INCREASING TRUSTWORTHINESS OF SECURITY CRITICAL APPLICATIONS USING TRUSTED COMPUTING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

YUSUF UZUNAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILISOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JANUARY 2014

INCREASING TRUSTWORTHINESS OF SECURITY CRITICAL APPLICATIONS USING TRUSTED COMPUTING

Submitted by **YUSUF UZUNAY** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems, Middle East Technical University** by,

Prof. Dr. Nazife Baykal _____
Director, **Informatics Institute**

Prof. Dr. Yasemin Yardımcı Çetin _____
Head of Department, **Information Systems**

Prof. Dr. Nazife Baykal _____
Supervisor, **Information Systems, METU**

Assoc. Prof. Dr. Kemal Bıçakcı _____
Co-Supervisor, **Computer Engineering, TOBB ETU**

Examining Committee Members:

Assoc. Prof. Dr. Altan Koçyiğit _____
Information Systems, METU

Prof. Dr. Nazife Baykal _____
Information Systems, METU

Assoc. Prof. Dr. Ali Doğanaksoy _____
Institute of Applied Math., METU

Assist. Prof. Dr. Erhan Eren _____
Information Systems, METU

Assoc. Prof. Dr. Bülent Tavlı _____
Electrical and Electronics Eng., TOBB

Date: 28.01.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Yusuf UZUNAY

Signature : _____

ABSTRACT

INCREASING TRUSTWORTHINESS OF SECURITY CRITICAL APPLICATIONS USING TRUSTED COMPUTING

Uzunay, Yusuf
Ph.D., Department of Information Systems
Supervisor: Prof. Dr. Nazife Baykal
Co-Supervisor: Assoc. Prof. Dr. Kemal Bıçakcı

January 2014, 192 pages

In this thesis work, we aim to increase the trustworthiness of security critical applications by utilizing trusted computing technologies. We focus on two case applications; authentication proxy systems and e-voting systems. Our first case application is authentication proxy systems which store users' sensitive credentials and submit them to the servers of the service providers on their behalf. To increase the trustworthiness of authentication proxy systems, we propose Trust-in-the-Middle a trusted platform module based proxy system which ensures that user credentials are securely stored and submitted without disclosing them even if the proxy is

compromised. We use remote attestation to guarantee that all critical operations on the proxy are performed securely and credentials are cryptographically protected when they are not in trusted platform module supported isolation. For our second case application, we propose Trusted3Ballot, a trusted computing based three-ballot e-voting system to increase the trustworthiness of poll-site e-voting systems. In our second proposal, we put forth an election process where security critical issues are processed in software applications attested by TPM. By integrating three-ballot voting mechanism into an electronic voting system secured by trusted platform module, we not only satisfy some contradictory requirements of voting such as providing individual and universal verifiability without causing vote trade, but also give users and the relevant parties the ability to attest the trustworthiness of the running software at each phase of the election. The analysis of Trusted3Ballot reveals that significant improvements to the three-ballot system are provided in terms of both security and usability.

Keywords: Trusted Computing, proxy, e-voting, Three-ballot Voting, TPM

ÖZ

GÜVENİLİR BİLİŞİM TEKNOLOJİLERİNİ KULLANARAK GÜVENLİK KRİTİK UYGULAMALARIN GÜVENİLİRLİĞİNİ ARTTIRMAK

Uzunay, Yusuf
Doktora, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Prof. Dr. Nazife Baykal
Ortak Tez Yöneticisi: Doç. Dr. Kemal Bıçakcı

Ocak 2014, 192 sayfa

Bu tez çalışmasında, güvenilir bilişim teknolojileri kullanılarak güvenlik kritik uygulamaların güvenilirliğinin arttırılması hedeflenmektedir. Temel olarak iki örnek uygulama üzerine odaklanılmaktadır. Bunlar doğrulama vekil sunucuları ve elektronik oylama sistemleridir. İlk örnek uygulamamız olan doğrulama vekil sunuları, kullanıcıların parolalar benzeri gizli bilgilerini saklamakta ve bunları kullanıcı adına ilgili servis sağlayıcıların sunucularına göndermektedir. Doğrulama vekil sunularının güvenilirliğini arttırmak için, kullanıcıların parolalar gibi gizli bilgilerinin vekil sunucu üzerinde güvenli bir şekilde saklandığını ve yine güvenli bir şekilde hedef sunuculara iletildiğini garanti altına alan güvenilir bilişim teknolojileri tabanlı "Ortadaki Güven" isimli bir sistem önerilmektedir. Vekil sunucu üzerinde

çalışan tüm kritik işlemlerin güvenli bir şekilde çalıştığını ve kullanıcıların gizli bilgilerinin DRTM destekli koruma altında olmadığı zamanlarda kriptografik koruma altına alındığını garantilemek için uzaktan kanıtlama yöntemi kullanılmaktadır. Tez çalışmamızdaki ikinci örnek uygulama olan e-oylama sistemleri ile ilgili olarak ise, güvenilir bilişim teknolojileri tabanlı üç oy pusulalı elektronik oylama sistemi önerilmektedir. Bu öneride, güvenlik kritik tüm uygulamaların TPM tarafından uzaktan kanıtlama yöntemi kullanılmak suretiyle doğrulandığı bir seçim süreci ortaya koyulmaktadır. Üç oy pusulalı oylama sisteminin güvenilir bilişim tabanlı bir elektronik oylama sistemine entegre edilmesi ile hem oylama sistemlerinin önemli fakat birbirleriyle çelişkili olarak görülen, oy ticaretine yol açmadan bireysel ve evrensel doğrulama gereksinimleri sağlanabilmekte, hem de oylama süreci dahil seçim sürecinin her safhasında kullanıcılara veya ilgili paydaşlara çalışan yazılımların güvenirliliğini uzaktan kanıtlama yöntemi ile doğrulama imkanı sunulmaktadır. Önerilen sistem aynı zamanda klasik üç oy pusulalı sistemin birçok güvenlik ve kullanışlılık problemine de çözüm getirmektedir.

Anahtar Kelimeler: Güvenilir Bilişim, E-oylama, Üç Oy Pusulalı Oylama, TPM

DEDICATION

This thesis is dedicated to:

My Wife Esin and My Son Yiğit

ACKNOWLEDGEMENT

It is a pleasure for me to express my sincere gratitude to my co-supervisor Dr. Kemal Bıçakcı for his patience, encouragement and guidance throughout the study. I greatly appreciate his helps when I take my first steps in academic world and always being with me in every phase of this Thesis.

I would like to also express my gratitude to my supervisor Dr. Nazife Baykal for her support, guidance, helps and suggestions throughout my research.

Also, I owe much to the committee members Dr. Altan Koçyiğit, Dr. Erhan Eren, Dr. Ali Doğanaksoy and Dr. Bülent Tavlı for helpful comments and discussions.

I would like to thank to my colleague Davut İncebacak for providing assistance and motivation in every phases of my Ph.D. work and I would also like to express my special thanks to Dr. Fatih Hasdemir and Dr. Fuat Oktay for their support and encouragement during my study.

I will also never forget the unending support my family have provided me with during all the hard times.

Finally, I would like to express my very special gratitude to my wife Esin, who beared to many difficulties and sacrifices, in order to help me to finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

AC	: Authenticated Code
AIK	: Attestation Identity Key
ASP	: Authentication Service Provider
BIOS	: Basis Input Output System
CA	: Certificate Authority
CC	: Common Criteria
C-MAS	: Cloud Mutual Authentication Scheme
CPU	: Central Processing Unit
CRTM	: Core Root of Trust for Measurement
DAA	: Direct Anonymous Attestation
DB	: Database
DNS	: Domain Name System
DRTM	: Dynamic Root of Trust for Measurement
DumCred	: Dummy Credentials
EAL	: Evaluation Assurance Levels
EDS	: Encryption Decryption Signature
EK	: Endorsement Key
encCredwithPal	: Encrypted Credentials with PAL public key
encCredwithPM	: Encrypted Credentials with PM public key
encNewCredwithPal	: Encrypted New Credentials with PAL public key
EncSenData	: Encrypted Sensitive Data with PAL public key
ICH	: Input Output Controller Hub

IDS	: Intrusion Detection System
IMA	: Integrity Measurement Architecture
IPS	: Intrusion Prevention System
ISA	: Industry Standard Architecture
KCM	: Key Cache Manager
LPC	: Low Pin Count
LT	: Lagrande Technology
MasPass	: Master Password
MITM	: Man-in-the-Middle
MK	: Migratable Keys
MLE	: Measured Launch Environment
NewCred	: New Credentials
NMK	: Non-Migratable Keys
ObC	: On-Board Credentials
OldCred	: Old Credentials
OPIE	: One-Time Passwords in Everything
OS	: Operating System
OTP	: One Time Password
PAL	: Piece of Application Logic
PALpriv	: Private Key of PAL
PALpub	: Public Key of PAL
PassList	: Password List
PCH	: Platform Controller Hub
PCR	: Platform Configuration Register
PM	: Proxy Module
PMpriv	: Private Key of Proxy Module

