usage of nick names instead of the real names of users and trust relationships are ensured by use of credit cards.

Financial transactions are one of the most high-risk activities performed online and tracking the money flow, banks have complete control over these financial transactions. Mobile based payments have been a very hot topic in financial industry and in such payments, bank and phone operators are the two key players. New methods of financial transactions are required to facilitate millions of customers who want to use mobile based payment systems. Those methods are needed which can provide simplicity & usability, interoperability, security & trust, cross border payments, market understanding etc.

[A.Vilmos and S.Karnouskos, year] A solution is proposed which is built on credit push concept. Merchant provides customer with specific data that can identify the merchant & particular transaction, client remains anonymous throughout the process. Customer receives the data, combines with his information, authorizes it and sends to the payment processor (can be a bank or MNO). It verifies the customer and forwards it to the data center which identifies the merchant's bank and sends the payment notification to merchant's payment processor. On confirmation, customer is notified via data center and customer's payment processor releases the funds it has kept for the purchase. This protocol is designed to be a standard technology worldwide, that is why it is evaluated from different perspectives like change of bank, MNOs, network technology, infrastructure, openness of transactions, flexibility etc.

Currently, mobile payment systems involve a large number of computations which in turn consumes a lot of mobile's resources. Apart, security in case of micro, macro & higher payment amount is also questionable. Whereas large use of hardware tokens might also create problems for payment solutions while the service monopoly is reduced. Therefore, light & secure mobile payment system is required with less startup costs & maintenance, anonymity of all the entities and with low cryptographic operations.

[Badway and Aboud, 2012] In addition to SEMOPS, there are two protocols; SIP (session initiation protocol) & ECC (elliptic curve cryptography). Of these two, SIP is meant for reducing the cost; however, ECC is responsible for less cryptographic operations. In SIP enhanced SEMOPS, SIP sessions are established between associated parties. 1st SIP session is established between merchant & customer to exchange transaction data, 2nd SIP session involves customer and customer payment processor (CPP) to generate a payment request and sends it to data center (DC) to process it. It is then forwarded to merchant payment processor (MPP) in session 3. In the last session, merchant payment processor communicates with merchant for the approval of transaction request. The merchant reply is delivered to CPP through MPP and DC in order to issue the transaction amount from the customer bank account. For signing, they have used ECC instead of RSA accomplishing higher security with smaller key size. They have used PIN, nonces & OTPs for mutual authentication, session keys for privacy, PKI for integrity & non- repudiation. A comparison of proposed protocol is given with the number of computations of both RSA & ECC.

Nowadays, mobile payment methods are progressing as a universal payment system; therefore, such methods are needed to be very secure & efficient in their operations. In the going age, studying different mobile communication scenarios and merging the developments into one framework would be beneficial in developing a new mobile payment method which is more secure, flexible and convenient.

[Xueming and Nan , 2009] A security mobile payment protocol is suggested which is built on studying the 3-D secure protocol. This proposed protocol uses the encryption scheme uniting the symmetric and asymmetric encryption and signature scheme. Client in 1st step sends his nickname and transaction request to the merchant who, in reply, sends his identity & transaction identity. Client then generates a payment request, merchant receives the request and send the particular information of client, and his order & issuer to the payment gateway and then funds are transferred after verification. Authors have proved their protocol with SVO logic and have done the security analysis of the protocol with regard to anonymity, non-repudiation, confidentiality & integrity.

3.2.3 Analysis

The performance analysis of SET, iKP and their protocol based on cryptographic operations shows that the protocol uses less computational resources. However, security issues of bank (Issuer) private network are not catered, despite the fact that protocol depends on the secret, Issuer shares with the client. Non-repudiation & computational power of mobile device are majorly focused in the above mentioned work. Secrecy and authentication of the users and merchant are not paid much attention and proofs are also not specified satisfying the CIA. We have also analyzed that more entities are not introduced, in order to reduce the trust dependencies on payment processor; a vulnerability in itself. Assuming the registration has been done initially, user authentication procedure is not elaborated. So, as a case, if attacker registers falsely, rest of the security aspects can be compromised. Another way is to assign a set of nick names to user, using which they will communicate with the merchant and be anonymous. But if several transactions are done, nick names can be mapped and associated to a user. Usually, banks have all the information about who bought what form whom and at what cost, therefore, if the bank is compromised, privacy will be highly compromised. A generic solution is also proposed which hasn't been evaluated from security perspectives that should have been the top priority. Above all, it seems a theoretical debate; there is no mathematical or symbolic design presented in support of it.
It is also investigated that a complicated protocol may use less number of cryptographic operations but the existence of a lot of messages overhead is uncertain for establishing different sessions. Merchant centric connectivity issue is also targeted; however, the solution is not found to be that much merchant centric.

3.3 Formal Methods and Security Protocols

3.3.1 Background

Formal methods are by far the most trusted scientific method to validate the functional correctness of hardware and software specifications. Security protocols are now also verified for their different properties by different methods, here is a brief literature review of security protocol verification via formal methods

3.3.2 Related Work

Working on the authentication of security protocols, BAN [Burrows, et al., 1989] aimed to formalize the assumption made from the messages received by an agent, without considering an intruder on the network. But somehow it was not capable of identifying a wide variety of protocol flaws. [Brackin 1996] An extension to this work, had addressed its weaknesses but these new methods were not sufficient to deal with the common problems of protocol verification.

Some methods follow state exploration method in which all the possible traces constitute a protocol. Such methods are powerful and automatic. [Syverson, et al., 2000] As a matter of fact, whenever there is an attack, messages would be in valid protocol form only. [Rusinowitch and Turuani 2001] This could happen if a polynomial sized attack occurs that is the case of limited sessions.

[Dolev and Yao 1983] A formal method of the intruder had been presented for verifying security protocols but it had only considered the confidentiality authentication with some cryptographic aspects.

[Lowe 1995] Communicating Sequential Processes (CSP) had also been used for protocol verification, in which a CSP model of each entity was written in the system including agents participating in the protocol, the Dolev-Yao spy and communicating network. This CSP theory of traces had helped a lot to analyze the security protocols.

[Lowe 1996] NSPK protocol had been modified for small number of participants (at least) into a secure protocol. Failures Divergences Refinement (FDR) had been used as a model checker to verify CSP programs but even then it'd been very time consuming.

[Lowe 1998] Alice and Bob picture of security protocol had been discussed in terms of a CSP model, which is appropriate for Failures Divergences Refinement; however, there was still the need of special-purpose tools to (possibly) verify large security protocols.

[Basin 1999] While analyzing these protocols, corresponding characteristics of Paulson's formalism and model checking were combined into a different approach. Data types were built to compute and represent the infinite data without evaluating their arguments. Later on, a trace-based interleaving semantics were used to model security protocols. In such a case, protocols are formalized as infinite trees with traces as their branches. A tree child node reflects a trace which is supposed to capture the accomplishment of a protocol step or a Dolev-Yao intruder's act. Hence, if any protocol is implemented, it means a boundless tree is defined and a property of nodes in the tree corresponds to a security property. However, some deviation from these security properties is also observed while searching an infinite tree in a lazy fashion. In case of large branching factor, dividing the search tree and its reordering was done in the way the tree was searched. Hence, the first branch consisted of trimmed traces with bogus events like those messages which do not follow the protocol rules. The next heuristic was priority based and assigned the top priority to those events which involve any type of intrusion or another execution of the protocol.

[Basin, 2003] A tool; On-The-Fly Model-Checker (OFMC) was introduced which combined lazy data types with various other techniques, that modeled the steps of a Dolev-Yao spy. Its methods considerably reduced the searching computations including all the intrusions along with finding new attacks. The formal model used for protocol analysis by OFMC tool was based on low-level and high-level protocol specification languages.

[Chevalier and Vigneron 2002] In the AVISPA European project, Constraint-Logic-Based Attack Searcher (CL-Atse) was developed which combined proof of first-order theorem with constrain-logic to cover things like associabilityof operators, commutativiy of operators etc. for demonstrating the messages. Later on, its procedure was modified to speed up the search for intruders and to simplify a variety of optimizations in the deduction rules, reducing and/or eliminating the redundancies and uselessness in the protocol run.

[Bozga 2003] Another tool called HERMES was dedicated to confirm secrecy features. HERMES, as input, takes a protocol and two sets. One set contains the (S) secret messages and other one has (H) hypotheses about secret keys. All possible states for a set of secrets S' and a set of encrypted messages H' are calculated then. Protocol assures the confidentiality for secrets, if all messages in set of secrets are protected under encrypted messages.

[Thayer-Fabrega, et al., 1998]Proposed model checking tools do not target the establishment of a correct protocol, rather, they identify those steps in protocol execution which violate any given security requirement.

[Paulson 1998] The inductive approach was invented to the verification of security protocols combining the idea of concrete notion of events and that of deriving guarantees from each message. According to the approach, a protocol is considered as a group of traces and that trace is defined as a list of events where sending or receiving a message is an event. All the attacks and unintended losses of important information (Oops event) are included in protocol descriptions. It is comprised of four different theories. The first theory called the message theory, has three leading operators where each operator is well-defined on unlimited group messages and is used to model protocol features and its execution. The next theory; event theory models the sending and receiving events, the knowledge of agents and a freshness operator. On the other hand, shared and public theories correspondingly validate shared-key and public-key cryptography.

[Meadows 1996], NRL Protocol Analyzer (NPA); a special-purpose tool was presented for verifying the protocol execution whether it satisfies principal authentication or not. It was a term-rewriting proof procedure according to which the spy aims not only to find out secret but more than that. For instance, it may try to prove that a message has some extra things which it is not having.

[Blanchet 2001] A Prolog-based tool was developed for the same purpose, representing the intruder's actions and the rules of the protocol (modeled as messages known to the intruder). Two abstractions were performed to avoid limiting protocol execution number. The first abstraction recorded only that it has been used. The other one distinguished when it was safe to use a unique term for the nonce linked to a couple of participants, irrespective of number of times the protocol has been executed.

[Weidenbach 1999] The benefits of the inductive method and finite state analysis are combined in Weidenbach's approach. According to it, initially, the protocol itself and its prerequisite are both interpreted into monadic Horn formulae (first-order). Then, these formulae are put into a first-order logic automated theorem prover based on saturation, which is known as SPASS. On SPASS termination, the protocol becomes flaw-free. However, if proof fails, the output can often be used to produce an intruder, irrespective of the fact that the process has not been automated.

[Cohen 2000] A first-order logiic method was introduced with protocol demonstrated as a transition system. In this method, protocol execution is modeled as a set of transitions executed along with the associate messages exchanged. A new value might be created by transitions and this fresh value can be used as either a nonce or a key. The author employed his method on TAPS7 and its results are used for the analysis of 70 other protocols

[Cohen 2003]. To analyze a verification proof, protocol description is used by TAPS to make various other protocols, useful in formation of security properties (secrecy or authentication).

[Steel et al., 2003] Author discussed its developed theorem-prover based tool that was supposed to target a set of protocols (instead of focusing any distinct protocol), with random figure of participants involved in an execution step [Meadows, 2000]. It was based on SPASS and its proof based on consistency contains the characteristics of being refutation complete [Comon et al., 1998]. This completeness outcome ensures the presence of an intruder, if there is any [Steel, 2002].

3.3.3 Analysis

Security protocol verification methods are observed to follow one of these three methodologies; belief logic, state exploration, and theorem proving. It has been noted that early attempts for verifying security protocols were imperfect because mainly the verification of confidentiality was considered.

BAN being simplest belief logic allows abstract proofs of verifying the secrecy properties but it is not capable of recognizing various protocol flaws, assuming all agents are honest. On the other hand, mostly state explosion implementations do not use practical results reducing the entire search space. Such automated methods are more influential. Traces generated to build a counterexample can be easily exploited, if a property does not hold.

In order to cover the weaknesses of state exploration method, Basin's methodology uses lazy data types for modeling the boundless state-space of the protocol. Suppose an attack exists in a trace at a tree node. But locating this point would be difficult in case of tree with infinite depth, yielding a rapidly increasingly branching factor. Hence, with a large branching factor, typical algorithms for searching are less likely be able to find attacks lying at shallow depths. Even OFMC of Basin's approach cannot verify group security protocols.

In Paulson's approach, there is no facility to automatically find attacks on faulty protocols. Besides this, interactive theorem proving is required for verifying protocols for which considerable effort is needed. Therefore, a user should be selective and should supervise the proving process including all the rules to be applied. It has been analyzed that the method described under this approach is a mixture of state exploration and belief logics.

In case of Blanchet's method, if any intrusion is found in the protocol violating the security rules, the tool is capable to identify it, so the protocol representation can be considered safe. False attacks can also be identified assuming the limited number of participants.

In Weidenbachs' approach, although some part of first-order logic is used to have infinite models yet automated theorem proving is needed. According to it, the spy is weaker than Yao's approach. In addition, only two agents are considered so a verification proof needs those meta-theorem proofs which make this constraint flexible. However, this approach is not much detailed in comparison with Paulson's approach.

Since, Cohen's method does not search for attacks, but for proofs, so the counterexamples are not generated. In comparison to Paulson's approach, Cohen's approach requires considerably less user guidance.

3.4 Conclusion

In this chapter, we have discussed the most important tools/methods to formal security protocol verification. After reviewing the literature, we have divided the discussion into three main verification approaches; belief logic, state exploration, and theorem proving. Within the same approach, for each method, we have highlighted the most significant contributions and have come out with a comparison.

In the following chapter, our research methodology outcomes on formal verification of security protocols will be presented in an organized fashion.

Chapter 4

Formal Verification Tool

Teaching to unsuspecting youngsters the effective use of formal methods is one of the joys of life because it is so extremely rewarding.

Edsger Dijkstra

Designing secure protocols is a pressing issue. In real world, for example, the Internet, secure protocols ought to work correctly even under unfavorable conditions e.g., that messages may be listened in or messed around with by an intruder. Serious attacks can be directed even without cryptography breach, yet by abusing shortcomings in the protocols themselves. Normally, these attacks are ignored, as it is troublesome for people, even via cautious protocol assessors. To accelerate the advancement of the coming era of security protocols, and to enhance their security, it is therefore absolutely critical to have tools backing the thorough examination of security protocols by either discovering faults or securing their functional correctness. Ideally, these tools ought to be totally computerized, resilient, expressive, and effectively usable, so they can be coordinated into the protocol development and standardization processes. In last few years many semi-automated tools have been introduced that can analyze different protocols for different parameters, out of which we have selected a security analyzer tool named "AVISPA", as it fitted best to our requirement and is very widely accepted for its certain results

4.1 AVISPA

It is a push button security protocol analyzer abbreviated as "Automated Validation of Internet Security Protocols and Applications" which adapts to present circumstances in an efficient manner by

(i) giving a measured and expressive formal dialect for indicating security protocols and properties, the High-Level Protocol Specification Language HLPSL, and
(ii) Joining several back-end techniques that perform automatic analysis ranging from protocol falsification to generalization-based verification methods for unbounded sessions.