PMpub	: Public Key of Proxy Module
RSA	: Rivest Shamir Adleman
RTM	: Root of Trust for Measurement
sealedPassList	: Sealed Password List
sealedPALPriv	: Sealed Private Key of PAL
sealedPMPub	: Sealed Public Key of Proxy Module
SecPhrase	: Secret Phrase
SenData	: Sensitive Data
SHA	: Secure Hash Algorithm
SKINIT	: Secure Kernel Init
SL	: Secure Loader
SLB	: Secure Loader Block
SML	: Stored Measurement Log
SMS	: Short Message Service
SMX	: Secure Mode Extensions
SRK	: Storage Root Key
SSH	: Secure Shell
SSL	: Secure Socket Layer
SVM	: Secure Virtual Machine
TBB	: Trusted Building Block
TCB	: Trusted Computing Base
TCG	: Trusted Computing Group
TNC	: Trusted Network Connect
TPM	: Trusted Platform Module
TTP	: Trusted Third Party
TXT	: Trusted Execution Technology

UDS	: Usability Deployability Security
VGA	: Video Graphics Array
VM	: Virtual Machine

CHAPTER I

INTRODUCTION

In recent years, especially improvements in software and network technologies enable users to interact with a growing number of operating systems and software applications ranging from standard desktop applications to various mobile applications. Today, people are continuously connected to internet through their mobile phones and tablets and use a growing number of software applications on these platforms each day. Especially due to the high penetration rates of social networking tools in many countries including Turkey, an internet based social life has become a part of our daily living activities.

Although this technological shift improves the quality of our lives, the diversity of applications and software platforms used creates crucial security concerns that should be taken into consideration. According to a recent Internet Security Report [1], hackers use social networks heavily to mount attacks e.g. by leveraging news-feeds to spread spams or by providing shortened URLs that hide malicious links. Another major issue emphasized in the report is the increase in the mobile threats. It is projected that mobile systems will be increasingly targeted as they are used more for financial transactions. All these data indicate that the volume of security issues regarding software applications on client side will keep increasing in the coming years.

I.1 Problem Definition

Establishing the security of software applications emerges as one of the biggest challenges we face today, as it depends on not only the security of software application itself and but also the security of a long list of external entities such as operating system, drivers, other installed software applications on user's computer or the entities on the network system interacting with the software. Therefore; we, in the following sub sections, address the possible problems by classifying them into two categories; internal risks and external risks:

I.1.1 Internal Risks

We can define internal risks as the risks having occurred because of the possible vulnerabilities in software code created during the development phase. In [2], top software vulnerabilities are identified under three categories:

Insecure Interaction between Components: This points out the vulnerabilities having occurred during the data exchange between the modules, programs, processes, threads or separate components of software i.e. SQL Injection, OS Command Injection, Cross-site scripting and etc.

Risky Resource Management: This points out the vulnerabilities having occurred because of the improper management of important system resources used by the software i.e. buffer copy without checking size of input, improper limitation of a pathname to restricted directory, download code without integrity check, incorrect calculation of buffer size, uncontrolled format string and etc.

Porous Defenses: This points out the defensive techniques that are misused, abused or just ignored i.e. missing authentication for critical function, missing authorization, missing encryption of sensitive data, execution with unnecessary privileges, incorrect permission assignment for critical resources and etc.

I.1.2 External Risks

External risks can be examined in three categories; risks in trusted computing base, risks in network and social engineering risks.

Risks in Trusted Computing Base: The security of software applications may be affected by various entities that share the same computer system such as the underlying operating system, firmware, other software applications running on the same platform, drivers installed, browsers and so on. All of those entities that can have an effect on the security of the software form the trusted computing base (TCB) of the software application. When we look at the TCB of a software application, we see that there is a long list of entities that should be secured and trusted. If we think all these entities as a part of a security chain, each of them should be secured one by one in order to establish the security of the whole system which is as secure as the weakest link of the chain. Keyloggers, screen-scrappers, malicious codes such as viruses, Trojan horses, backdoors having infected the operating system or malicious browsers are the examples of well-known attacks on the computer systems. (See Section IV.1 for a detailed threat analysis)

Network Risks: Software applications can also be affected by various network based attacks including eavesdropping, pharming, man-in-the-middle attacks.

Social Engineering Risks: Human based errors can also put the secure execution of a software application under risk such as revealing sensitive data to malicious entities,

clicking untrusted links, ignoring the SSL certificate warnings, ignoring browser security warnings and etc.

I.1.3 Trustworthiness of Software

As well as providing the security of the software applications, another important issue is how to establish the trustworthiness of them. Trustworthiness of a system is defined as being trusted to satisfy its specified requirements with some quantifiable measures of assurance [3]. Based on this definition, if we try to explain how we can establish the trustworthiness of a security critical software application (i.e. e-banking, e-voting, e-commerce and etc.), we can say that we first need to provide the security of the software as the main requirement and secondly find ways to prove the users that the software has really performed its functions in a secure fashion.

We, in this dissertation, try to increase the trustworthiness of security critical applications using trusted computing technologies which help us to run security critical functions of a software application in a secure and isolated environment created by Trusted Platform Module (the core component of trusted computing) and also to prove this to the users by a special operation called Remote Attestation.

We focus on two case applications which we believe as two of the most critical systems in terms of security and trustworthiness, authentication proxies and poll-site e-voting systems.

I.2 First Case Application: Authentication Proxy Systems

As we have seen in previous sections, due to various internal and external risks, users' software applications are under threat by several different attacks. Although

there is an intensive work to secure software applications on user side, the expected progress towards more secure platforms is still not on the horizon. There are many reasons for this deficiency which include:

- Proposed solutions could not catch up the increasing rate of new technologies and online services offered to users.
- The variety of different environments increases the number of platforms that need to be secured.
- Frequent software updates, new browser technologies, etc. make it more challenging to bootstrap trust.
- With wireless and mobile technologies, it becomes more difficult to implement appropriate security measures against threats like real-time network attacks.
- The highly decentralized structure complicates the security management.

It is a well-known fact that managing security is easier and cheaper in centralized systems. Central Firewalls, IDS and IPS systems, central antivirus and antispam gateways, central log and update servers are among the widely deployed central systems for this reason. With these systems, security management focuses on a few central systems instead of trying to secure each client system separately.

The important question at this point is whether we can also implement centralized solutions to increase the security of users' sensitive credentials in highly distributed environments as we live in today. In fact, the idea here is not a new one. When we examine the literature, we see that there is a considerable amount of work on authentication proxy systems which act as an intermediary between clients and servers and undertake the sensitive credential input operation from the users during authentication [4-9]. With a proxy server, users are not required to enter the (whole) sensitive credential on the client side and hand over this operation to the proxy server.

The proxy intercepts the connection, inserts the credentials into correct fields and then submits the page to the (target) server. By this way, most of the client-side attacks can be mitigated.

Since authentication proxy systems stores and processes user's sensitive credentials, the security and the trustworthiness of those systems have paramount importance. We, in this thesis, try to increase the trustworthiness of authentication proxy systems utilizing trusted computing technologies.

I.3 Second Case Application: Poll-Site Electronic Voting Systems

On the way going towards e-democracy, e-voting stands out as a core milestone receiving an amazing attention and interest in academia for several years. E-voting is deemed as the technological opportunity to reduce vote counting time, provide evidence that a vote has been correctly accounted, reduce fraud, remove errors in filling out ballots and improve system usability especially for people with special needs [10,11]. However; it also poses several security concerns due to the nature of core democratic principles which have many contradictories i.e. voter authenticity and vote anonymity, providing a vote-counting proof while preventing vote trade, allowing electronic voting but avoiding voting coercion, guaranteeing the uniqueness of the vote in decentralized voting, allowing vote automation while providing vote materialization and ensuring auditability in a software or hardware environment that could malfunction [10].

As well as dealing with the given conflicting requirements, social concerns like people's trust on electronic voting systems emerges as another critical factor that acts as a barrier preventing those systems being widely deployed and used.

In a general perspective, voting has four different stages [12]:

1- Setup-Stage: In this stage, voting procedures, candidates, voters and authorities' eligibility criteria, counting and ballot validity rules are determined. Registration and tally authorities are assigned and eligible candidates are registered.
2- Registration stage: This is the stage where eligible voters registers themselves in order to be able to vote in the Election Day. The eligibility criteria applied are determined in the previous steps.
3- Voting Stage: Voting stage involves the following steps:
    a. Voter authentication: Before being able to cast a vote, an authentication is first performed according to the registration list created in registration step.
    b. Vote registration: Voter takes an empty ballot from the poll site workers and registers his vote in the ballot in a private and secure location.
    c. Ballot Casting: Voter, then, puts his ballot into a sealed ballot box. Since there is not any information belonging to user on the ballot, anonymity is provided inside the ballot box which is sealed to be opened at the end of the voting period by official election workers.
4- Tally Stage: This is the final stage where all ballots are processed to find the election results. Tallying stage includes the following steps:
    a. Ballot Collection: At the end of the voting period, ballot boxes are opened and all the ballots are collected by tallying authority.
    b. Ballot verification: All ballots are passed through an eligibility check whether they are valid or not according to the rules determined in setup stage. The ballot that are not valid does not go into tabulation step.

c. Vote tabulation: Each valid ballots are counted and tabulated according to the counting rules. The results are then published.