The functional hierarchy of the AVISPA Tool is shown in Fig 4.1. It takes a security protocol as input written in HLPSL and using its syntax and different properties to check cryptographic parameters of the protocol. The HLPSL is an expressive, modular, part based, formal dialect that takes into consideration the particular of control-stream designs, information structures, diverse cryptographic proprietors and their mathematical properties, and complex security properties. The AVISPA Tool consequently interprets (through the HLPSL2IF Translator) a user designed security issue into a detail formalism Intermediate Format IF. IF particulars are therefore given as an input to the backend procedures of the AVISPA Tool, which realize diverse systems to search the infinite-state transition system for states that symbolize attacks on the projected properties of the protocols.

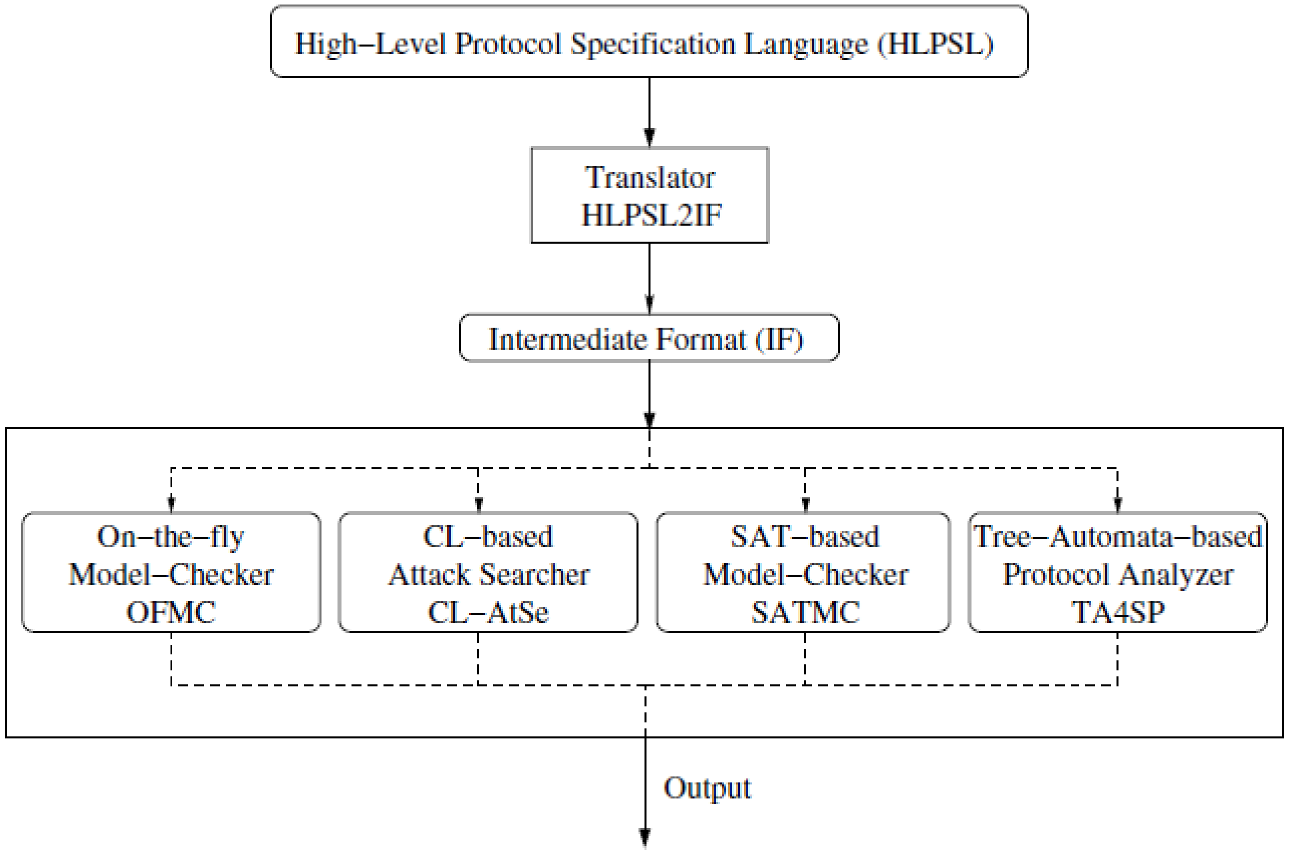
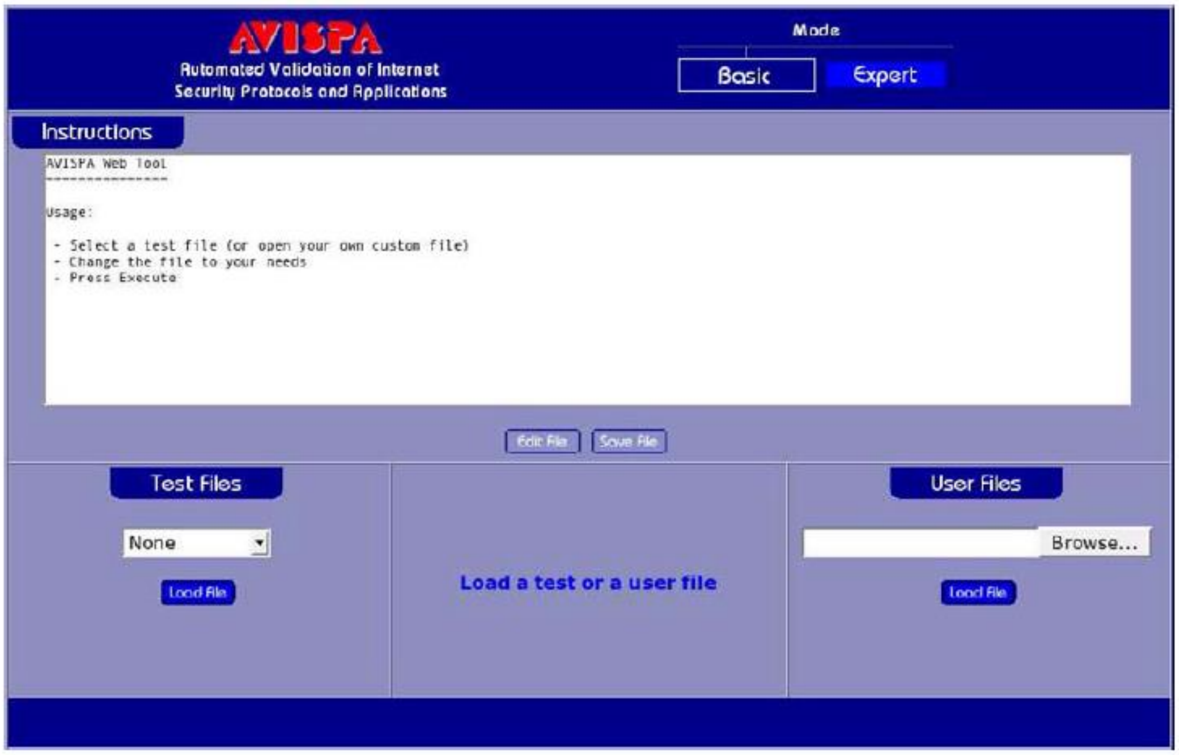

High–Level Protocol Specification Language (HLPSL)
Translator HLPSL2IF
Intermediate Format (IF)
On–the–fly Model–Checker OFMC
CL–based Attack Searcher CL–AtSe
SAT–based Model–Checker SATMC
Tree–Automata–based Protocol Analyzer TA4SP
Output

Figure 1 - Architecture of AVISPA

The recent version of the tool incorporates four back-ends:

I. **On-the-fly Model-Checker OFMC**,
II. **Constraint-Logic-based Attack Searcher CL-AtSe,**
III. **SAT-based Model-Checker SATMC,** and
IV. **TA4SP protocol analyzer,**

Which confirms conventions by executing tree automata focused around programmed estimates? All the back-closures of the tool break down protocols under the suppositions of impeccable cryptography and that the messages are transacted over a system that is under the control of a Dolev-Yao invader. That is, the back-end process break down conventions by considering the standard protocol independent, offbeat model of a dynamic intruder who controls the system yet can't break cryptography; specifically, the attacker can intercept messages and examine them in the event that he has the comparing keys for decoding, what's more he can create messages from his insight and send them under any false name. Upon end, each back-end of the AVISPA Tool yields the result of its investigation utilizing a typical and decisively characterized yield organization expressing whether the information issue was correct (giving a depiction of the considered protocol objective or, in the event that it was abused, the related assault follow), some of the framework assets were depleted, or the issue was not handled by the needed back-end for unknown reasons.



```
AVISPA Web Tool
---------------

Usage:

 - Select a test file (or open your own custom file)
 - Change the file to your needs
 - Press Execute
```

Figure 2 - Screenshot of AVISPA web interface

4.2 The Back-End Processes

AVISPA has four back-end processes:

4.2.1 On-the-fly Model-Checker OFMC,

OFMC performs both protocol distortion and limited session confirmation, by investigating the transition framework portrayed by an IF particular in an interest driven manner (i.e., on-the-fly, henceforth its name). OFMC considers both wrote and typed convention models. OFMC's adequacy is because of the joining of a number of typical, imperative based systems, which neither are right and complete, in a manner that no attacks are lost nor are new ones presented by them. One such strategy is lazy intruder system. As an alternate critical case, the imperative separation system is a general hunt constraint-based method that incorporates the lazy intruder with thoughts from incomplete constraint-based request, and which can be formally ended up being right and complete, along these lines diminishing inquiry time by a variable of two to a few requests of size. Besides, OFMC likewise actualizes various effective inquiry heuristics. It likewise gives backing to the displaying an intruder who is fit for performing speculating attacks on powerless passwords, and for the detail of mathematical properties of cryptographic operators.

4.2.2 Constraint-Logic-based Attack Searcher CL-AtSe,

This technique applies provisions to perform both protocol alteration and confirmation for limited quantities of sessions. The protocol messages can both be typed and associative. A few properties of the XOR operator can be taken care of as well, and, all the more for the most part, CL-Atse is implicitly a measured way and is therefore open to expansions for taking care of mathematical properties of cryptographic operations. CL-Atse performs a few sorts of improvements to diminish, and frequently wipe out, redundancies or pointless limbs in the typical protocol execution. Case in point, the lazy intruder method speaks to terms typically to maintain a strategic distance from unambiguously counting the plausible messages the Dolev-Yao intruder can create, and in this way fundamentally lessens the hunt space without barring any assaults or presenting new ones. This is attained by speaking to intruder messages utilizing terms with variables, and putting away and controlling stipulations about what terms must be created and which terms may be utilized to create them.

4.2.3 SAT-based Model-Checker SATMC, and

SATMC considers the typed protocol display and performs both protocol falsification and limited session check by lessening the information issue to a succession of a SAT solvers. All the more particularly, SATMC first forms a propositional equation encoding a limited unfolding of the transition connection determined by the IF, the beginning state, and the set of states communication to a breach of the security properties. The propositional equation is then augmented to a SAT solver and any model found is made an interpretation of go into an attack. The interface between the SATMC and the SAT solver consents to the DIMACS group (a true standard for SAT issues) and accordingly SATMC can without much of a stretch consolidate and misuse new SAT solvers when they get to be accessible.

4.2.4 TA4SP protocol analyzer,

The Ta4sp (Tree Automata focused around Automatic Approximations for the Analysis of Security Protocols) back-end performs unbounded protocol check by approximating the intruder information by utilizing standard tree dialects and revising. Its beginning stage is an augmentation of a close estimation strategy focused around tree automata, for confirming security properties. The past techniques of this kind obliged the locality of a user to change by hand a security protocol into a term-changing framework and register an unprepared rough guess capacity. Our result permits one to figure an estimate capacity naturally. For mystery properties in the typed model, Ta4sp can demonstrate whether a protocol is defective (by under-close estimation) or whether it is protected for any number of sessions (by over-estimate).

4.3 Model Checking with OFMC

OFMC is a tool for falsification and verification for a bounded number of sessions. It is often used to contest the state explosion problem:

- ✓ The inexhaustible Dolev-Yao intruder model
- ✓ Number of concurrent sessions executed by honest agents

It is a very useful technique that helps reduce the search-space without excluding or introducing attacks.

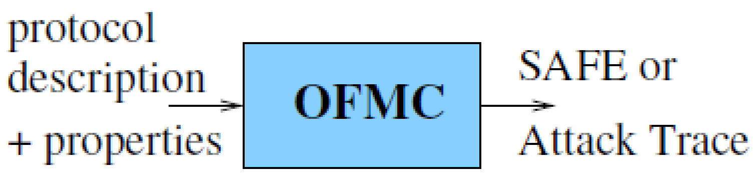


Figure 3 - OFMC as a model checker

It has five different modules:

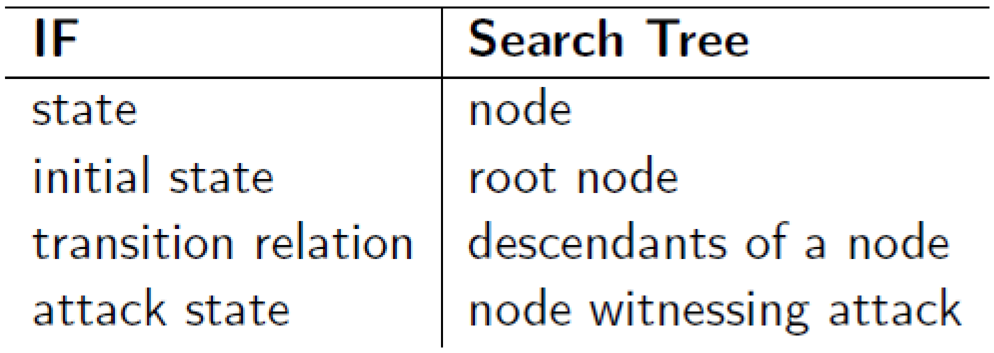
I. Introduction: IF
II. Compressions
III. Lazy intruder
IV. Symbolic sessions
V. Constraint Differentiation

4.3.1 Introduction: IF

IF is a Protocol Model. In this kind of modeling protocols are modeled as a transition system. The main factors of this model are its “states” and their “transitions”. States are defined as local conditions of honest agents and current knowledge of the attacker while Transitions are described as actions of the honest proxies and the attacker.

The intruder is well-defined as “*Dolev-Yao intruder”*. There are some conditions to this intruder model, one that it controls the whole network, second it has perfect cryptography (i.e., no information about plain text can be inferred from encrypted text) and third composition of messages is unrestrained. Messages are made up of a set of variables, constants and different cryptographic functions. IF has Asynchronous communication which means that sending and receiving messages are isolated atomic actions.