In an electronic election scenario, most of those phases explained are tried to be automated. Now, in order to see the threats against an e-voting system, we give possible attacks against each stage in Table 1:

Table 1: Possible Attacks According to the Voting Stages

Voting Stage	**Attack Definition**	**Type of Attack**	**Source of Attack**	**Assumptions**
Setup Stage	Modify/Delete Candidate Information	Active Attack	Internal	System is not open to outside.
Setup Stage	Acquire personal data	Passive Attack	Internal	System is not open to outside.
Registration Stage	Create/Modify/Delete registration information i.e. Ineligible voter can register himself/herself	Active Attack	Internal/ External	There is an online e-voting registration system.
Registration Stage	Identity Theft (Adversary can register himself by impersonating another person)	Active Attack	Internal/ External	There is an online e-voting registration system.
Registration Stage	Acquire personal data	Passive Attack	Internal/ External	There is an online e-voting registration system.
Registration Stage	Denial of Service by making online registration program out of service.	Active Attack	Internal/ External	There is an online e-voting registration system.
Registration Stage	Acquire user credentials such as passwords	Passive Attack	Internal/ External	Malware in the registration software
Voting Stage (Voter Authentication)	Identity Theft by using some other's smart card	Active Attack	External	For all e-voting systems with smartcard authentication
Voting Stage (Voter Authentication)	Credential Misuse – Internal system operator can give some registered credentials to someone else	Active Attack	Internal	For all e-voting systems with password authentication
Voting Stage (Voter Authentication)	Denial of Service Attack by making the credential database out of service	Active Attack	Internal	For all e-voting systems that make authentication

				according to a credential db.
Voting Stage (Voter Authentication)	Man-In-The-Middle-Attacks (Eavesdropping, Replay Attacks)	Passive/ Active Attacks	External	Remote voting i.e. internet voting is in place and there is not an encryption or there is a weak encryption in the communication channel.
Voting Stage (Voter Authentication)	Acquire user credentials such as passwords	Passive Attacks	External	Remote voting in place and Malware in client's device
Voting Stage (Ballot Casting)	Create/Modify/Delete Electronic Ballot by a malicious voter (can vote more than once), malicious code or a third party	Active Attacks	External/ Internal	In remote voting both client side and server side attacks, In polling-booth e-voting application e-voting machine and server side attacks are possible.
Voting Stage (Ballot Casting)	Disturb the anonymity of the voter and divulge the owner of the votes	Passive Attacks	External/ Internal	-
Voting Stage (Ballot Casting)	Denial of Service Attacks preventing ballot casting	Active Attacks	External/ Internal	-
Tally Stage	Create/Modify/Delete Electronic Ballots	Active Attacks	Internal	We assume that tally servers are accessible only from inside.
Tally Stage	Modification on the total number of the votes by an adversary or malicious code.	Active Attacks	Internal	We assume that tally servers are accessible only from inside.
Tally Stage	Denial of Service preventing tallying operation.	Active Attacks	Internal	We assume that tally servers are accessible only from inside.

In table 1, a general threat model is given including remote and poll-site electronic voting systems both. However; in real life implementations, we see that poll-site electronic voting systems are more preferable due to the fact that it exhibits less

security issues and can be performed under physical control. In poll-site voting systems, the most security critical entity is the e-voting machines deployed.

E-Voting Machines

As it is often the case that the wider the topology of a system is, the more security risks it has, internet and remote voting come out as a big challenge and still a very hot topic for all the academicians, governments, companies and for all the other stakeholders. When the previous e-voting work in different countries is examined, we see that most of them preferred using poll-site e-voting mechanisms [13].

The main underlying reasons can be listed as follows:

- To provide security in controlled environment is always easier.
- We can ensure that nobody can coerce the user in the time of the voting.
- Because the e-voting systems are not connected to the internet, possible attacks are minimized.
- By designing security improved black-box solutions, the risks with the operating systems and the other software running on the existing desktop systems are prevented.
- Authentication can be healthier and more secure since we can implement some kind of extra physical authentication schemes in the entrance of the polling-booth.

But the question is whether the existing black-box e-voting solutions do provide enough security, reliability and trust. As it is seen in [14], although black-box e-voting solutions seem to provide better security, they have various security and reliability problems. The point that should not be missed is that they use some kind of software application and always exposed to the same problems with the available

software applications. In order to more focus on the issue, let us list the possible software problems associated with e-voting machines and software.

- The software developer can make undeliberate errors in the code.
- The software developer can insert some malicious codes to the software.
- The software developer can leave some kind of backdoors in the software and not only can make use them in the time of voting but also distribute this flaw to the others and able to make a deep effect on the election results. For example in the time of voting, the votes can be configured to be tripled when the user presses 3 special keyboard buttons at the same time or user can change the previous vote results.
- A malicious user can find some kind of flaws of the software and exploit it. Dan Wallach, in [15], discussed the issue and showed that e-voting machines could not be well safeguarded and somehow a malicious user could have an access to machine and copied and analyzed the software by utilizing various exploiting tools and reverse engineering methodologies.

In this dissertation, as a second case application, we try to increase the security and trustworthiness of poll-site electronic voting systems using trusted computing technologies.

I.4 Scope of the Thesis

In this thesis work, we propose two systems; Trust-in-the-Middle [16], a trusted computing based authentication proxy system and Trusted3Ballot [17], a trusted computing based Poll-Site Three-Ballot E-Voting system.

In our first proposal, we present the design and implementation of a trusted authentication proxy system called Trust-in-the-Middle. By utilizing trusted computing technologies and its core functionalities, we make all the security critical

software processing the users' credentials on proxy secure by adding them into a trust chain protected by Trusted Platform Module (TPM). All security critical operations are processed in a secure and isolated environment created by dynamic root of trust for measurement functionality of TPM. Users' sensitive credentials are never moved out from this isolation without being encrypted. We propose several protocols that show how Trust-in-the-Middle registers and authenticates user and how it stores and processes the credentials in a secure fashion. We not only maintain the security even if the proxy system is compromised but also ensure users that their credentials are stored and processed securely in TPM protections by using remote attestation.

In our second proposed system, we present a Trusted Computing based Poll-Site Three-Ballot E-Voting System (Trusted3Ballot). The main goal in the design of paper based standard Three-Ballot system was to provide an end-to-end auditable voting system in a simple way without use of cryptography to bolster voter confidence in the system. However, later it was shown that this system has significant security and usability problems. To solve these problems, we propose Trusted3Ballot; an electronic Three-Ballot based voting system which uses trusted computing technology. One notable feature of the proposed system is the use of TPM remote attestation property to address a number of trust and security problems. The analysis of our proposal reveals that significant improvements to the Three-Ballot system are provided in terms of both security and usability.

I.5 Outline of the Thesis

This dissertation is composed of 6 chapters. First chapter is the introduction. In this chapter, we first try to define the problem, our two case applications, authentication proxy systems and three ballot e-voting systems, which we have focused in this thesis work, are then introduced. After defining our scope, we finally give the outline of the thesis.

Chapter II is dedicated for background information. Since trusted computing and TPM are the core elements of our thesis work, this chapter heavily gives information on TPM architecture, key structure, credentials, functionalities, and the technologies used as root of trust for measurement.

Our first proposed system, Trust-in-the-Middle, a trusted computing based authentication proxy system, is introduced in chapter III in detail. First the related work is discussed. Then, the system model, objectives and assumptions are explained. Overview and architecture of the system are given. After presenting the PAL, the security sensitive code block that we would like to execute in TPM protections, we explain main and auxiliary protocols used in the system framework. The subsections including implementation details and performance evaluation then follow. We conclude the section by analyzing the security of the proposed system and a discussion.

We carry out a detailed security and usability analysis of Trust-in-the-Middle in Chapter IV. We first give possible threats under a threat model. Then, we discuss how Trust-in-the-Middle addresses those threats. Finally, we make a usability-deployability-security comparison of Trust-in-the-Middle with previous 20 authentication systems and explain in detail where Trust-in-the-Middle has better and worse functionalities.