Security property of any protocol are formalized as base for characterizing “attack states” so they can be tested and analyzed. Attacks are defined by a combination of *facts* and *conditions*. Facts joined with set-dot and conditions adjoined in at conclusion. Semantically, they depict a set of states: where all facts and conditions remain intact.



IF	Search Tree
state	node
initial state	root node
transition relation	descendants of a node
attack state	node witnessing attack

Figure 4-OFMC IF

States are at all times base terms. The search tree is considerably deep and may have countless branching. On basis of that Semi-decision practice for insecurity is adopted. Lazy data-structures explore every state and heuristics and attack search as tree-converters

4.3.2 Compressions

It is a technique which syndicates rules for getting messages and directing the reply. It radically condenses the exploration space.

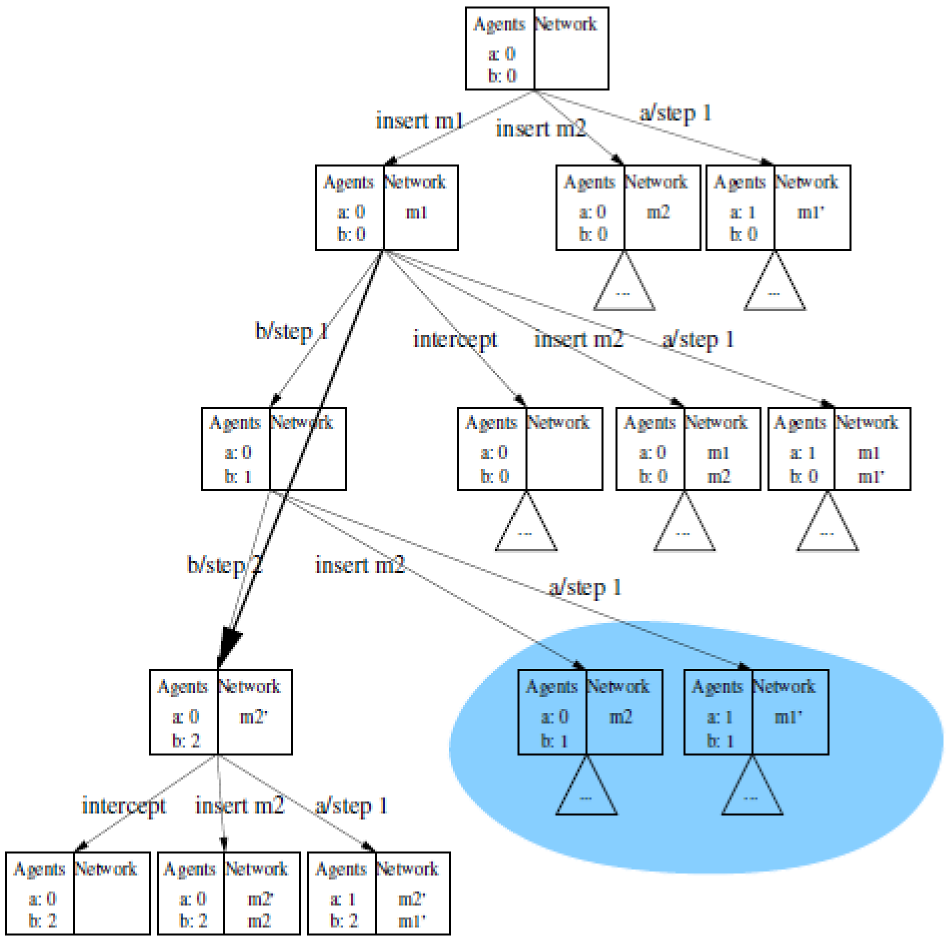


Figure 5- OFMC Compression

Since it’s not possible to differentiate intruder and network so each message that an authentic agent sends to the network is spontaneously captured; and any message that an authentic agent accepts from the network was previously introduced by the intruder.

4.3.3 Lazy intruder

Checks are assessed in a demand-driven way, hence lazy intruder. Messages are composed of different variables which are then applied to required functions for unification.it is better described for communication protocols. Additionally, intruder must be able to generate messages received up to that point

Symbolic state = term with variables + constraint set

There are two different levels of searching

- ✓ Layer 1: search in the symbolic state space
- ✓ Layer 2: constraint reduction

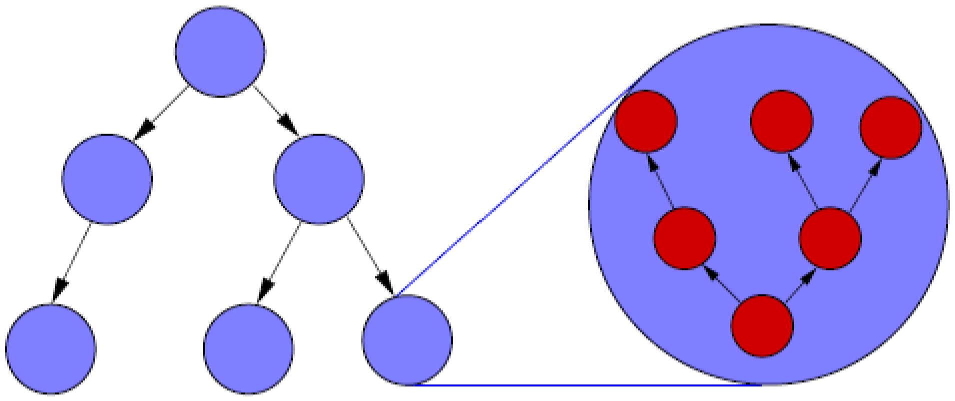


Figure 6-OFMC Lazy Intruder

4.3.4 Symbolic sessions

There is a different message for every session and for every agent Instantiation of all roles is required. It means intruder can always send messages under any identity. Check sets must be systematic; all variables must be bound by a constraint.it is assumed that intruder initially knows all agent. The agent names are selected by the intruder as well.

4.3.5 Constraint Differentiation

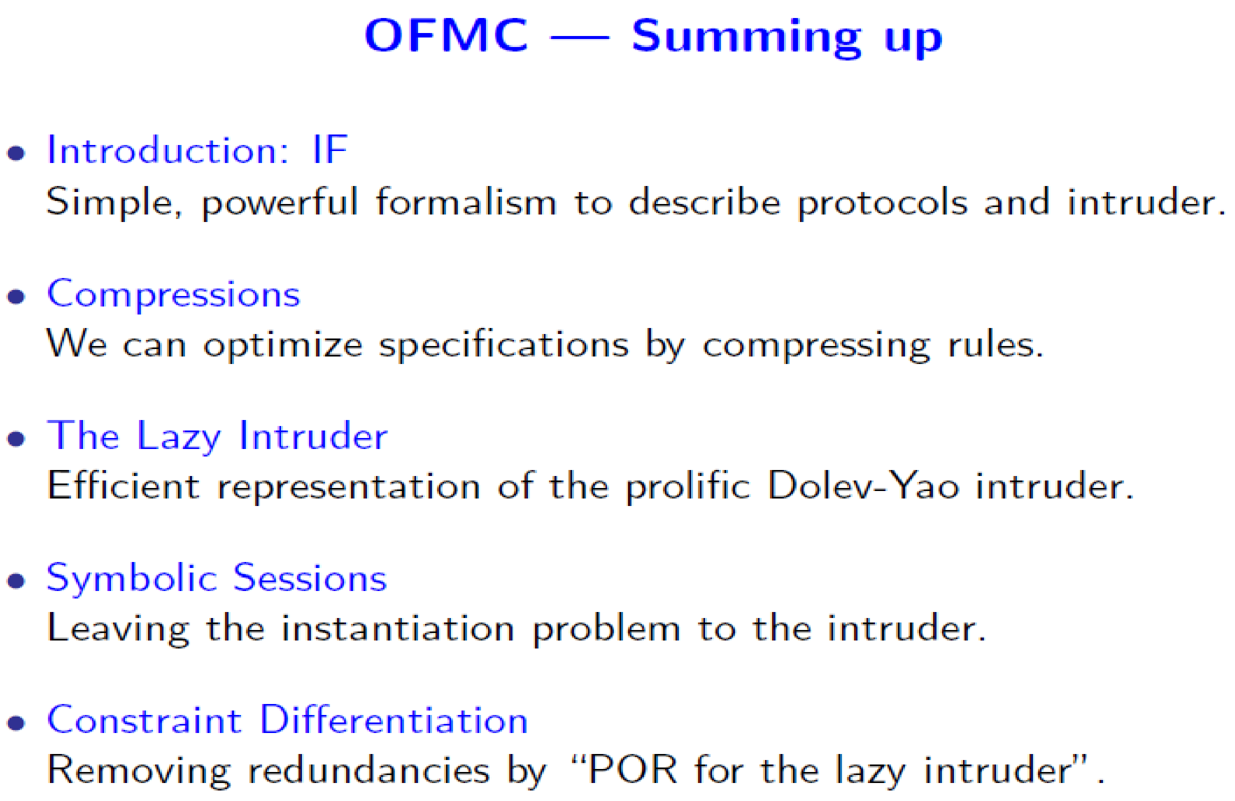
There are two key challenges of model-checking security protocols:

1. The inexhaustible Dolev-Yao intruder model.

- ✓ intruder can generate unlimited number of messages
- ✓ Lazy Intruder: symbolic demonstration of intruder. “Often just as if there were no intruder!”

2. Concurrency: Execution of parallel sessions

- ✓ Reffered as Partial-Order Reduction (POR).
- ✓ POR is restricted when using the lazy intruder method.



OFMC — Summing up

- Introduction: IF
  Simple, powerful formalism to describe protocols and intruder.
- Compressions
  We can optimize specifications by compressing rules.
- The Lazy Intruder
  Efficient representation of the prolific Dolev-Yao intruder.
- Symbolic Sessions
  Leaving the instantiation problem to the intruder.
- Constraint Differentiation
  Removing redundancies by "POR for the lazy intruder".

Figure 7-OFMC Summary

4.4 Effectiveness and Performance

The convention determinations in the IF are furnished with a mark area depicting the sorts of the messages traded among the sharing agents. This segment may be disregarded by the back-closures keeping in mind the end goal to scan for sort defect attacks; when this is the situation, we say that the back-end considers the untyped model of the security issue. It is basic that both models are considered within range as, from one viewpoint, it is vital to have the capacity to distinguish all conceivable attacks, however then again, numerous sort defect attacks are of minimal viable centrality as real usage of security protocols which frequently uphold straightforward components that avoid their appropriateness. All the four back ends have the capacity do the investigation as for the typed model, and CL-Atse and OFMC are additionally ready to embrace the untyped model. The AVISPA Tool can subsequently examine protocols as for both models.

4.5 Conclusion

The AVISPA Tool is an advanced, unified environment tool for the automatic validation of Internet security protocols. It provides unrestrained verification using generalizations, supports Algebraic properties, predicts intruder and links cryptographic and formal proof techniques. We in this thesis have focused on its formal technique OFMC. In next chapter we have described in detail the design and implementation of our proposed protocol

Chapter 5

5. Design and Implementation

SPFT is a smartphone based payment system for close network. Now -a- days smart phones have become a daily life necessity, there are all those applications which have made phones a single unit to handle most of the chores like utility bills, online financial transactions e.g., what we propose is to make it more convenient that it eliminates the use of paper money in a close environment (cafeteria bills, buying/selling, pays) like big organizations. An application that can control daily life small or big transactions just with few clicks and does not have to keep trail of paper money on daily basis. The protocol is based on conventional actors' i.e., customer, merchant, payment gateway & a financial intuition. What is different is the kind of access these parties have with each other, bank is always thought to be a close system, in this protocol it's not. All parties will be completely transparent; there will be no transaction trails and anonymity will be preserved

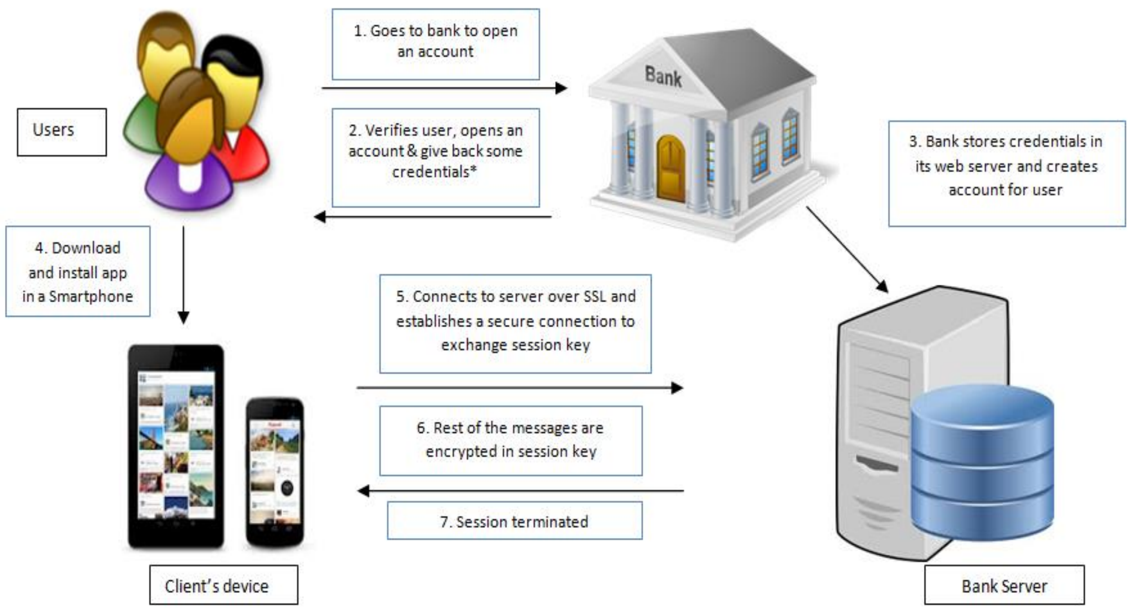

Users
1. Goes to bank to open an account
2. Verifies user, opens an account & give back some credentials*
Bank
3. Bank stores credentials in its web server and creates account for user
4. Download and install app in a Smartphone
5. Connects to server over SSL and establishes a secure connection to exchange session key
6. Rest of the messages are encrypted in session key
7. Session terminated
Client's device
Bank Server

Figure 8-SPFT Architecture

5.1 Proposed Solution

The proposed protocol comprises of following steps:

1) All users (customer & merchants) get registered with the financial institution. Users get their usernames & passwords to access the service along with a unique master key shared with bank and every user

2) User logs in to the system, views the multi-merchant multi services and choses a service he wants to avail and puts an order request to merchant. Every service has unique ID (e.g. café: 01, printer: 02) and then further every item has a unique ID (tea: 02, coffee: 04) etc.