Our second proposed system, Trusted3Ballot, a trusted computing based electronic three ballot system, is explained in Chapter V. In this chapter, we first introduce three ballot scheme. Then, we discuss some security problems of three ballot in a threat model. After giving previous work, we give the details of the proposed system including the introduction of voting machine used, design principles, the activities in preparation step and how the system works and voting process is carried out in

Election Day. We, then, present our prototype trusted3ballot software. We sum up the section by making security and usability analysis and giving a concluding discussion.

Finally, we wrap up our thesis with a conclusion and future work part in chapter VI.

CHAPTER II

BACKGROUND

In this chapter, we provide background information on TPM, its architecture and core functionalities.

II.1 Trusted Platform Module

Trusted Platform Module (TPM), the core component of Trusted Computing [18], is a chip attached directly to the motherboard of the computer and stores keys, passwords and digital certificates. It has cryptographic capabilities such as RSA key generation, encryption, signing and verification, secure random number generation and SHA1 hashing.

II.2 TPM Architecture

The architecture of TPM is illustrated in figure 1. All the components of a TPM should be trusted to work in a proper fashion which is intended to be guaranteed by Common Criteria (CC) evaluation and Evaluation Assurance Levels (EAL) [19-21]. Regarding the compliance to Tamper Protection standards, a TCG compliant TPM should be able to achieve FIPS PUB-140-2 certification [22, 23].

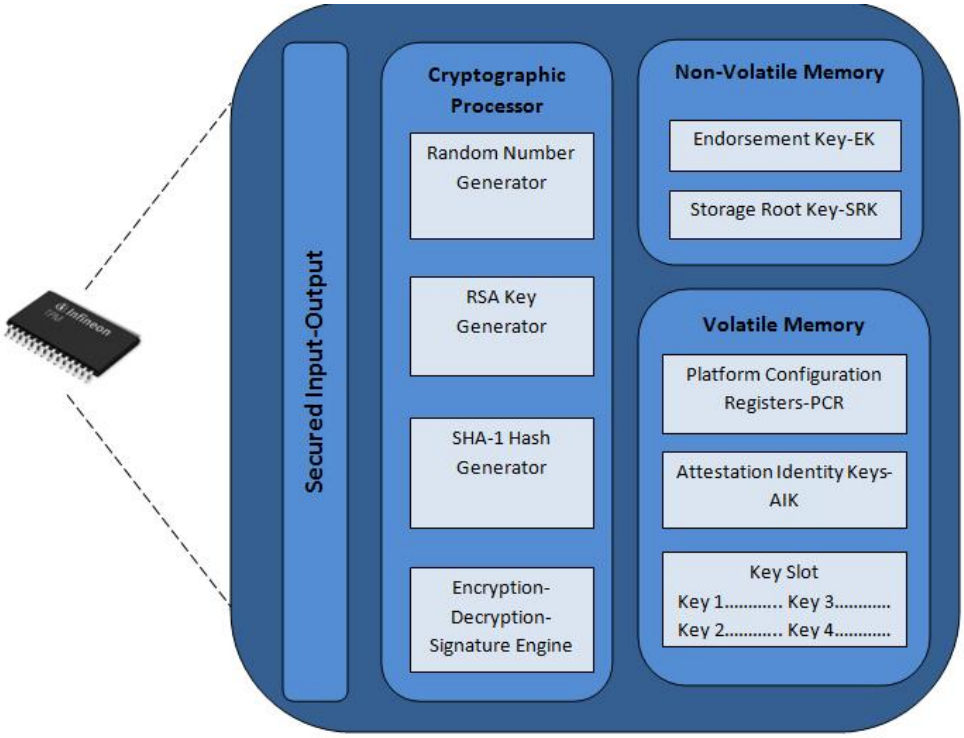


Figure 1: Architecture of TPM

Now, let us define the tasks of those components in the TPM Architecture given in figure 1:

II.2.1 Secure Input-Output and LPC Bus

LPC bus connected to the Southbridge of the mother board (see the following figure) is used by TPM to carry out its I/O operations. In 1998, Intel first introduced LPC bus as a substitute for the Industry Standard Architecture (ISA) bus. Although physically different from ISA replacing the 16-bit-wide, 8.33 MHz ISA bus with a 4-bit-wide bus operating at 4 times the clock speed (33.3 MHz), it is very similar to ISA in terms of software [24].

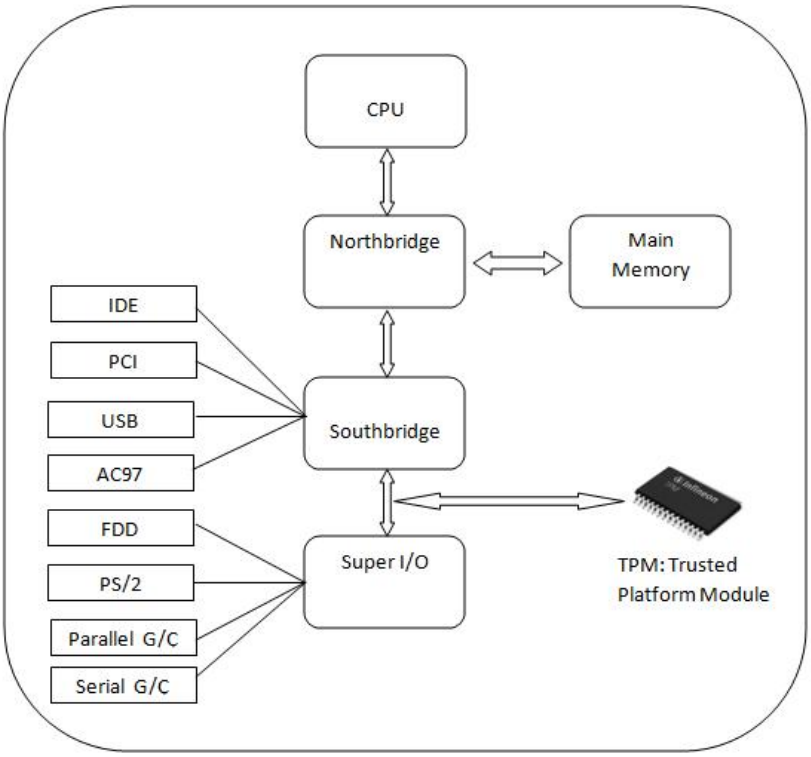


Figure 2: TPM Connection to Motherboard

To look a little bit closer to LPC, it requires only seven signals which make routing on modern motherboards easier. Compared to ISA equivalent, an integrated circuit utilizing LPC needs 30-72 fewer pins. To further ease integration, the clock rate was matched that of PCI. Furthermore LPC is designed to be motherboard-only bus. For this reason, no connector is defined no LPC peripheral daughterboards are available [24].

In Intel systems, the Southbridge functions as an I/O controller hub (ICH) or a platform controller hub (PCH). It implements the slower capabilities of the motherboard in a northbridge/southbridge chipset architecture. Southbridge is not directly connected to CPU. Instead, the Northbridge links it to the CPU [25].

II.2.2 Cryptographic Processor

Cryptographic processor located in TPM is responsible for carrying out all cryptographic operations. To prevent software attacks, a hardware encryption protection is implemented. There are 4 main subcomponents of Cryptographic Processor:

- Random Number Generator
- RSA Key Generator
- SHA-1 Hash Generator
- Encryption-Decryption-Signature Engine

Random Number Generator is used to generate true random numbers which are a sequence of numbers or symbols that lack any pattern [26]. RSA Key Generator is responsible for generating RSA keys. SHA-1 Hash Generator calculates SHA-1 hash values from a messages to produce a 160-bit digest. And finally EDS (Encryption-Decryption-Signature) engine is used to do all encryption, decryption and signature operations.

II.2.3 Memory

There are two types of memory in TPM. One of them is non-volatile memory and the other is volatile memory.

Non-Volatile Memory: also known as persistent memory, non-volatile memory of TPM is used to hold some special date such as special keys in case of a power cut. Non-volatile memory incorporates two important keys: Endorsement Key (EK) and Storage Root Key (SRK).

Volatile Memory: Also known as versatile memory, volatile memory is the memory the contents of which are lost in case of power cut. So when computer is restarted all the memory contents are reset. Different from non-volatile memory, volatile memory has not any limit in writing operations. Platform Configuration Registers (PCR), Attestation Identity Keys (AIK) and Storage Keys are located in this memory.

II.2.4 Key Slot and Key Cache Manager

Key Cache Manager (KCM) is located outside the TPM and responsible for managing key slots of TPM. Key slots are used to hold the relevant keys temporarily which will be used in the current operations of TPM. In order to be able to use such a key, it first must be loaded into key slots. Unused keys are not kept inside the TPM. Rather, they are encrypted by a storage key and kept on hard disk.

II.3 TPM Keys

TPM has various type of keys with different usage goals. During the generation, each key is stored with several attributes pointing out the type of key and its intended usage area. These attributes are assigned during the creation operation and cannot be modified later.

II.3.1 Migratable Non-Migratable Keys

We can basically classify TPM keys into two categories; migratable and non-migratable keys [27].