3) Merchant receives the requests and reserves the order and replies to user with a vending-token. The vending token has item and price info (not item id)encrypted with sessionKey$_{CM}$ while an additional token for bank which only has price info signed by banks public key and then merchant private key user will 1st peel of merchant's seal and verifies the hash which will prove that it came from the merchant

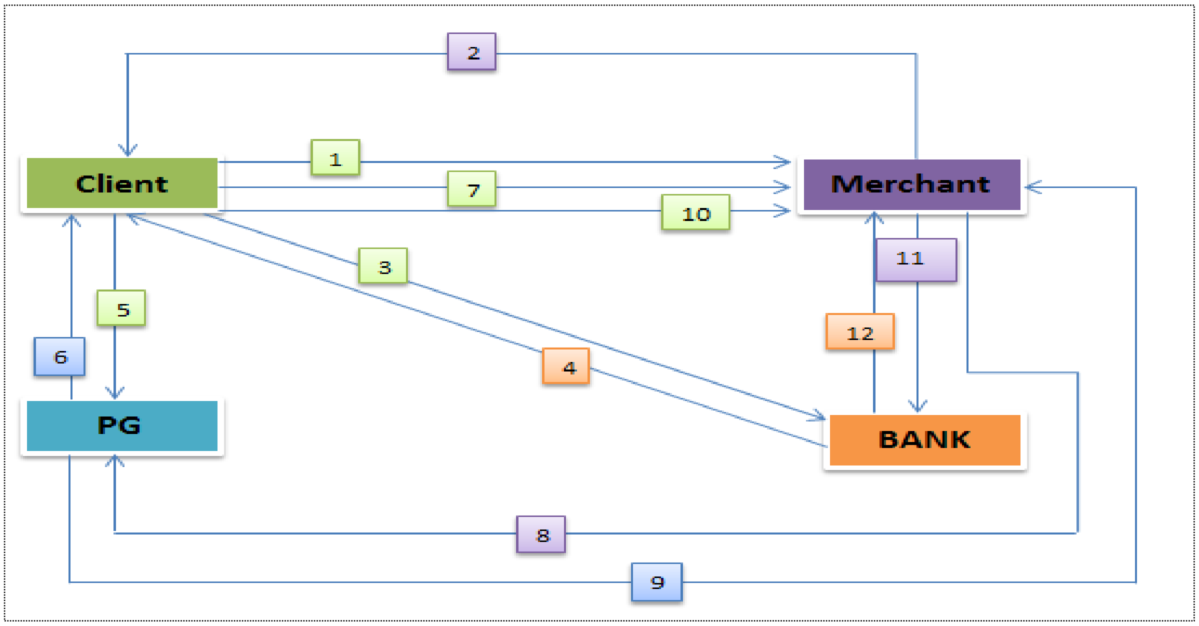


Figure 9- Flow Diagram

4) User accesses his bank account, requests the money equivalent to {Price} amount form the account. {Price} passed to bank is signed by bank's public key

5) Bank checks 1st if user is legit and requested funds are available then it generates a purchase- ticket and reserves that money from the users account and sends a sms/email confirmation to user.

6) Bank sends the ticket to client enveloped in a session key

7) Client with an addition of session key sends to payment gateway to envelope it with two keys i.e. two man rule and payment gateway generates a new ticket which is in actual the old ticket locked by two keys and signs the original ticket. Bank deducts money from the account and keeps with him for safe keeping so conflicts don't occur, bank cant access this money until its authorized by payment gateway

8) Payment gateway replies user with TICKET2, PIN enveloped in Client's public key so only client can access it and a hash of PIN signed by payment gateway to check if it is authentic or not

9) Client sends a TOKEN and TICKET2 to merchant enveloped in a session Key shared between merchant and client

10) Merchant then sends a TICKET ID to payment gateway to request for OTP related to this ID

11) Payment gateway replies user with a TICKET ID, OTP encrypted by Merchant's public key so only he can access it, hash of PIN and OTP both signed by payment for verification and all above are enveloped in a message encrypted by a session key

12) Client sends TICKET ID and PIN relative to that ID encrypted in merchant's public key and whole enveloped with session key

13) Merchant unlocks the ticket2 from OTP and then by the PIN sent to him by the client after receiving product

14) Bank transfers the amount reserved by token to merchant's account and sends merchant a conformation message

5.2 Notation Scheme

This table shows the abbreviation and full form of different notations used in protocol

Table 1- Notation Scheme

Symbols	Definition
TransactionID	Unique ID for specific transaction
ItemID	Unique ID for specific item provided by related service
ServiceID	Unique ID for specific service provided by merchant
TOKEN	Token ID, Item (description + price)
HASH1	**sha256(**TransactionID, ItemInfo, Price**)**
HASH2	**sha256 (PIN)**
HASH3	**sha256 (OTP)**
HASH4	**sha256(**{Digital money} B_{PU}**)**
TICKET1	TICKET ID, {Digital money} B_{PU} , [HASH4] B_{PR} , timestamp
TICKET2	TICKETID,{{Digital money} B_{PU} }PIN}OTP, timestamp
$\mathrm{MasterKey}_{CB}$	Master key shared between client & bank
$\mathrm{MasterKey}_{MB}$	Master key shared between merchant & bank
$\mathrm{sessionKey}_{CM}$	Session key shared between client & merchant
$\mathrm{sessionKey}_{CM2}$	2nd Session key shared between client & merchant
$\mathrm{sessionKey}_{CB}$	Session key shared between client & bank
$\mathrm{sessionKey}_{MB}$	Session key shared between bank & merchant
$\mathrm{sessionKey}_{CT}$	Session key shared between client & ticket checker
M_{PR}	Merchant's private key
M_{PU}	Merchant's public key
C_{PR}	Client's private key
C_{PU}	Client's public key

B_{PR}	Bank's private key
B_{PU}	Bank's public key
PG	Ticket checker
PG_{PR}	Payment gateway's private key
PG_{PU}	Payment gateway's public key
ItemInfo	Item description + its price

5.3 Protocol in Alice-Bob notation

5.3.1 Registration process:

1. **U** $\rightarrow$ **B**:{name, email ID, Cell No, DoB , username} B_{PU}
2. **B** $\rightarrow$ **U**:{userID, password ,$MasterKey_{CB}$}

5.3.2 Payment process:

1. **C** $\rightarrow$ **M** : {TransactionID, ServiceID ,ItemID, n1 , $sessionKey_{CM}$} M_{PU}
2. **M** $\rightarrow$ **C** : {TransactionID ,TOKEN, {Price}B_{PU}, [Hash1]M_{PR} } $sessionKey_{CM}$
3. **C** $\rightarrow$ **B** : { userID, password, {Price}$B_{PU,}$ $sessionKey_{CB}$ } $MasterKey_{CB}$
4. **B** $\rightarrow$ **C** : { TICKET1} $sessionKey_{CB}$
5. **C** $\rightarrow$ **PG**: { TICKET1, $sessionKey_{CT}$ } PG_{PU}
6. **PG** $\rightarrow$ **C** : { TICKET2, {PIN} $C_{PU,}$ [HASH2] PG_{PR} } $sessionKey_{CT}$
7. **C** $\rightarrow$ **M** : {TOKEN ,TICKET2 } $sessionKey_{CM}$
8. **M** $\rightarrow$ **PG**: { TICKETID, $sessionKey_{MT}$ } PG_{PU}
9. **PG** $\rightarrow$ **M**: { TICKETID ,{OTP} $M_{PU,}$ [HASH3]$PG_{PR,}$ [HASH2] PG_{PR} } $sessionKey_{MT}$
10. C $\rightarrow$ M: { TICKETID, [PIN] M_{PU} } $sessionKey_{CM}$
11. **M** $\rightarrow$ **B** : {TICKET1} $MasterKey_{MB}$
12. **B** $\rightarrow$ **M** : { TICKETID, Conform Message} M_{PU}

5.4 Analysing Security Properties

5.4.1 Authentication

Entity authentication is justified by the use of session keys between two parties at a time (which are symmetric in nature) which ensures that the message is actually sent by the posing party

5.4.2 Secrecy

Secrecy of the transaction messages is assured by the use of certificates and symmetric keys, as certificates are being used for exchange of session keys and signing so unless certificates itself are compromised there is no way to intrude the privacy of transactional messages

5.4.3 Integrity

Transactional integrity is ensured by two HASH1 and HASH2. Being a financial transaction relies on two things, first to approve the same price, second to make sure that the receiving party gets what the paid for or sold. HASH1 guarantees the transaction ID and price is what both parties agreed on while HASH2 confirms the correction of PIN so money can be transferred.

5.4.4 Anonymity

Client’s Identity is not known to the merchant, merchant does not know to whom he is selling until the time of fund transferring for avoiding repudiation. Payment gateway is an honest third party in this case even It does not know the identity of either merchant or client. Payment gateway is just a lock smith which locks the digital money by two keys OTP and PIN, it doesn’t care about identities

5.4.5 Non Repudiation

It is assured by the use of digital certificates and signed hashes. PIN and OTP are digitally signed by the honest payment. Both client and merchant need to provide the PIN and OTP for transference of funds and bank will transfer the money after confirmation from both parties so neither of the parties can refute

5.4.6 Two factor authentication

Digital money is locked by two keys OTP and PIN. Even if any intruder gets the either OTP or PIN he cannot access the money without second key

5.5 ANALYSIS USING AVISPA

It is an automation tool to validate security protocols. Protocols that need to be verified against properties like (secrecy, authentication, proof of origin etc.) are written in a specification language HLPSL. AVISPA at back-end works on principles of formal methods like model checking to achieve security goals and exemplify threat models. It covers four back-end practices; OFMC (on the fly model- checker), CL-AtSe (attack searcher), SATMC (SAT model checker) and TA4MC (automata based protocol analyser). We have tested the proposed protocol with all four techniques

5.6 Program Code

Code written in HLPSL (modeling language for AVISPA) is attached in appendix

5.7 Attacks

There are three major concerns for any protocol when seen from the security perspective, Secrecy, authentication and integrity. We have analysed the protocols from these three viewpoints in AVISPA

5.7.1 Secrecy

Most important parameter in financial transaction is the secrecy of transaction details and privacy of user’s personal information. In our protocol we have put secrecy check on critical points when modelling in AVISPA e.g.

I. **secret**(SessionKeyCM',purchase_order,{C,M})
II. **secret**(SessionKeyCB',sessioncb, {C,B})
III. **secret**(SessionKeyCP,sessioncp, {C,P})
IV. **secret**(OTP,otp, {C,P})

These are some security goals written in AVISPA format to check if the session keys and OTP are secure or they have been compromised during the protocol. They have all given SAFE results which means there is no information leakage

5.7.2 Authentication

Authentication is a property which ensures that both parties are what they are posing to be; actually it is to develop a trust to communicate with each other. Assuming that digital certificates haven’t been compromised when any party digitally signs something it assures that the certain thing belongs to that party. In code below hash messages have been digitally signed to confirm authentication and proof of origin. E.g.

I. RCV({TICKETID'.{DigMoney'}**_inv(SignK_B)**}_SessionKeyCB')
II. RCV({(TOKENID.ItemInfo.Price).(TICKETID'.{{{DigMoney'}**_inv(SignK_B**)}_PIN'}_OTP').h(PIN)}_SessionKeyCM)

In above code statements message parameter { Digital Money} is digitally signed by Bank's private key, which shows that certain message came from Bank that can be verified by decrypting the message by bank's public key

5.7.3 Integrity

This property ensures that data has not been altered or destroyed, and mostly its proved by the use of hashes. In our protocol we have attached a digitally signed hash of a message with itself to certify the integrity of message itself. E.g.

I. SND({TransID'.(TOKENID'.ItemInfo.{Price}_SignK_B**).{h(TransID'.Price.ItemInfo)}_inv(SignK_M)**}_SessionKeyCM')
II. SND({(TOKENID.ItemInfo.Price).(TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP').**h(PIN)**}_SessionKeyCM)

In first message hash of (**TransID +Price +ItemInfo**) is digitally signed by merchant public key, now if anyone tries to alter the ID or price in any of the message, this hash wont be equal to the hash calculated of the alter values which will show that data in transition has been tempered with and that transaction will be dropped

5.8 Results

Results of different AVISPA back-end methods on above protocol is as follow

5.8.1 OFMC

OFMC practices several symbolic techniques to symbolize the state-space. OFMC is used to prove the falsification of protocols by finding efficient attacks on them and also for the verification. i.e., for proving the protocol correct in certain situations for bounded number of sessions. Output generated by OFMC in AVISPA is as follow:

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileBBKzL9.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.53s
visitedNodes: 73 nodes
depth: 14 plies
```

5.8.2 ATSE

It's a constraint based attack searcher works on the principle of reducing redundant data. It translates the protocol in such specific language which can be useful to effectively find attacks on protocol. Output of our protocol is:

```
SUMMARY
SAFE

DETAILS
BOUNDED_NUMBER_OF_SESSIONS
TYPED_MODEL

PROTOCOL
/home/avispa/web-interface-computation/./tempdir/workfileBBKzL9.if

GOAL
As Specified

BACKEND
CL-AtSe

STATISTICS

Analysed   : 55 states
Reachable: 20 states
Translation: 0.13 seconds
Computation: 0.00 seconds
```

5.8.3 SAT

SAT-based Model-Checker (SAT) constructs a propositional formula, taking care of number of sessions and state transitions signifying an abuse of the security properties. Results of above protocol are:

```
SUMMARY

SAFE

DETAILS
STRONGLY_TYPED_MODEL
BOUNDED_NUMBER_OF_SESSIONS
BOUNDED_SEARCH_DEPTH
BOUNDED_MESSAGE_DEPTH

PROTOCOL
workfileBBKzL9.if

GOAL
%% see the HLPSL specification..
```

```
BACKEND
SATMC

COMMENTS

STATISTICS
attackFound           false    boolean
upperBoundReached   true      boolean
graphLeveledOff       0       steps
satSolver            zchaff   solver
maxStepsNumber       11       steps
stepsNumber           1       steps
atomsNumber           0       atoms
clausesNumber         0       clauses
encodingTime         0.04    seconds
solvingTime          0        seconds
CompilationTime      4.76     seconds