Migratable keys (MK): Migratable keys are the keys that can be moved to another platform with a different TPM. It only depends on the party generated them. So no one can guarantee that migratable keys belongs to a specific TPM chip.

Non-migratable keys (NMK): Non-migratable keys are the keys that cannot be moved to another platform with a different TPM. They are kept in TPM-shielded location. A certificate, indicating that a key is non-migratable, can be created by TPM.

Migratable versus Non-Migratable Keys: Migratable keys are used in the cases where same key is required to be used on other platforms. For example someone who needs to change his PC or make any upgrade on his platform, should use a migratable key in order not to lose the associated data or certificates. Or someone who wants to use the key installed in one computer system i.e. work computer, on other computer system i.e. home computer, then again he has to prefer migratable keys. Migratable keys can be generated either inside the TPM or outside the TPM. On the other side non-migratable keys are used for different purposes i.e. identifying the machine uniquely. For example if someone wants the encrypted files can only decrypted on a specific computer, then the encryption operation can be performed with a non-migratable key. So that the associated data and certificates are bounded to the platform where non-migratable key is generated. Non-migratable keys reside inside the TPM.

II.3.2 Non-Volatile Keys

Two main non-volatile keys are endorsement key and storage root key.

Endorsement Key: Endorsement key is a key that is embedded into TPM during the manufacturing process and uniquely identifies TPM. Endorsement key cannot be moved out of TPM and cannot be deleted. The manufacturer publishes an endorsement certificate to indicate that the endorsement key has been properly created and the embedded into a valid TPM [27].

Storage Root Key: Storage Root Key (SRK) is a nonvolatile key inside TPM, which is used to wrap keys to be stored on hard disk. As was previously mentioned, it is not possible to store all the keys into TPM due to the limited storage capacity of TPM. For this reason a key hierarchy whose security is bootstrapped from SRK has been created. SRK is created by the platform owner who executes "logical take ownership" command on TPM. So SRK can be changed by platform owner. However; key hierarchy and all its keys are destroyed when the SRK is updated. Therefore; if any encryption has been performed using the keys in the key hierarchy, the encrypted data will not be recovered as well.

II.3.3 Functional Keys

Other than the classification made according to the migratable or non-migratable features of keys, keys can also be categorized according to their functionality [28].

Storage Keys: Storage keys are 2048 bit RSA private keys which are used to store other keys such as another storage, binding or signature key. Storage keys are not used to store symmetric keys and can be either migratable or non-migratable.

Binding Keys: Binding keys are the keys used to store one or multiple symmetric keys. Basic RSA encryption is used.

Identity Keys: Identity keys are the keys used to sign PCRs when an attestation request is made to TPM and to sign other keys as being non-migratable. Identity keys are produced inside the TPM and then provided with a certificate. Since identity keys are created with the SRK as parent, it is guaranteed that they do exist only for that TPM.

Signing Keys: Signing keys are the standard RSA signature keys used in signing operations. The maximum length of the key TPM is able to handle is 2048 bits.

II.3.4 Attestation Identity Keys

Attestation identity keys are non-migratable signature keys used in TPM Attestation operation to attest platform configuration states. The public part of the AIK key is certified by a Privacy CA (Certification Authority) which ensures that the signature key is really generated in the protections of a genuine TPM and the signed state is really the one sent by TPM. The security of AIK key is bootstrapped from the TPM's EK (Endorsement key) which is unique for each TPM. In the attestation process, Privacy CA is a trusted third party certifying that the AIK is generated by a legitimate TPM. There is another attestation type called direct anonymous attestation (DAA) [29] which enables trusted computers to attest directly and anonymously without using a third party. However due the complexity of DAA, most work prefers using a Privacy CA.

II.4 TPM Key Hierarchy

When a new TPM is purchased, it comes with an embedded endorsement key (EK) which has been burned into the chip by the TPM vendor as we explained before. TPM vendor or platform vendor also provides an endorsement certificate with the shipped TPM that guarantees that the endorsement key was generated in a genuine TPM.

After the user activates the TPM, he should take ownership in order to start using it. During taking ownership process, a storage root key (SRK) is created. This key is located in non-volatile memory of TPM.

Since the limited storage capacity of TPM, only some specific keys such as EK and SRK have been kept in non-volatile memory. Other keys are stored on hard disk after an encryption operation by a parent key. TPM maintains a key hierarchy tree as it is seen in figure 3. SRK is the rook key and the security of all the other keys are bootstrapped from SRK. For this reason, SRK is called as root of trust. In key hierarchy mechanism, there is a key slot which is used to temporarily hold the keys whenever they are going to be used by TPM.

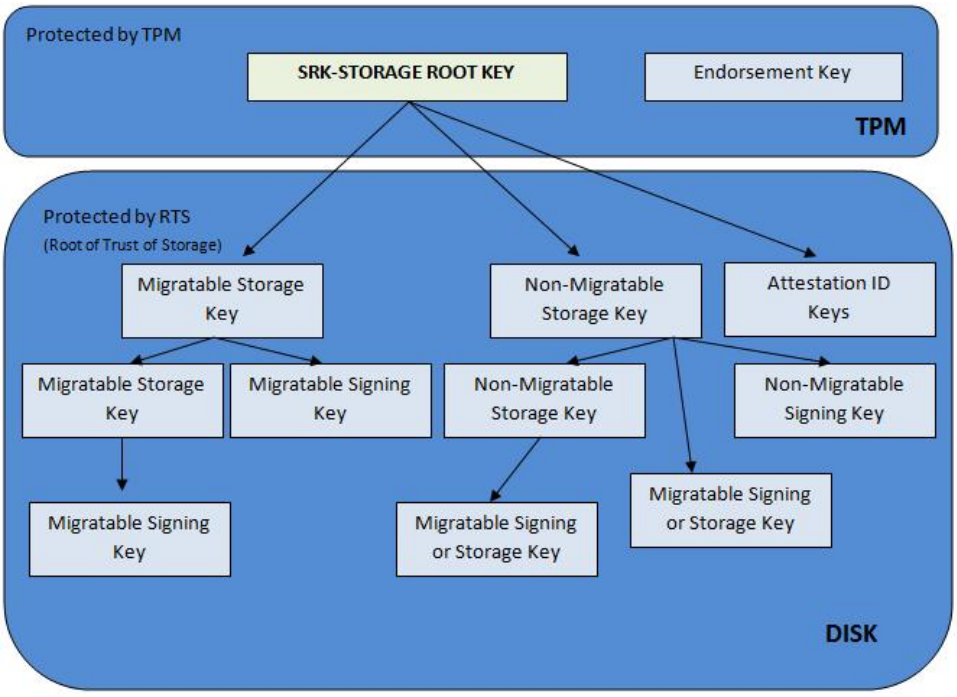


Figure 3: Key Hierarchy

In the key hierarchy mechanism, all key are encrypted by their parent keys. Eventually, every external key is secured by TPM's Storage Root Key. In order to be used, keys should first be loaded to key slot with its parent keys. The decryption

operation is entirely performed inside the TPM. During the creation and usage of keys, some sort of authorization credentials are needed.

When we analyze the key hierarchy figure a little bit closer, we that after SRK, there are different branches in the tree, migratable, non-migratable and Attestation ID keys. The important point here is that although a migratable signing or storage key can be secured by a non-migratable key, the opposite, where non-migratable key is secured by migratable key is not possible. Furthermore, the first migratable key on the left hand side is usually called as Platform Migratable Key which is usually the first key that is loaded into the chip after the machine is booted. This key, usually owned by the system administrator, has the well-known secret for its authorization, but requires the system owner's authorization to migrate. If it is migrated, all other migratable keys in the chain can also be migrated [28].

II.5 TPM Credentials

In order to satisfy the requirements of TCG Specifications, some credentials are defined for a trusted platform; endorsement credential, conformance credential and platform credential [27].

Endorsement Credential: Endorsement credential is the certificate used as evidence that the endorsement key has been created by using a proper TPM and embedded into this TPM during the manufacturing process. The credential basically includes the following information; the manufacturer of the TPM chip, the part model number, version number and stepping and the public part of the EK. The EK can be used along with the Endorsement, Platform and Conformance credentials in platform's identity verification in a protocol to establish AIKs.

Conformance Credentials: The Conformance Credential is a certificate used to provide credibility to properly evaluate the TPMs or platforms containing a TPM. It indicates that the Trusted Building Block (TBB) design and implementation are proper according to the evaluation guidelines. An evaluation service which may be platform manufacturer, vendor or an independent lab can issue this credential. Multiple conformance credentials for multiple TBBs can be issued for a single platform. Conformance credentials include the following information; name of evaluator, platform manufacturer, the model number and version of the platform, the name of the TPM manufacturer, TPM model number and version or stepping and a pointer to the location of the TPM and platform documentation. The conformance credential does not include any privacy information to be used to uniquely identify a specific platform.

Platform Credential: The Platform Credential is a certificate indicating that the platform includes a TPM as described by the endorsement credential. It can be issued by platform manufacturer, vendor and or an independent body. Platform credential includes the following information; the name of the platform manufacturer, the platform model number, version, references to the endorsement credential and the conformance credentials. Platform Credential contains information that can be used to uniquely identify a specific platform. For this reason it is privacy sensitive.

II.6 TPM Functionalities

II.6.1 Integrity Measurement and Extent Operation

Establishing integrity means to ensure that something has not been changed since a period of time. In order to make this, an integrity measurement is carried out at the beginning of the time and a second integrity measurement is carried out at the end of

the time period. If those two measurements match each other then we can say that the integrity is provided.

Integrity measurement is generally performed by calculating the one way hash of the entity. In one way hash functions it is very easy to calculate H(x) from any x value but it is mathematically infeasible to calculate x from a given h value in the H(x)=h equation. The most important feature of hash functions is that they take any message with different lengths as an input and produces output at a constant length. Therefore one way hash functions are also used in increasing the efficiency of cryptography algorithm in the public key cryptography by reducing the high sized files to a constant value.

If we would like to measure the integrity of a document for example, we can basically calculate the hash of the document and store in a secure place. So whenever we would like to check whether the integrity is still provided, a second hash is calculated on the document and after making a comparison with the previous hash value, if they are the same we become sure that the document has not been modified.

One of the important challenge in the given integrity measurement scheme is to store the hash values in a secure fashion. In TPM, a hardware based solution is adapted to store and protect the integrity measurements. TPM has special registers called PCR (Platform Configuration Registers) which are used to store 160 bit SHA1 hash values. There are at least 16 PCRs in a TPM. PCRs cannot be directly written. Instead, they are extended. TPM Extend is a special operation which calculates the new value of the PCR by hashing the concatenation of the old value and a new SHA1 hash value.

The extend operation works like this:

PCR := SHA-1(PCR + measurement)

The extend operation has the following benefits [30]:

- It is unfeasible to find two different measurement values such that when extended returns the same value
- It preservers order in which entities' measurement were extended (for example, extending A then B results in a different value than extending B then A)
- the operation allows unlimited number of measurement to be stored in a PCR, because result is always a 160-bits value

II.6.2 Chain of Trust

One of the important functions of TPM is forming a chain of trust for which a set of entities are hashed and chained to each other by using TPM Extend operations. In practice, this functionality is often used to create a trust chain for software programs by verifying the integrity of all entities that have a potential to affect the trustworthiness of the software. This trust chain is also known as trusted computing base (TCB) of the software.

First entity in the chain of trust is called as Root of Trust for Measurement (RTM). This entity should be indisputable trusted as it has to measure the other entities without faults or errors.

II.6.3 Remote Attestation

Attestation stands for proving the trustworthy status of a machine to a third party, which means that the machine has an original and enabled TPM and the requested hash values are correctly retrieved from the PCRs of the TPM chip. Basically, an attestation request includes a nonce and some PCR numbers. The attested machine

then performs a TPM quote operation and produces a quote as a reply. This quote includes the signed values of nonce and the contents of the requested PCRs. The attested machine also sends an untrusted event log including the hash values of each entity that forms the trust chain in the relevant PCRs. The attester can then verify the untrusted event log by computing the aggregate hashes expected to be in the PCRs and compare the final value with the one in the quote signed by TPM.

Sign operation can be performed via the private portion of either endorsement key (EK) or attestation identity key (AIK). The security of AIK key is bootstrapped from the TPM's EK (Endorsement key) which is unique for each TPM. Using AIK instead of EK has the following benefits [30]:

- EK can be used only by TPM. However; AIK can be used by CPU. Therefore; using AIK instead of EK will reduce the overload on TPM.
- Prevents cryptanalysis attacks against EK.
- Adds an anonymity layer and strengthen the privacy as the AIK is not directly associated with the hardware.

In the attestation process, Privacy CA, which is a trusted third party certifying that the AIK is generated by a legitimate TPM, is used. There is another attestation type called direct anonymous attestation (DAA) [29] which enables trusted computers to attest directly and anonymously without using a third party. However due the complexity of DAA, most work prefers using a Privacy CA.

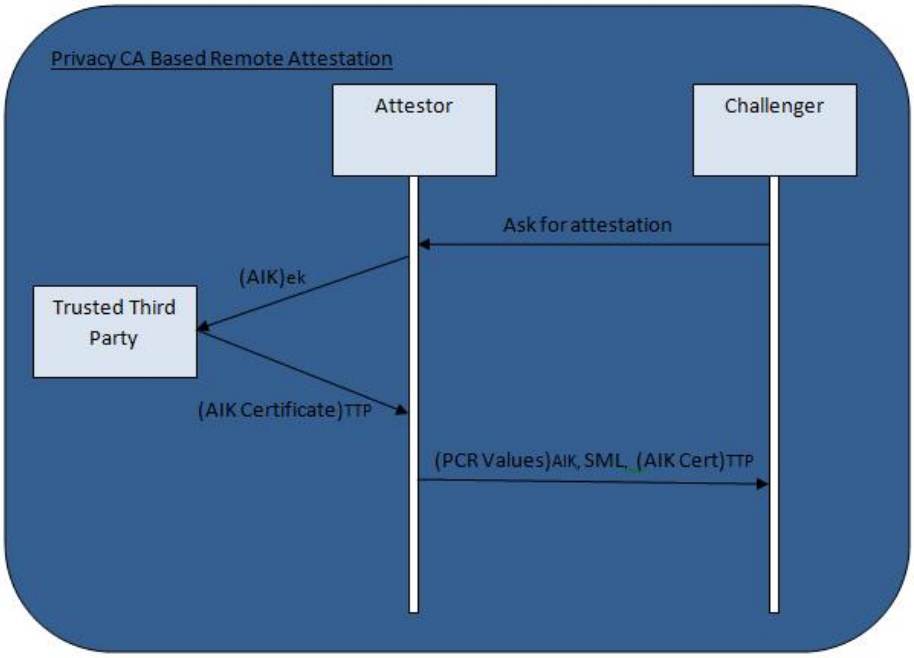


Figure 4: Privacy CA Based Remote Attestation

Privacy CA based remote attestation is depicted in the above figure. Receiving an attestation request, attestor creates an AIK key and sends the public part of it by signing it with the EK public key to a trusted third party which is privacy CA in our scenario. Privacy CA first verifies the EK and if it is correct, it creates an AIK certificate and sends it to attestor. The attestor then signs the requested PCR values with the AIK public key and sends it to the challenger with a stored measured log (SML) and AIK certificate. SML includes the hash values of each entity being extended into the PCR. Receiving all these data, challenger first verifies the AIK public key according to the generated certificate by Privacy CA. If it is ok, then challenger calculates the ultimate hash value by using the SML. If the ultimate hash value matches with the PRC value, the platform is verified as trusted.

II.6.4 Binding

Binding means to bound a message to a specific TPM and thus also the platform including that TPM. As it is well known that in public key cryptography, when a message is encrypted with the public key of the receiver, only the receiver can decrypt it using its private key. So with this point of view, if a message is encrypted with one of the TPM generated public keys the associated private key of which is a non-migratable TPM key, then we ensure that the message can only be decrypted with the same platform including that specific TPM. Therefore; the message is bound to the TPM that protects the corresponding private key.

II.6.5 Signing

In public key cryptography, signing operation is defined as encrypting the message content with the private key of the sender and sending this encrypted text as a signature with the plain text message. So that the receiver can check whether the sent content has not been changed and can verify the sender. In the verification operation digital certificates obtained from a certificate authority in a public key infrastructure is used. With the same logic, TPM can mark some keys as signing keys which are used only signing operations and cannot be used for other purposes for security reasons. By this way the origin and the content of any message signed with these keys can be verified easily.

II.6.6 Sealing

TPM Seal operation, another important function provided by TPM, bounds the encrypted message with a non-migratable private key of TPM and contents of selected PCR values. By this way, it is guaranteed that the encrypted message can only be decrypted by the TPM which performed the encryption and when the contents of the relevant PCRs are as same as the contents available during the encryption operation.

II.7 TPM Root of Trust for Measurement

There are two types of Root of Trust for Measurement, SRTM-Static Root of Trust for Measurement and DRTM-Dynamic Root of Trust for Measurement, which are explained in the following sections:

II.7.1 SRTM: Static Root of Trust for Measurement

By using chain of trust that was mentioned previously, we can also perform a trusted boot which sets up a chain by adding all the entities that have been executed since the boot. As root of trust for measurement, a special trustworthy code called CRTM, Core Root of Trust for Measurement, which is a special BIOS boot block code, is inserted into the BIOS. When the power button is pressed on the computer, the first code to be executed in the BIOS becomes CRTM which will then measure the integrity of the remaining of the BIOS code and extends it into TPM. Then CRTM passes control to the BIOS. The BIOS then measures hardware and the bootloader and passes control to the bootloader. The bootloader measures the OS kernel and passes control to the OS. After the OS has been started, one can understand whether all the integrity values are true which means the computer is a good (expected) state by checking the final hash value in the relevant PCR values. If any of those entities were modified, the value of the PCR would be different than the expected. So the user can decide whether to trust his platform or not.