ATTACK TRACE
%% no attacks have been found.
```

5.9 Code written in AVISPA

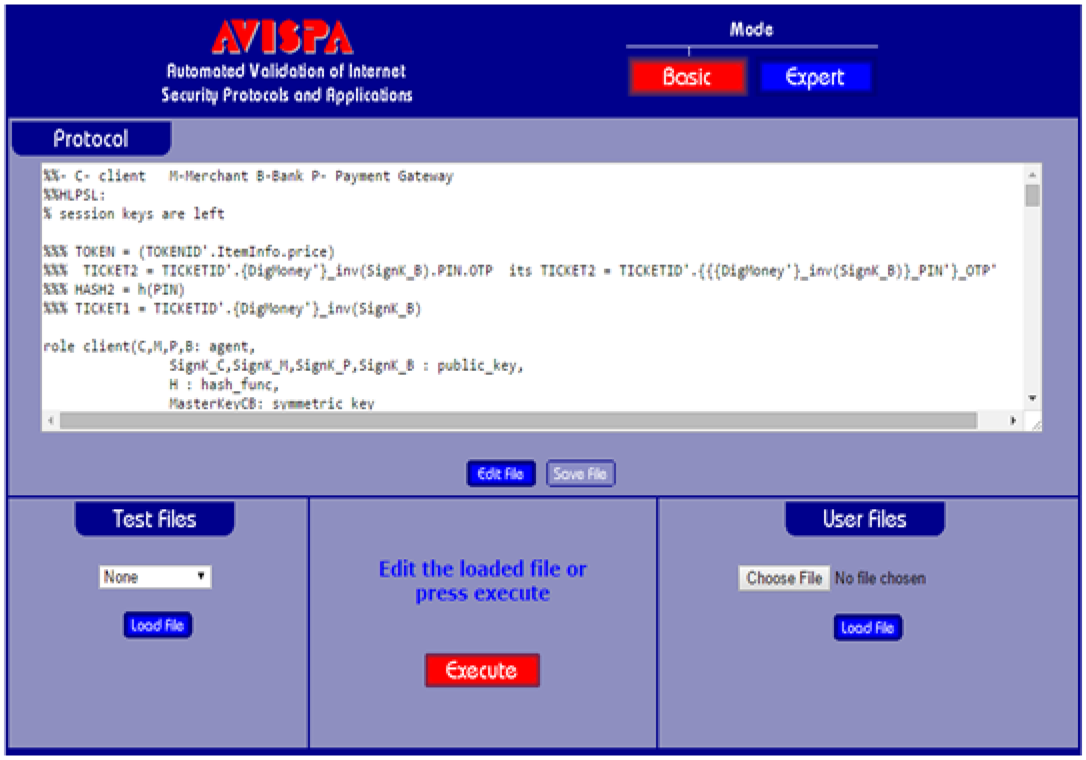
Code is written in hlpsl. Which is as follow :

```
%%- C- client      M-Merchant B-Bank P- Payment Gateway
%%HLPSL:

role client(C,M,P,B: agent,
                        SignK_C,SignK_M,SignK_P,SignK_B : public_key,
              H : hash_func,
                        MasterKeyCB: symmetric_key
                        ) played_by C def=

local S : nat,
                        TransID : nat,
                        ServID : nat,
              ItemID : nat,
                        Nonce : nat,
              TICKETID : nat,
                        TICKETID2 : nat,
                        TOKENID: nat,
              Price : text,
              ItemInfo : text,
                        UserID : text,
              Password : text,
                        DigMoney : text,
              N1 : text,
                        PIN,OTP : symmetric_key,
             SessionKeyCM, SessionKeyCB,SessionKeyCP: symmetric_key,
       SND, RCV: channel (dy)

const purchase_order,authent_request,sessioncb,sessioncp,sessioncm,otp : protocol_id
             init S := 0
transition

1.  S = 0 /\ RCV(start)  =|>
       S' := 1 /\
             TransID' := new() /\
             ServID'  := new() /\
             ItemID'  := new() /\
             N1'        :=  new() /\
             SessionKeyCM' :=  new() /\
             SND ({TransID'.ServID'.ItemID'.N1'.SessionKeyCM'}_SignK_M)        %% sent to merchant
             /\     witness(C,M,c_m_nonce,N1')
       /\ secret(SessionKeyCM',purchase_order,{C,M})
```

```
2.                              S                               =                               1                          /\
RCV({TransID'.(TOKENID'.ItemInfo.{Price}_SignK_B).{H(TransID'.Price.ItemInfo)}_inv(SignK_M)}_SessionKeyCM')
=|>   %%from merchant


          S':= 2 /\
     SessionKeyCB' :=  new() /\
     SND ({UserID.Password.{Price}_SignK_B.SessionKeyCB'}_MasterKeyCB)   %%sent to Bank      %%3rd message
    /\     secret(SessionKeyCB',sessioncb, {C,B})

3. S = 2 /\  RCV ({TICKETID'.{DigMoney'}_inv(SignK_B)}_SessionKeyCB')   %% received 4th message from bank
            =|>       S':= 3 /\
       SessionKeyCP' :=  new() /\
    SND  ({TICKETID'.{DigMoney'}_inv(SignK_B).SessionKeyCP'}_SignK_P)    %5.  client  to  PG      //C->PG:  {  TICKET1,
sessionKeyCT } PGPU
  /\  secret(SessionKeyCP,sessioncp, {C,P})


4.          S          =          3                /\          RCV({          TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP'.{
{PIN'}_inv(SignK_P)}_SignK_C}_SessionKeyCP')      %%6t message, received from PG
            =|>   S':= 4 /\


       %%7th message, sent to Merchant     %%%C -> M:{TOKEN ,TICKET2,HASH2} sessionKeyCM
   request(M,C,info,h(TransID.Price.ItemInfo)) /\

            SND
({(TOKENID.ItemInfo.Price).(TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP').h(PIN)}_SessionKeyCM)
%%sent to merchant , 7th message

5.  S = 4  =|>                          %%%.       C->M: { [PIN] CPR} sessionKeyCM
   SND ({{PIN}_inv(SignK_C)}_SessionKeyCM)                %%sent 10th message to Merchant
     /\ secret(OTP,otp, {C,P})
   /\  S' := 5


end role

%% Bank=======================================================

role bank(C,M,P,B: agent,
                    SignK_C,SignK_M,SignK_P,SignK_B : public_key,
             H : hash_func,
             MasterKeyCB, MasterKeyMB: symmetric_key

                    )
played_by B
def=
                local S : nat,
                TransID : nat,
        ServID : nat,
        ItemID : nat,
                Nonce : nat,
        TICKETID : nat,
        TOKENID: nat,
                Price : text,
        ItemInfo : text,
                UserID : text,
        Password : text,
                DigMoney : text,
                PIN,OTP : symmetric_key,
                Conform_Msg : text,
            SessionKeyCM, SessionKeyCB,SessionKeyCP : symmetric_key,
        SND, RCV: channel (dy)

const purchase_order,authent_request,sessioncb,sessioncp,sessioncm : protocol_id

    init S := 2

transition
1. S = 2 /\
 RCV ({UserID.Password.{Price}_SignK_B.SessionKeyCB'}_MasterKeyCB)   %%received from client  ,,        %%3rd
message
 =|>   S':= 4 /\
    TICKETID' := new() /\
    DigMoney' := new() /\
    SND ({TICKETID'.{DigMoney'}_inv(SignK_B)}_SessionKeyCB')   %% bank to client    %%4th message
   /\    secret(SessionKeyCB',sessioncb, {B,C})

2.  S = 4 /\

  %%received 11th message from merchant

RCV({TICKETID.{DigMoney}_inv(SignK_B)}_MasterKeyMB)    =|>

%% confirmation to merchant ,,12th message and final message

S' := 6 /\
SND ({TICKETID.Conform_Msg}_SignK_M)

end role


%%=======Merchant=============================================
role merchant( C,M,P,B: agent,
            SignK_C,SignK_M,SignK_P,SignK_B : public_key,
            H : hash_func,
            MasterKeyMB    : symmetric_key
           ) played_by M def=

local S : nat,
                 TransID : nat,
            ServID : nat,
            ItemID : nat,
                 Nonce : nat,
            TICKETID : nat,
            TOKENID: nat,
                 Price : text,
            ItemInfo : text,
                 UserID : text,
            Password : text,
                 DigMoney : text,
                 Conform_Msg : text,
                 N1 : text,
                 PIN,OTP : symmetric_key,
           SessionKeyCM, SessionKeyCB,SessionKeyCP,SessionKeyMP: symmetric_key,
            SND, RCV: channel (dy)

const purchase_order1,authent_request,sessioncb,sessioncp,sessioncm : protocol_id
           init S := 1

transition

1.  S = 1 /\ RCV ({TransID'.ServID'.ItemID'.N1'.SessionKeyCM'}_SignK_M) =|>          %%received from clinent (1
message)
           S' := 3 /\
           TOKENID' := new() /\

  %%sent to client by merchant %%%      M -> C: { TransactionID ,TOKEN, {Price}BPU, [Hash1]MPR} sessionKeyCM

   SND({
TransID'.(TOKENID'.ItemInfo.{Price}_SignK_B).{h(TransID'.Price.ItemInfo)}_inv(SignK_M)}_SessionKeyCM')
%    /\     witness(C,M,info,h(TransID'.Price.ItemInfo))


2.  S = 3 /\

%%received 7th message from client, and merchant sent it to PG for 8th message

RCV({(TOKENID.ItemInfo.Price).(TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP').h(PIN)}_SessionKeyCM)
=|>
    S':= 5 /\
    SessionKeyMP' :=  new() /\
%%%8th message, sent to PG from merchant         %%%%8.        M-> PG: { TICKETID, sessionKeyMT } PGPU
SND( {TICKETID.SessionKeyMP'}_SignK_P)



3. S = 5  /\

   RCV({ TICKETID.{OTP}_inv(SignK_P)}_SessionKeyMP')      %%% 9th message received from PG
      =|>
       S' := 7

4. S = 7 /\

%% 10th message received from client
   RCV({TICKETID'.{PIN}_inv(SignK_C)}_SessionKeyCM) =|>
           S' := 9 /\

 %%11th message sent to bank from merchant           %%%         M->B: {TICKET1} MasterKeyMB

SND({TICKETID.{DigMoney}_inv(SignK_B)}_MasterKeyMB)


5.  S = 9 /\
  %% received from bank confirmation, 12the message
RCV ({TICKETID.Conform_Msg}_SignK_M)  =|> S' := 11

end role

%%===payment=================================================
role paymentGateway (C,M,P,B: agent,
            SignK_C,SignK_M,SignK_P,SignK_B : public_key,
            H : hash_func,
            MasterKeyCB: symmetric_key
)
played_by P
def=
                      local S : nat,
                      TransID : nat,
            ServID : nat,
            ItemID : nat,
                      Nonce : nat,
            TICKETID : nat,
            TICKETID2 : nat,
                      TOKENID: nat,
                      Price : text,
            ItemInfo : text,
                      UserID : text,
                Password : text,
                        DigMoney : text,
                        PIN,OTP : symmetric_key,
               SessionKeyCM, SessionKeyCB,SessionKeyCP,SessionKeyMP: symmetric_key,
        SND, RCV: channel (dy)
const purchase_order1,authent_request,sessioncb,sessioncp,otp,pin : protocol_id
init S := 3

transition


1. S = 3  /\  RCV ({TICKETID'.{DigMoney'}_inv(SignK_B).SessionKeyCP'}_SignK_P)  =|>         %%5th message
received from client by PG %%
  S':= 6 /\
     PIN' := new() /\
     OTP' := new() /\
 %%6th message, sent this to client  %%PG->C: { TICKET2, {[PIN] PGPR} CPU } sessionKeyCT
        SND({ TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP'.{ {PIN'}_inv(SignK_P)}_SignK_C}_SessionKeyCP')

        /\ secret(OTP',otp, {C,P})

2. S = 6  /\  RCV({TOKENID.SessionKeyMP'}_SignK_P)    =|>  %% 8th message received from merchant,
  S' :=9 /\

%%% 9th message sent to merchant     %%%.    PG->M : { [OTP] PGPR } sessionKeyMT


  SND ({ TICKETID.{OTP}_inv(SignK_P)}_SessionKeyMP')

end role

%=================================================================

role session(C,M,P,B: agent,
                SignK_C,SignK_M,SignK_P,SignK_B : public_key,
                H : hash_func,
                MasterKeyCB, MasterKeyMB: symmetric_key
               ) def=

local  SA, SB,SC,SD, RA, RB,RC,RD: channel (dy)
 composition
  client(C,M,P,B,SignK_C,SignK_M,SignK_P,SignK_B,H,MasterKeyCB) /\

 bank(C,M,P,B,SignK_C,SignK_M,SignK_P,SignK_B,H,MasterKeyCB,MasterKeyMB) /\

  merchant(C,M,P,B,SignK_C,SignK_M,SignK_P,SignK_B,H,MasterKeyMB) /\

  paymentGateway(C,M,P,B,SignK_C,SignK_M,SignK_P,SignK_B,H,MasterKeyCB)
```

```
end role

role environment()
def=
%local
%local sa, ra: channel (dy)
  %const h: hash_func,
const c_m_nonce  : protocol_id,
    h: hash_func,
     c,m,p,b: agent,
        sign_c,sign_m,sign_p,sign_b,sign_i: public_key,
        masterkeycb, masterkeymb: symmetric_key


intruder_knowledge = {c,m,p,b,sign_c,sign_m,sign_p,sign_b,sign_i,inv(sign_i)}

composition

session(c,m,p,b,sign_c,sign_m,sign_p,sign_b,h,masterkeycb,masterkeymb)
%%/\ session(c,m,p,i,sign_c,sign_m,sign_p,sign_i,h,masterkeycb,masterkeymb)
%%/\session(c,i,p,b,sign_c,sign_i,sign_p,sign_b,h,masterkeycb,masterkeymb)



end role

goal
  %%%%%%%%%%%%%%%%we are achieving following Goal %%%
  % Message authentication (G2)
  % Replay protection (G3)
  % Accountability (G17)
  % Proof of Origin (G18)

 % authentication_on info
 % weak_authentication_on N1
  % Conifidentiality (G12)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

authentication_on c_m_nonce
%authentication_on info
secrecy_of purchase_order
secrecy_of purchase_order1
secrecy_of sessioncb
secrecy_of sessioncp
secrecy_of otp
end goal
environment()
```

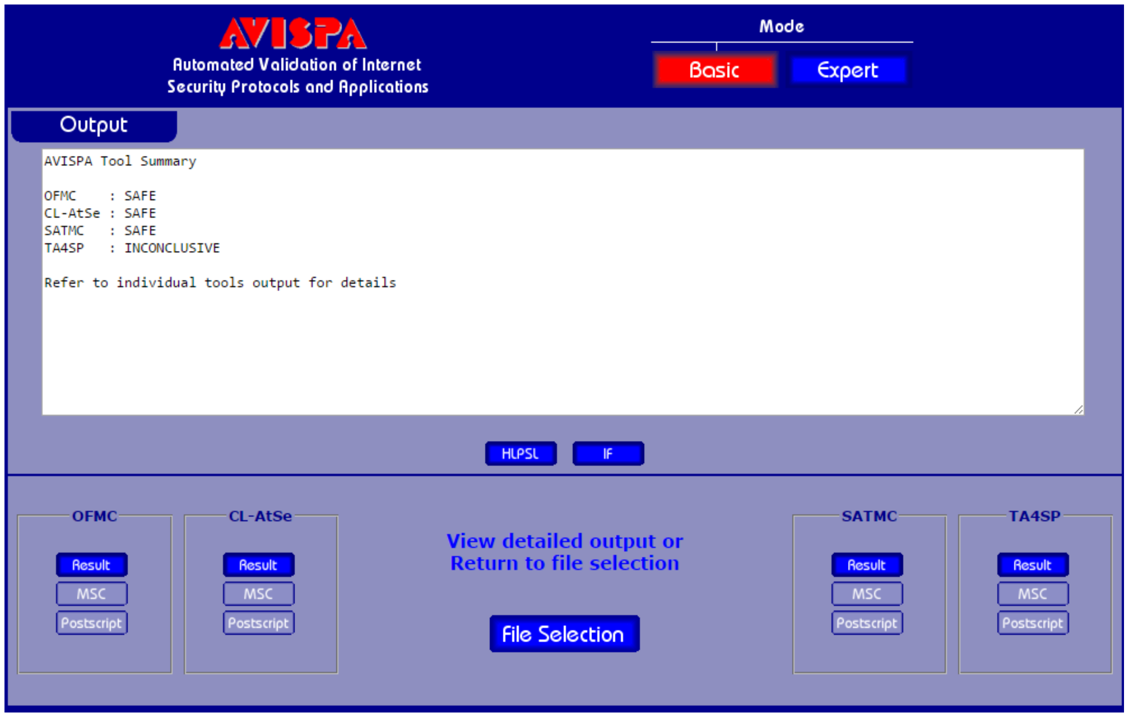
The written code is saved in a file whose extension is set as .hlpsl. this file is loaded into a web interface of AVISPA