This chain of trust, starting with BIOS and includes Option ROMs, bootloader, OS and applications, is called SRTM: Static Root of Trust for Measurement.

Trustworthiness of Chain in SRTM: To provide the trustworthiness of the hash chain in SRTM, the following conditions need to be satisfied [31]:

1. Core Root of Trust for Measurement should be trustworthy and cannot be modified.
2. The PCRs are not resettable, without passing control to the trusted code.
3. The chain is contiguous. There is no code in between that is executed but not hashed

Weaknesses of SRTM

A) Bootloader Flaws

Three publicly available TPM enabled bootloaders have been examined in OSLO work [31].

The first one is trusted bootloader built as a part of the Bear project from Dartmount College [32, 33]. In this project they have used a modified version of LILO [34]. They have extended LILO in two ways: the Master Boot Record hashes the rest of LILO and the loaded Linux kernel image is also hashed. The problem detected here was that only the last part of the image containing the kernel itself has been hashed missing the other parts.

The second bootloader that has been laid on the table was the patched GRUB v0.97 from IBM Japan [35] which had been used in IBM's IMA: Integrity Measurement Architecture [36]. The problem here, which had also told as same problem in TCG enabled GRUB [37], is that GRUB loads and extracts a kernel image at the same time instead of loading them completely into memory and extracting them afterwards. Because the program code is loaded twice from disk or from a remote host over the network, an attacker who has a physical access either to the disk or to the network can send different data at the same time.

TrustedGRUB [38] which is another bootloader based on grub solved the issue above by taking the hash in a lower layer when a read operation will be executed. However the version 1.0-rc5 of TrustedGRUB was told to have 3 other bugs. One of these bugs is that its own hash is not calculated when being started from harddisk. The other is that the corresponding PCR is never extended and remains always zero.

B) TPM Reset Attacks

Two TPM reset attacks have been discovered in OSLO. One of them is valid to a specific chip and the other is a general attack. At the first one, it was found out that setting the reset bit in a control register of a v1.1 TPM was able to reset the chip without resetting the whole platform. By this way any PCR value can be reproduced without the opportunity for a remote entity to see the difference via remote attestation.

The second attack was a hardware based attack which is done by physically connecting the LRESET# pin to ground. By this way they were able to perform a reset of the chip itself.

C) BIOS Attack

The trust chain in SRTM starts from the CRTM (Core Root of Trust for Measurement) which is a piece of code in BIOS that extends PCR 0 initially. Normally a CRTM has only to be exchanged with vendor signed code. However it was seen that the CRTM of many machines is freely patchable. It is stored in flash and no signature checking is performed on updates.

II.7.2 DRTM: Dynamic Root of Trust for Measurement

With TPM version 1.2, a new concept called DRTM has been introduced. DRTM avoids the disadvantages of SRTM and removes BIOS, bootloader, OS and other entities from the trust chain (a fresh boot is no more needed). With DRTM, CPU can reset the relevant PCRs at any time by using a specific instruction (SKINIT for AMD, SENTER for Intel) that atomically initializes the CPU, disables the interrupts and loads a piece of code into its cache. This code is sent to the TPM to be an input for TPM Extend operation and written on specific PCRs and then executed.

DRTM makes it possible to run a piece of code in an isolated environment which is not affected by any other entity on the computer system and stores the integrity measurements of the entities used during the DRTM operation on specific PCRs in order to provide the proof whether the relevant piece of code and all its components have been executed correctly.

II.8 TPM DRTM Technologies

In order to make use of DRTM, AMD offers SVM-Secure Virtual Machine technology [39] and Intel offers TXT-Trusted Execution Technology formerly LaGrande Technology (LT) [40].

II.8.1 AMD Secure Virtual Machine Technology

AMD Secure Virtual Machine (SVM) technology, also named as Pacifica, provides virtualization support where all the resources can be shared on a single machine by

multiple operating systems in a secure and efficient fashion with resource guaranteed isolation [39]. SVM technology also supports TPM. Although it is not a must for virtualization technology, by adding TPM support, trusted systems can be built.

With a specific CPU command, SKINIT, a verifiable startup is possible using TPM. So that it is possible to invoke a virtual machine in a secure manner. SVM also supports automatic memory clearing, which protects secrets stored in system memory upon reset.

AMD SKINIT

AMD SKINIT instruction is used to start a root of trust from an initially untrusted operating mode. When SKINIT is executed, it reinitializes the processor to set up a secure execution environment for secure loader (SL), which is a special software code to load and run security critical code. After that SL is executed in a secure manner. SKINIT also copies the secure loader executable image to a Trusted Platform Module (TPM) for verification using unique bus transactions that prevent SKINIT operation from being emulated by software in a way that the TPM could not readily detect.

One of the important features of SKINIT is that it allows to initiate SVM protections in a reliable manner while the system is already up and running without need a boot process.

Secure Loader

The function of secure loader is to initialize SVM hardware mechanisms and related data structures to initiate the execution of a trusted piece of software such as a VMM or hypervisor. Before passing control to this piece of software, secure loader

calculates the hash of this software and extends it to the relevant PCRs. By this way; the integrity of the executed code can be verified later.

Secure Loader Image

Secure loader image incorporates the secure loader code and its initialized data sections which are used to initialize and start a security kernel (i.e. VMM, Hypervisor) in a completely safe manner including setting up DEV protection for memory allocated for use by SL and SK. The SL image is loaded into a region of memory called the secure loader block (SLB). The size of SL image can be maximum 64Kb. The SL image is defined to start at byte offset 0 in the SLB.

The first 16 bits of the SL image points out the SL entry point. The second word contains the length of the image in bytes. All these values are used by SKINIT instruction. The layout of the rest of the image is determined by software conventions. The image also contains a digital signature for validation purposes.

Secure Loader Block

The secure loader block, depicted in figure 5, is a physical memory block with 64Kb size and located at any 64Kb aligned address below 4 GB. Before SKINIT execution, SL image should be loaded into the SLB starting at offset 0. The physical address of the SLB is provided as an input operand (in the EAX register) to SKINIT, which sets up special protection for the SLB against device accesses.

The SL must be defined as to execute in flat 32-bit protected mode with paging disabled. By using EAX, a base address is derived to access data areas within the SL image using base + displacement addressing, to make the SL code position-independent. Memory between SL image and the end of the SLB is used as SL

runtime data area. SKINIT sets the ESP register to the appropriate top-of-stack value. The following figure illustrates the layout of the SLB, showing where EAX and ESP point after SKINIT execution.

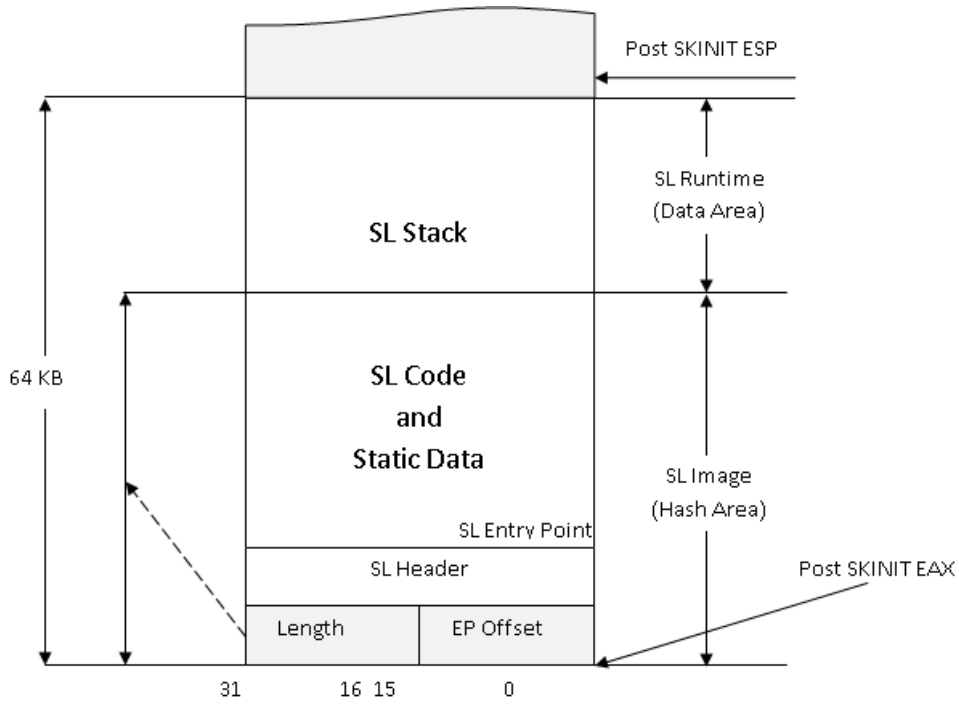


Figure 5: Secure Loader Block [39]

Trusted Platform Module

During the SKINIT execution, Trusted Platform Module recognizes SKINIT transactions, receives the SL image and verifies its signature. Based on the outcome, the device decides whether or not to cooperate with the SL or subsequent SK. SKINIT uses special support logic in the processor's system interface unit, the internal controller and the I/O hub to which the TPM is attached. SKINIT uses special transactions that are unique to SKINIT and securely transmits the SL Image to the TPM for validation.