```
%%- C- client   M-Merchant B-Bank P- Payment Gateway
%%HLPSL:
% session keys are left

%%% TOKEN = (TOKENID'.ItemInfo.price)
%%%  TICKET2 = TICKETID'.{DigMoney'}_inv(SignK_B).PIN.OTP  its TICKET2 = TICKETID'.{{{DigMoney'}_inv(SignK_B)}_PIN'}_OTP'
%%% HASH2 = h(PIN)
%%% TICKET1 = TICKETID'.{DigMoney'}_inv(SignK_B)

role client(C,M,P,B: agent,
                SignK_C,SignK_M,SignK_P,SignK_B : public_key,
                H : hash_func,
                MasterKeyCB: symmetric key
```

Figure 10- file loading in AVISPA

Then when execute is pressed, written code is verified via the backend processes of AVISPA one by one and results are shown as follow:


AVISPA
Automated Validation of Internet Security Protocols and Applications
Mode
Basic
Expert
Output
HLPSL
IF
OFMC
CL-AtSe
SATMC
TA4SP
Result
MSC
Postscript
View detailed output or Return to file selection
File Selection

```
AVISPA Tool Summary

OFMC    : SAFE
CL-AtSe : SAFE
SATMC   : SAFE
TA4SP   : INCONCLUSIVE

Refer to individual tools output for details
```

Figure 11-Results in AVISPA

5.10 Implementation

Implementation of the proposed protocol has been done to show the proof of concept. Payment Gateway has been done in java on Net beans as platform with Apache server, merchant is implemented in JSP while client/user is implemented in android

5.10.Coding and User Interface

Client is implemented in android. It has a user friendly interface, and mainly it is used to order or buy something from merchant, Following are the screen which

On start of the application first login screen is displayed, it asks for username and password, on entering information, these credentials are sent to the central database for validation, when data is successfully validated, user is directed to the next screen, which is the available list of merchants, out of which if any merchant is selected, then merchant id of that merchant is selected and is sent to the central database to fetch the available items that a particular merchant is selling so a user can select the item of his choice. The screen shots of these activities are as follow:

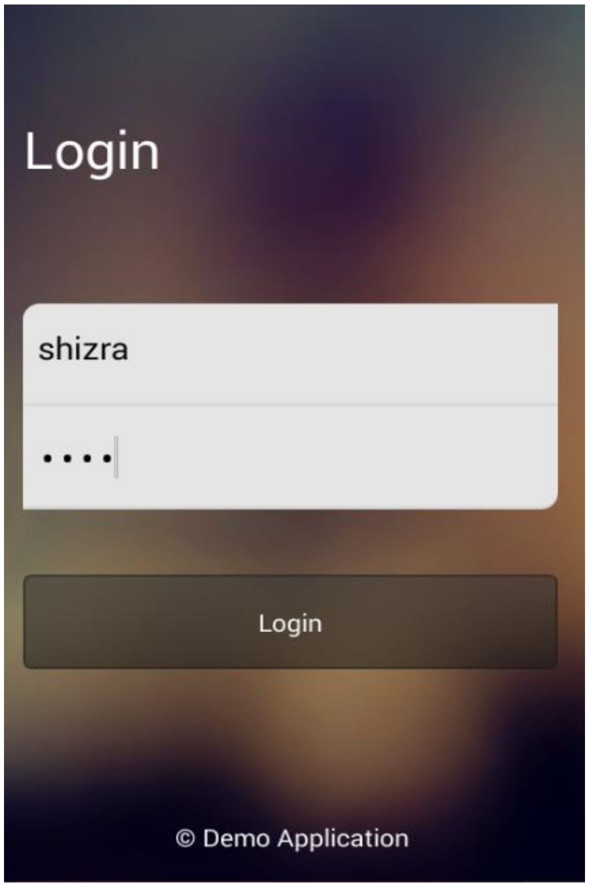


Figure 12- SPFT Login

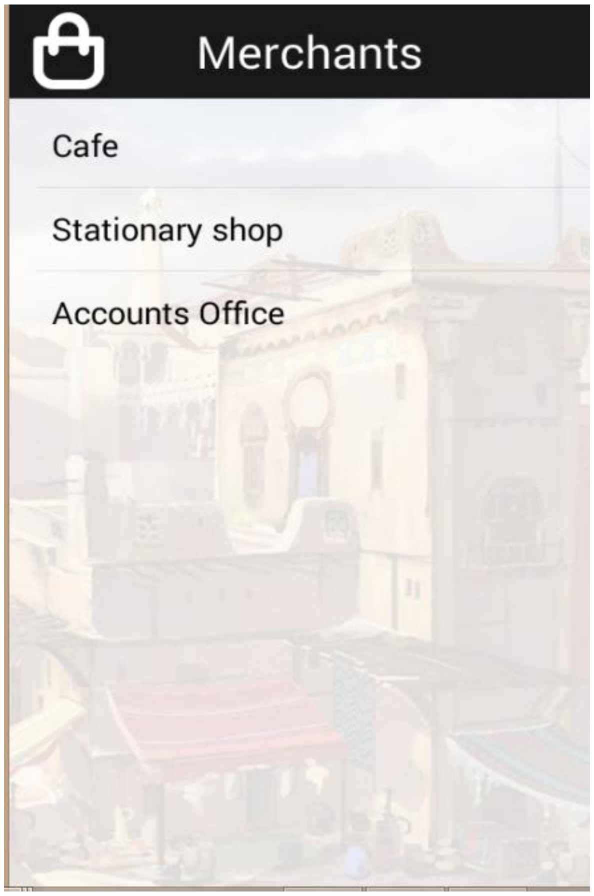
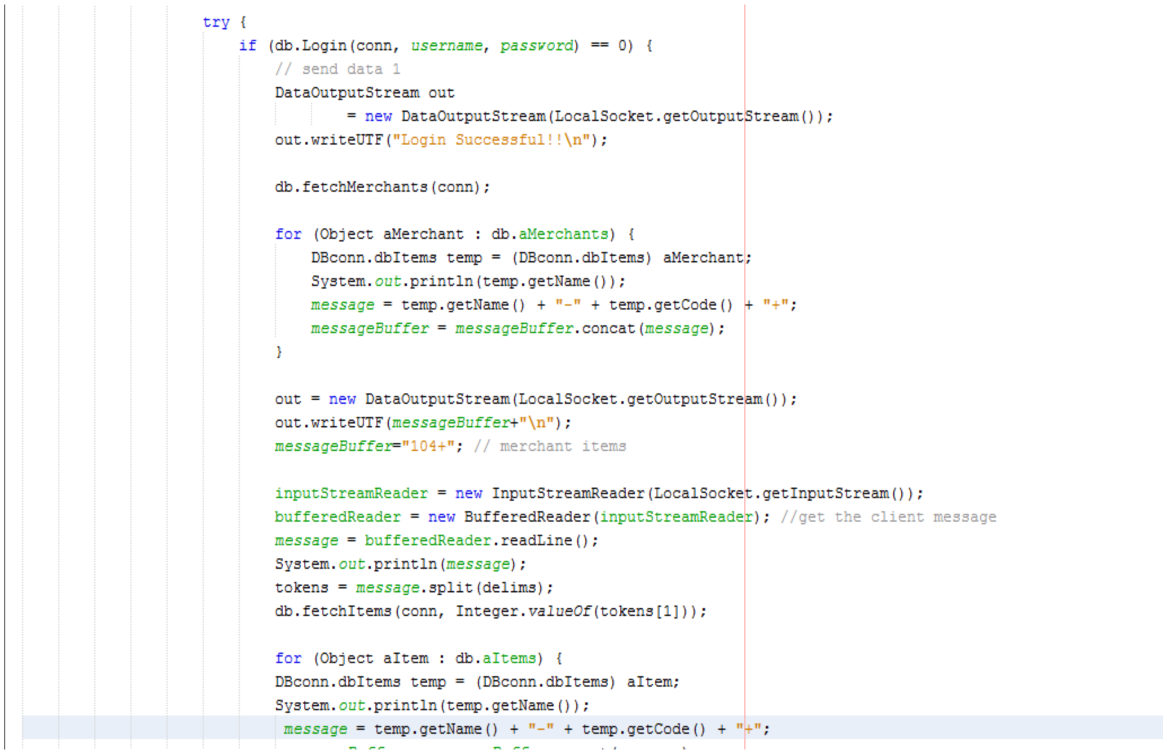
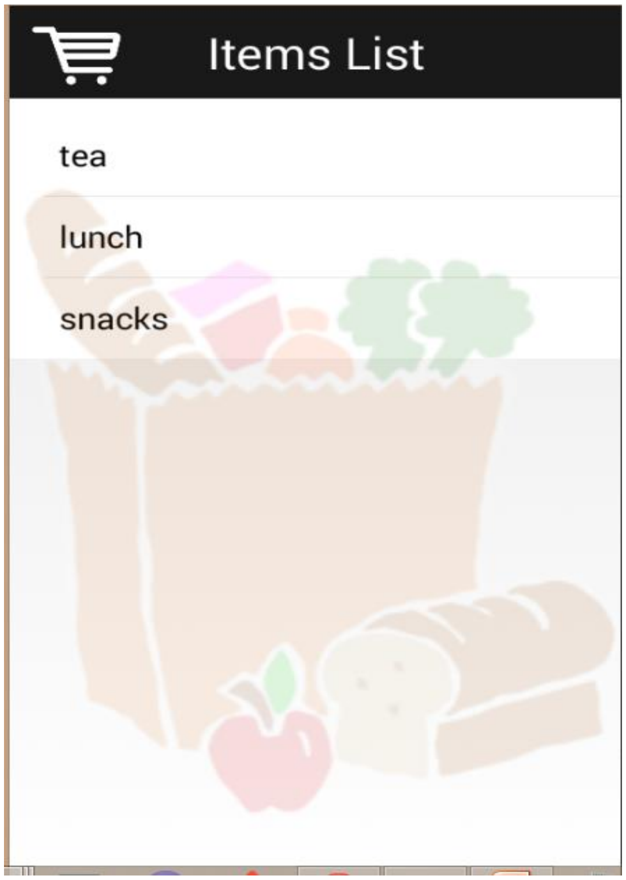


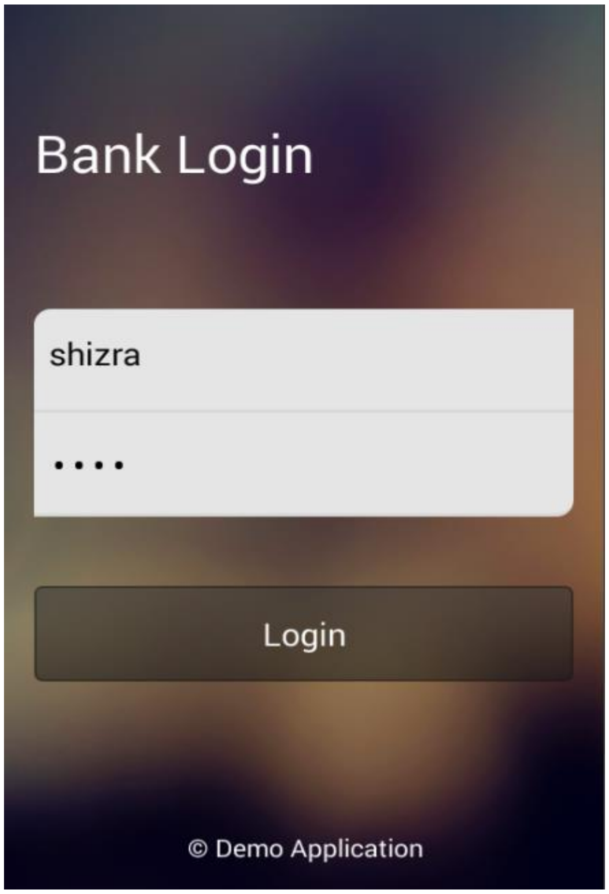
Figure 13- SPFT Merchant List

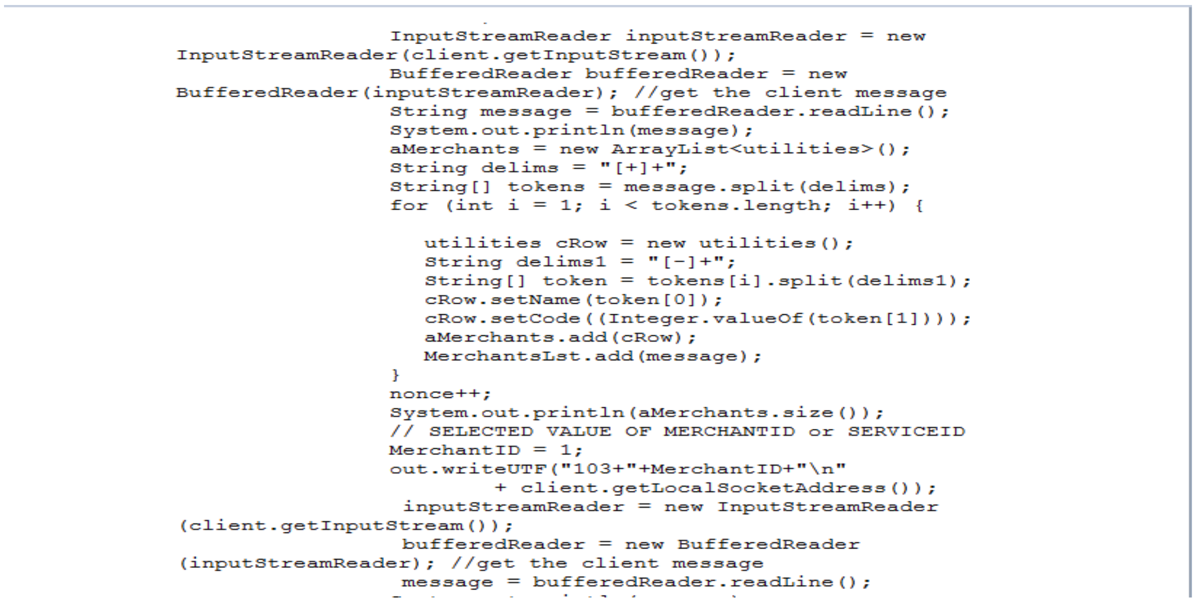


```
try {
    if (db.Login(conn, username, password) == 0) {
        // send data 1
        DataOutputStream out
                = new DataOutputStream(LocalSocket.getOutputStream());
        out.writeUTF("Login Successful!!\n");

        db.fetchMerchants(conn);

        for (Object aMerchant : db.aMerchants) {
            DBconn.dbItems temp = (DBconn.dbItems) aMerchant;
            System.out.println(temp.getName());
            message = temp.getName() + "-" + temp.getCode() + "+";
            messageBuffer = messageBuffer.concat(message);
        }

        out = new DataOutputStream(LocalSocket.getOutputStream());
        out.writeUTF(messageBuffer+"\n");
        messageBuffer="104+"; // merchant items

        inputStreamReader = new InputStreamReader(LocalSocket.getInputStream());
        bufferedReader = new BufferedReader(inputStreamReader); //get the client message
        message = bufferedReader.readLine();
        System.out.println(message);
        tokens = message.split(delims);
        db.fetchItems(conn, Integer.valueOf(tokens[1]));

        for (Object aItem : db.aItems) {
        DBconn.dbItems temp = (DBconn.dbItems) aItem;
        System.out.println(temp.getName());
         message = temp.getName() + "-" + temp.getCode() + "+";
```

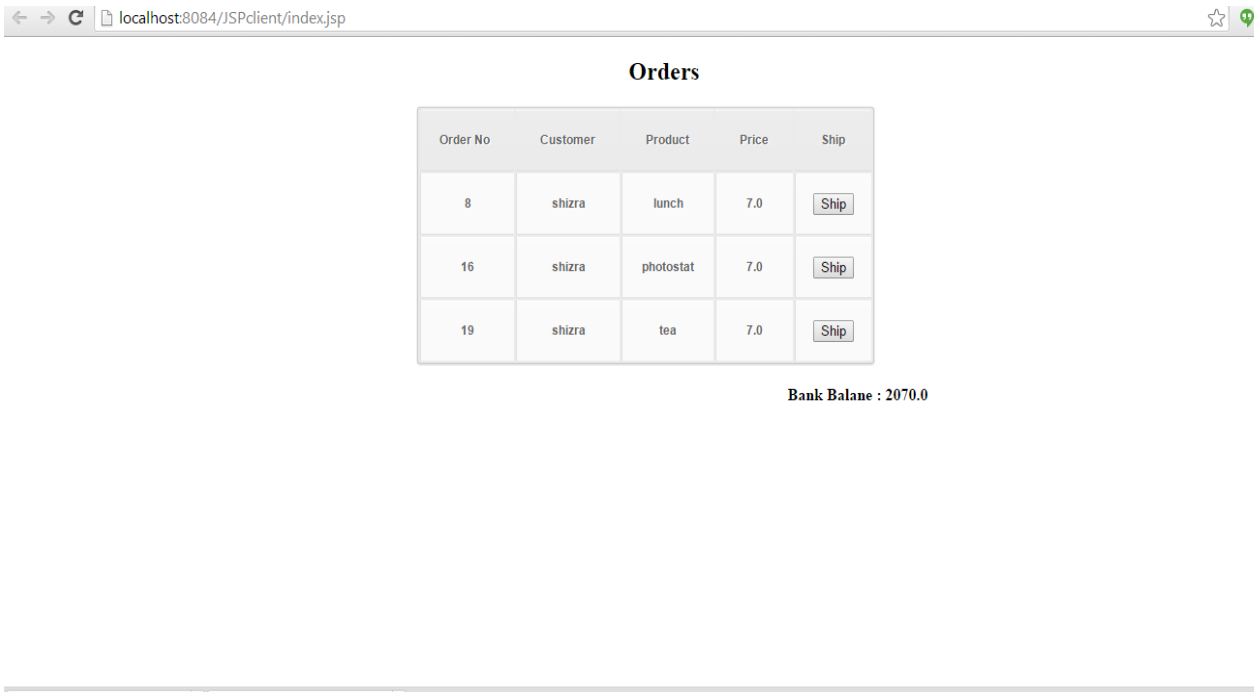
Figure 14- Screenshot of SPFT Payment Gateway







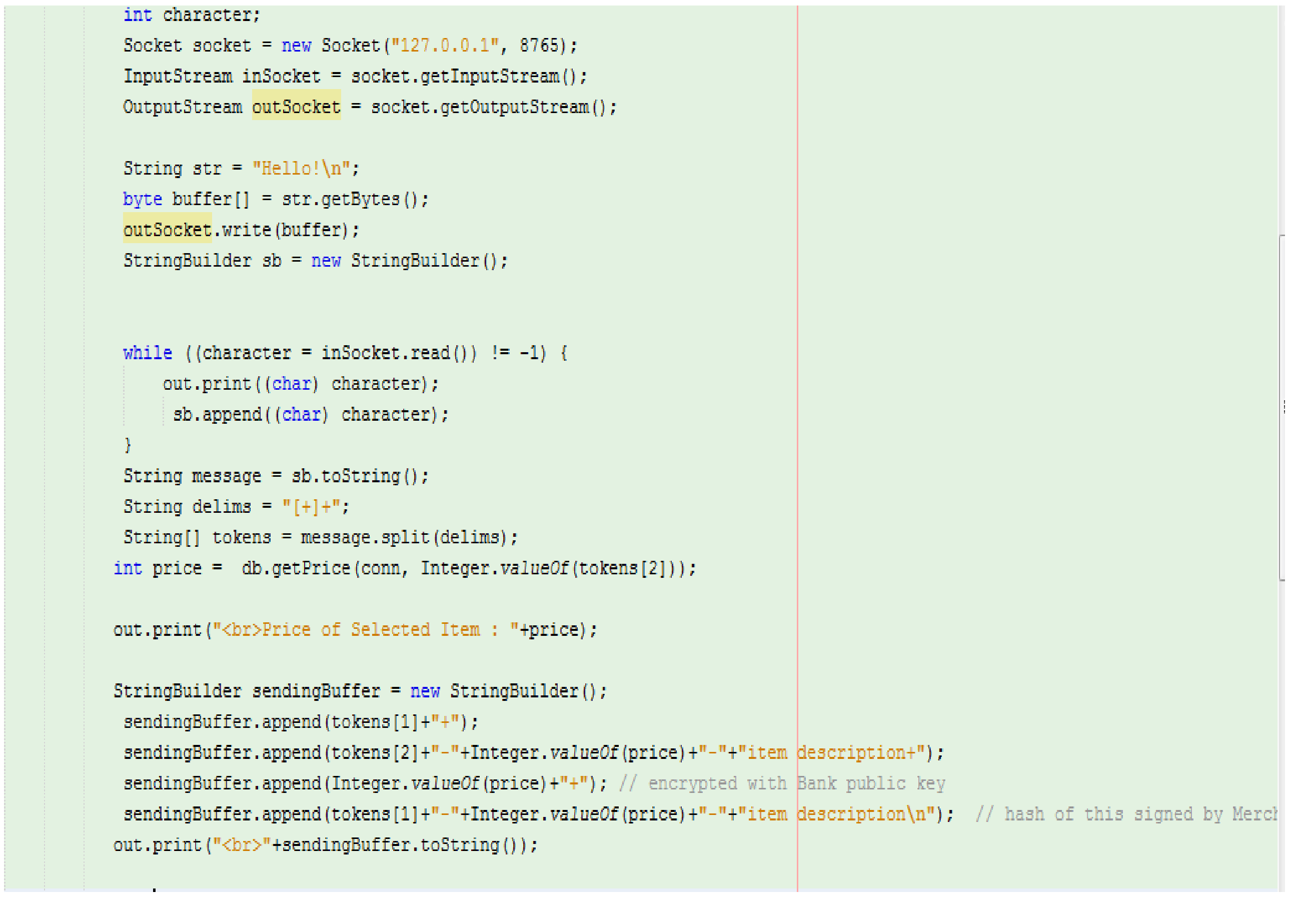
```
                    InputStreamReader inputStreamReader = new
InputStreamReader(client.getInputStream());
                    BufferedReader bufferedReader = new
BufferedReader(inputStreamReader); //get the client message
                    String message = bufferedReader.readLine();
                    System.out.println(message);
                    aMerchants = new ArrayList<utilities>();
                    String delims = "[+]+";
                    String[] tokens = message.split(delims);
                    for (int i = 1; i < tokens.length; i++) {

                       utilities cRow = new utilities();
                       String delims1 = "[-]+";
                       String[] token = tokens[i].split(delims1);
                       cRow.setName(token[0]);
                       cRow.setCode((Integer.valueOf(token[1])));
                       aMerchants.add(cRow);
                       MerchantsLst.add(message);
                    }
                    nonce++;
                    System.out.println(aMerchants.size());
                    // SELECTED VALUE OF MERCHANTID or SERVICEID
                    MerchantID = 1;
                    out.writeUTF("103+"+MerchantID+"\n"
                            + client.getLocalSocketAddress());
                     inputStreamReader = new InputStreamReader
(client.getInputStream());
                     bufferedReader = new BufferedReader
(inputStreamReader); //get the client message
                     message = bufferedReader.readLine();
```



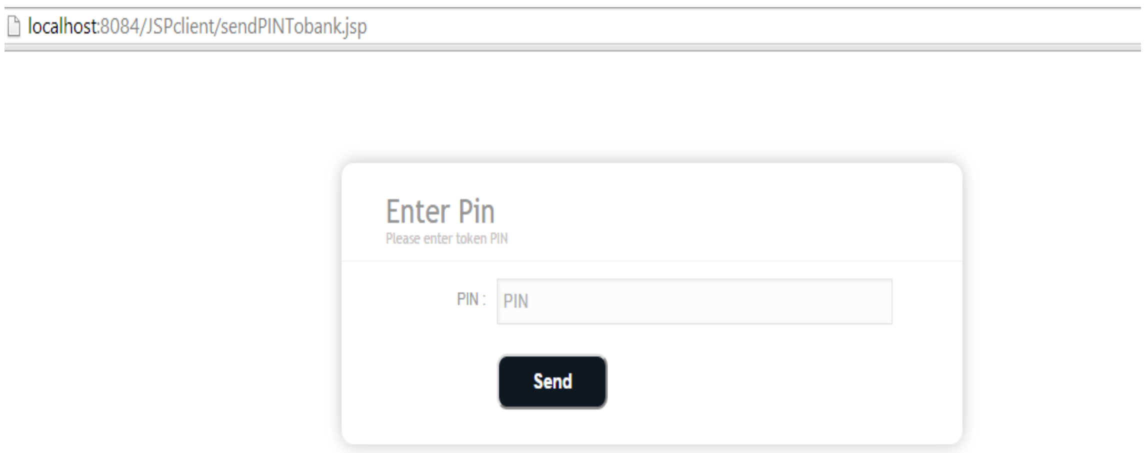
Orders

Order No	Customer	Product	Price	Ship
8	shizra	lunch	7.0	Ship
16	shizra	photostat	7.0	Ship
19	shizra	tea	7.0	Ship

Bank Balane : 2070.0



```
    int character;
    Socket socket = new Socket("127.0.0.1", 8765);
    InputStream inSocket = socket.getInputStream();
    OutputStream outSocket = socket.getOutputStream();

    String str = "Hello!\n";
    byte buffer[] = str.getBytes();
    outSocket.write(buffer);
    StringBuilder sb = new StringBuilder();


    while ((character = inSocket.read()) != -1) {
        out.print((char) character);
         sb.append((char) character);
    }
    String message = sb.toString();
    String delims = "[+]+";
    String[] tokens = message.split(delims);
   int price =  db.getPrice(conn, Integer.valueOf(tokens[2]));

   out.print("<br>Price of Selected Item : "+price);

   StringBuilder sendingBuffer = new StringBuilder();
    sendingBuffer.append(tokens[1]+"+");
    sendingBuffer.append(tokens[2]+"-"+Integer.valueOf(price)+"-"+"item description+");
    sendingBuffer.append(Integer.valueOf(price)+"+"); // encrypted with Bank public key
    sendingBuffer.append(tokens[1]+"-"+Integer.valueOf(price)+"-"+"item description\n");  // hash of this signed by Merch
   out.print("<br>"+sendingBuffer.toString());
```


localhost:8084/JSPclient/sendPINTobank.jsp

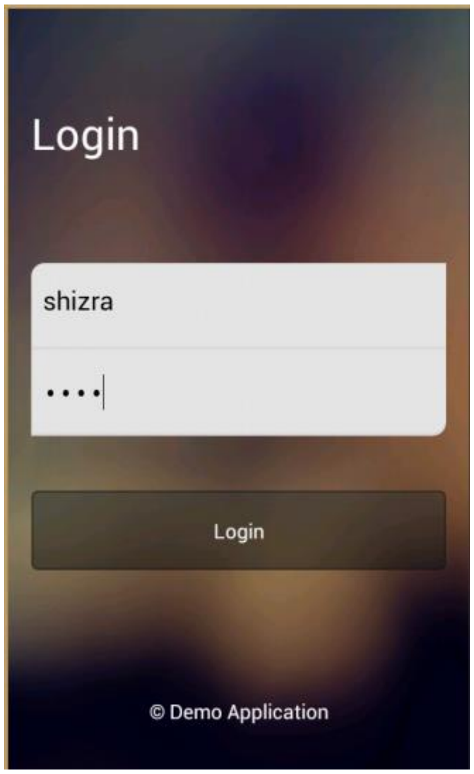
Enter Pin

Please enter token PIN

PIN : PIN

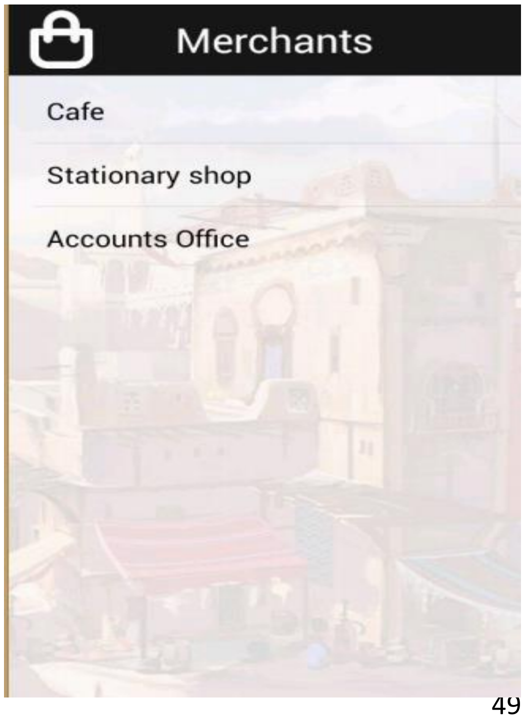
```
            i
                 nonce = 1;
             String serverName = "10.0.2.2";
             int port = Integer.parseInt("6066");
             System.out.println("Connecting to " + serverName
                     + " on port " + port);
              try {
                  Socket client = new Socket(serverName, port);

                  System.out.println("Just connected to "
                          + client.getRemoteSocketAddress());
                  OutputStream outToServer = client.getOutputStream
();
                  DataOutputStream out =
                          new DataOutputStream(outToServer);
                  nonce++;

              /*    out.writeUTF("101+username+password\n"
                          + client.getLocalSocketAddress());*/
                  out.writeUTF("101+"+User+"+"+PassCode+"+\n"
                          + client.getLocalSocketAddress());

                  InputStream inFromServer = client.getInputStream
();
                  DataInputStream in1 =
                          new DataInputStream(inFromServer);
                    loginStatus= in1.readUTF();
                  System.out.println("Login Info : " +
loginStatus);
```

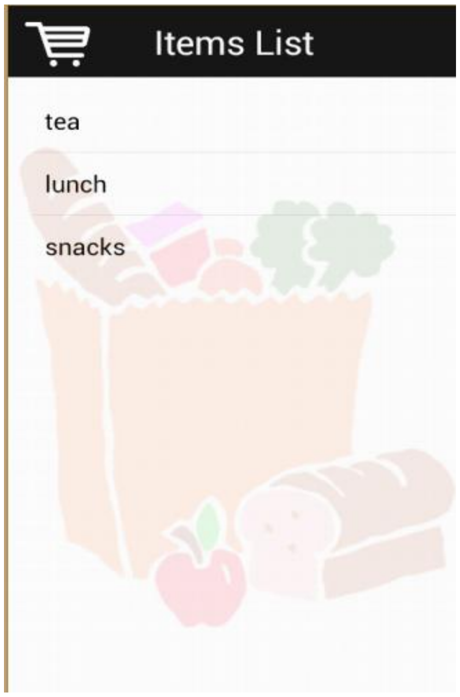
5.10.3 Workflow



1. First of all the user enter his/her credentials to login into the app.

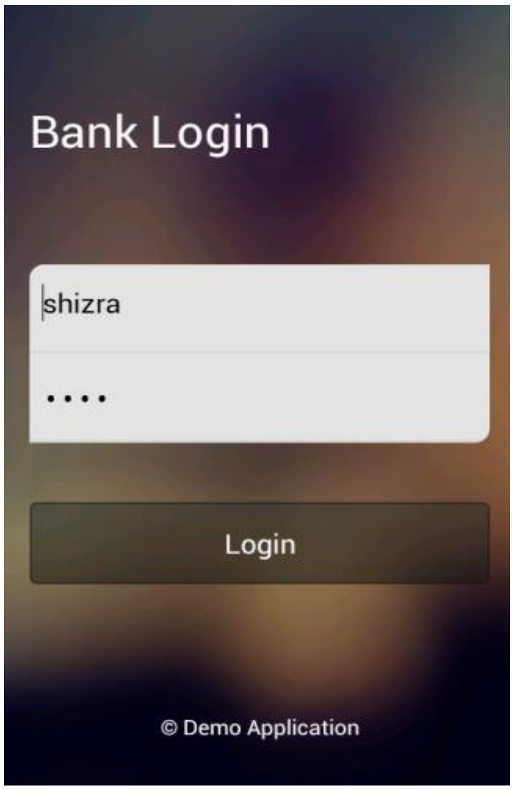
2. The user is then presented with a list of merchants to select from to buy a product.

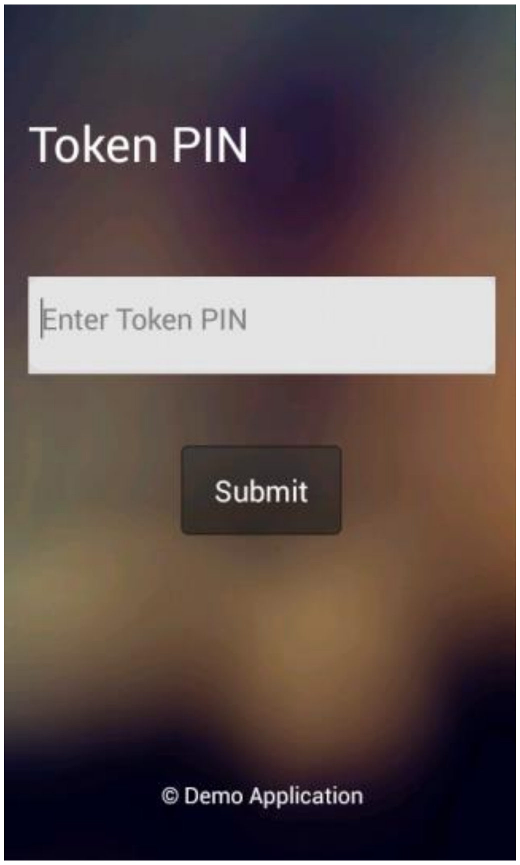




3. The user is presented with a list of items available for purchase from the merchant selected in the previous screen.

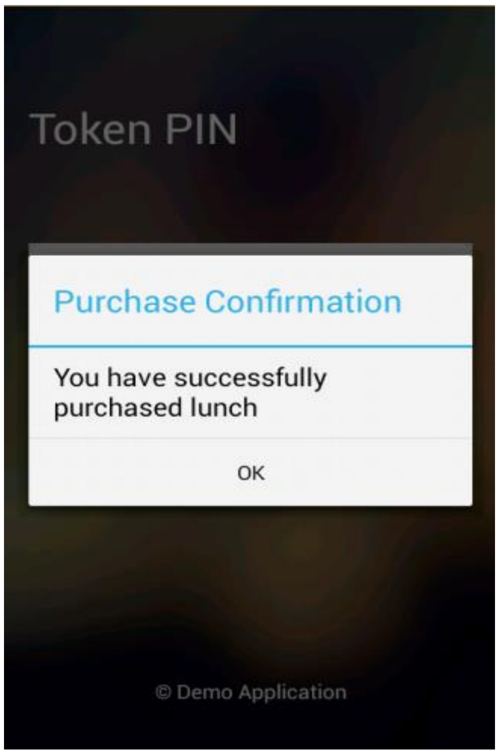
4. Upon selecting the item the user is sent a token. When the delivery boy comes to deliver the product, the user enters his/her bank credentials.





5. The token assigned to the user is entered on the delivery boy's device.

6. The user is then prompted with a message that the transaction is successful. The user activity ends here.



You have successfully purchased lunch

5.11 Conclusion

This application is developed to show proof of concept and also depicts a good idea of an efficient payment system in a close environment. Next chapter concludes the thesis and discusses the future directons

Chapter 6

6. Conclusion and Future Direction

Over the years technology has covered every aspect of human life and has transformed into a utility. Aim of technology is to facilitate humans as much as possible so it's moving towards integrating real life critical areas such as education, health, finance and many others with emerging technologies**.** There has been so much overlapping among various fields of IT that one cannot clearly demarcate the boundary of any technology. So now when we talk about throughput, efficiency and security of any system we just can't look at one component, we have to take in account the share of all modules involved in a finished product **.**We need to utilize different technologies in a way that they combine to give a better product. For example online banking on Smart phones; both mentioned technologies have benefits and problems of their own so we efficiently incorporate both to gain as much throughput as we can. On one hand it benefits the users; and on the other hand there have been evil elements involved which provide a greater harm by exploiting the vulnerabilities of such systems. Financial indiscretion in e-commerce is becoming a major concern for individual users as well as for the organizations worldwide. Cyber criminals are gradually launching well-organized and effective attacks by exploiting the vulnerabilities in existing architectures. Credit card information is being hijacked, financial institutions are being attacked and money is being stolen online by cyber thieves without even entering the bank .Taking in account all the above facts, there is a need of a secure e-commerce solution which does not only facilitate users' financial needs but also fulfills the security parameters compulsory in any transaction. To do that we have to accommodate many different problems like mobility and ease of access for users so we suggest a financial solution based on smart phones. Conventional financial solution lack extensibility, openness, privacy, and cost effectiveness **.** We realize that Smart phones are prone to attacks too so if we want to use them for financial transactions, we need a highly efficient and secure design .

For large multi-national organizations there are a lot of business transactions within & outside the organization which requires hardware tokens, PIN or access codes to acquire the resource . What if personal information of an employee is stolen or one of the insiders tries to exploit the system weaknesses like stolen card information or mobile devices. So why not make the whole organization transactions and resources strictly need to know basis and anonymous which are highly secure and easy to use. So we recommend a financial solution that does not require hardware tokens or physical presence and is based on smart phones focusing on close networks

We propose a **S**ecure **P**rotocol for **F**inancial **T**ransactions **SPFT**- based on smart phones. All transactions are performed by smartphone and a user does not have to carry cash or cards. Entities involved in a process are; Client-C, Merchant-M, Bank- B, Payment Gateway– PG.SPFT ensures privacy, authentication and integrity of all entities, provides anonymity and mechanism to resolve disputes and is formally tested before implementation. And to achieve that we have used low cryptographic operations, less reliance on banks, an honest payment gateway, Digital Certificate & time stamping. **[6]**. Formal techniques are an efficient way to verify the security specifications of a system. We have formalized the authentication and secrecy properties of our protocol.

We have suggested a radical secure payment protocol to make daily life transactions easy and secure for users, where client and merchant does not need to blindly rely on financial service providers. Each entity has a part of whole transaction; all entities need to put their part to make an effective transaction. Client's identity is hidden from merchant and bank does not need to know what is bought. Client places a request with merchant and requests bank to reserve money for specific deal after it is routed to the payment gateway to look over the transaction and locks the digital money by two-factor authentication and authorizes both parties to complete the transaction. After conformation by both client and merchant, funds are transferred to the merchant's account. It fulfills all the security parameters required in a payment protocol like secrecy, authentication and conflict resolution and to prove this we have formally tested the code by AVISPA.

For future work we will modify the protocol involving two different banks and formally prove it by Model checking to draw the comparison and to employ it in a cloud environment. As an end result, we state that suggested protocol is flexible and extensible to all environments. Besides it also ensures the secrecy of personal information as well as the anonymity of user

References

Kungpisdan, S., Srinivasan, B., and Dung Le. P., 2004. “A Secure Account-Based Mobile Payment Protocol” In (ITCC’04), Proceedings of the International Conference on Information Technology: Coding and Computing

Liu, J., Liao, J., Zhu, X., 2005. “A System Model and Protocol for Mobile Payment” .In (ICEBE’05), Proceedings of the IEEE International Conference on e-Business Engineering

T´ellez, J., Camara, J., 2007. “An Anonymous Account-Based Mobile Payment Protocol for a Restricted Connectivity Scenario” In (DEXA’03), 18th International Workshop on Database and Expert Systems Applications

Vilmos, A., Karnouskos, S., 2003. “SEMOPS: Design of a new payment service” In 14th international workshop on Database & Expert Systems Applications

Abdel-Hamid, A., Badway, O., Aboud, M., 2012. “SEMOPS+SIP+ECC: Enhanced secure mobile payments” In (INFOS2012), 8th international conference on Informatics & systems

Xueming, W., Nan, C., 2009, “Research of security mobile payment protocol in communication restriction scenarios”. In international conference on computational intelligence & security

Chang, C., Yang.J., Chang,k., 2012. "An Efficient and Flexible Mobile Payment Protocol”. In (ICGEC ’12) Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference

Ahamad, S., Sastry, N., Udgata, K., 2012. "Enhanced Mobile SET Protocol with Formal Verification”, In (ICCCT ‘12), Third International Conference of Computer and Communication Technology

Nicholas O'Shea, 2008. “Using Elyjah to analyse Java implementations of cryptographic protocols, In Joint Workshop on Foundations of Computer Security”, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security (FCS-ARSPA-WITS), pages 221-226

Nicholas O'Shea, 2010 . “Verification and Validation of Security Protocol Implementations”, PhD thesis, School of Informatics,University of Edinburgh (UK)

Lawrence C. Paulson, 1999. “Inductive analysis of the Internet protocol TLS”, In ACM Transactions on Information and System Security, 2(3):332-351.

David Basin, Sebastian M¨odersheim, Luca Vigan`o, 2005, “OFMC: A symbolic model checker for security protocols,” International Journal of Information Security,

David Basin, Sebastian M¨odersheim, Luca Vigan`o, 2003, “Constraint Differentiation: A New Reduction Technique for Constraint-Based Analysis of Security Protocols”, CCS, ACM Press.

Alessandro Armando, David Basin, and others, 2005, “The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications”,CAV’05, LNCS 3576.

Gavin Lowe, 1995, “An attack on the Needham-Schroeder public-key authentication protocol”, Information Processing Letters, 56(3):131 - 133.

Gavin Lowe, 1996, “Breaking the Needham-Schroeder public-key protocol using FDR”, In 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS), pages 147-166,.

Sarah Meiklejohn, C. Chris Erway, Theodora Hinkle, and Anna Lysyanskaya, 2010, “ZKPDLA: language based system for efficient zero-knowledge proofs and electronic cash”, In 19th USENIX Conference on Security, pages 193-206.

John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern, 1998, “Finite-state analysis of SSL 3.0”, In 7th USENIX Security Symposium (SSYM), pages 201-216

Roger M. Needham and Michael D. Schroeder, Using encryption for authentication in large networks of computers, Communications of the ACM, 21(12):993-999,

Kazuhiro Ogata and Kokichi Futatsugi, 1978, “Equational approach to formal analysis of TLS”, In 25th IEEE International Conference on Distributed Computing Systems (ICDCS), pages 795-804, 2005.

Dave Otway and Owen Rees, 1987, “Efficient and timely mutual authentication”, In ACM Operating Systems Review, 21(1)

Isaac, J., Camara, J, 2007. "An Anonymous Account-Based Mobile Payment Protocol for a Restricted Connectivity Scenario," In (DEXA '07) Database and Expert Systems Applications

Avalle, M., Pironti, A., Sisto, R., 2014. "Formal verification of security protocol implementations: a survey". Journal of Formal Aspects of Computing Volume 26, Issue 1, pp 99-123 2014

Secure Electronic Transaction (SET) Protocol, http://www.isaca.org/Journal/Past-Issues/2000/Volume-