SKINIT Operation

SKINIT, taking the physical base address of the SLB as its only input operand in EAX, performs the following steps [39]:

1. *Reinitialize processor state in the same manner as for the INIT signal, then enter flat 32-bit protected mode with paging off. The CS and SS selectors are set to 0008h and 0010h respectively, and CS and SS base, limit and attribute registers are set to (base = 0, limit = 4G, CS:read-only, SS:read/write, expand-up). DS, ES, FS and GS are left as 16-bit real mode segments and the SL must reload these with protected mode selectors having appropriate GDT entries before using them. (Initialized data in the SLB may be referenced using the SS segment override prefix until DS is reloaded.) The general purpose registers are cleared except for EAX, which points to the start of the secure loader, EDX, which contains model, family and stepping information, and ESP, which contains the initial stack pointer for the secure loader. Cache contents remain intact, as do the x87 and SSE control registers. Most MSRs also retain their values, except those which might compromise SVM protections. The EFER MSR, however, is cleared. The DPD, R_INIT and DIS_A20M flags in the VM_CR register are unconditionally set to one.*
2. *Form the SLB base address by clearing bits 15–0 of EAX (EAX is updated), and enable the SL_DEV protection mechanism to protect the 64-Kbyte region of physical memory starting at the SLB base address from any device access.*

3. *In multiprocessor operation, perform an inter-processor handshake.*
4. *Read the SL image from memory and transmit it to the TPM in a manner that cannot be emulated by software.*
5. *Signal the TPM to complete the hash and verify the signature. If any failures have occurred along the way, the TPM will conclude that no valid SL was started.*
6. *Clear the Global Interrupt Flag. This disables all interrupts, including NMI, SMI and INIT and ensures that the subsequent code can execute atomically. If the processor enters the shutdown state (due to a triple fault for instance) while GIF is clear, it can only be restarted by means of a RESET.*
7. *Update the ESP register to point to the first byte beyond the end of the SLB (SLB base + 65536), so that the first item pushed onto the stack by the SL will be at the top of the SLB.*
8. *Add the unsigned 16-bit entry point offset value from the SLB to the SLB base address to form the SL entry point address, and jump* to it.

II.8.2 Intel Trusted Execution Technology

Intel Trusted Execution Technology defines platforms level enhancements to create building blocks for trusted platforms [40]. It establishes the authenticity of the controlling environment in a way that can be verified by the entity who will take trust decision on the platform.

Intel TXT defines some set of extensions to provide a measured and controlled launch of a system software which will then create a protected environment for itself and any additional software to be executed in this environment. The extensions enhances two

area; launching of a Measured Launched Environment (MLE) and the protection of the MLE from potential corruption.

The enhanced platform provides these launch and control interfaces using Safer Mode Extensions (SMX) which include the following functions:

- Measured launch of the MLE
- Mechanisms to ensure the above measurement is protected and stored in a secure location
- Protection mechanisms that allow the MLE to control attempts to modify itself.

Measured Launched Environment (MLE)

With the measurement term, Intel TXT means the integrity measurement which can be performed through cryptographic hash functions. The software launched using the SMX instructions is known as the Measured Launched Environment (MLE). MLEs provide different launch mechanisms and increased protection.

Launch Sequence

Intel TXT establishes the authenticity of a measured launched environment (MLE) and protects this environment from potential corruption. MLE then establishes an isolated environment for itself and additional software it may execute. In order to be able to launch MLE, first of all, an Authenticated Code (AC) Module, which is specific for the chipset and digitally signed by the chipset vendor, should be loaded into the memory. Only when AC module and MLE are in memory, the launching environment can invoke an instruction (i.e., GETSEC[SENTER]) which initiates the TPM DRTM functionality on the processor. This specific command brings the

chipset and CPU in a stable state and loads, validates and executes the AC. AC module then ensures that the platform has a proper configuration and measures and launches the MLE.

Storing the Measurement

During the launch operation, the integrity of the MLE is accurately measured. Then this measurement is extended into the relevant PCRs in TPM. Intel TXT supported platform ensures that this measurement of MLE is properly reported to the TPM. Then MLE can use these measurements in TPM to protect sensitive information and detect unauthorized changes to the MLE itself.

Controlled Take Down

Intel TXT implements a controlled take down while exiting the MLE. During the take down, any guest VMs (if there are) are shut down and the previously used memory is ensured not to leak any sensitive information. The MLE cleans up after itself and terminates the MLE control of the environment. If a VMM was running, the MLE may choose to turn control of the platform over to the software that was running in one of the VMs.

Authenticated Code Module

Authenticated Code Module, a special code module loaded into internal RAM of the CPU, supports the establishment of a measured environment. Before being executed, AC module is first authenticated. This is done through a digital signature located in the header of the AC module. The processor calculates a hash of the AC module and uses the result to validate the signature. SMX technology executes the AC module only if it can successfully authenticate the AC module. As the authenticated code

resides within the internal RAM of the CPU, the module can execute in isolation with respect to the contents of external memory or activities on the external processor bus.

Chipset Support

In Intel TXT, DMA protection via VT-d emerges as one of the important features of the chipset. VT-d, under control of the MLE, allows the MLE to protect itself and any other software such as guest VMs from unauthorized device access to memory. VT-d blocks access to specific physical memory pages and the enforcement of the block occurs for all DMA access to the protected pages.

The extensions defined with Intel TXT, can access certain chipset registers and TPM address space. Using read/write protocols, system software can access to chipset registers which interact with SMX from two regions of memory, Intel TXT public space and Intel TXT private space. System software cannot access to Intel TXT Private Space until it is unlocked by SMX instructions.

The storage spaces accessible within a TPM device are grouped by a locality attribute and are a separate set of address ranges from the Intel TXT Public and Private spaces. The defined localities are as follows:

- Locality 0 : Non-trusted and legacy TPM operation
- Locality 1 : An environment for use by the Trusted Operating System
- Locality 2 : Trusted OS
- Locality 3 : Authenticated Code Module
- Locality 4 : Intel TXT hardware use only

II.9 Some Important TPM Projects

We, in this section, introduce three important projects using TPM DRTM functionality, OSLO, Flicker and Tboot.

II.9.1 OSLO

OSLO [31] is one of the first projects implementing TPM DRTM functionality. It is started as kernel from a multi-boot compliant [41] loader. During the startup, OSLO first initiates the TPM in order to be able to perform Extend operation. Since the SKINIT instruction can only be run on single CPU, OSLO stops the other CPUs (if there are). By this way any malicious intervention from other CPUs are also prevented.

After the required operations are done for platform initialization, OSLO executes SKINIT to switch to the secure mode. OSLO, then, hashes every module that is preloaded from the parent boot-loader before starting the first module as a new kernel. OSLO uses chainloading via the multiboot specification to be flexible with respect to the operating system it loads. OSLO can be loaded normally from a multiboot-compliant loader started by the BIOS such as SysLinux [42] or GRUB. However; OSLO can also be loaded from Linux kexec environment [43].

II.9.2 FLICKER

Flicker [44], is a platform that utilizes TPM DRTM functionality to execute security-sensitive code block of a software in hardware-supported isolation from all the other software and devices on the platform.

Flicker uses AMD SVM and Intel TXT capabilities to achieve its properties. Instead of launching a security kernel, Flicker pauses current execution environment, which might be untrusted, executes security sensitive code using SKINIT and resume the

operation of the execution environment. The security sensitive code executed in TPM DRTM protection is called as PAL – Piece of Application Logic in Flicker jargon.

Execution of Flicker is illustrated in figure 6. Flicker is written as a SYSFS module which is a virtual file system capable of exporting information about devices and drivers from kernel to user space so that it becomes possible to make data exchange between a user level application and the Flicker module. In the SYSFS, Flicker module have four files; control, inputs, outputs and slb. User level applications interact with flicker-module vie these files. Applications first writes an uninitialized SLB including its PAL code into the slb entry. If there is any input to be given to PAL, it is written in inputs SYSFS file. The inputs are made available at a well-known address once execution of PAL begins. The application initiates the Flicker session by writing to the control entry in the SYSFS.

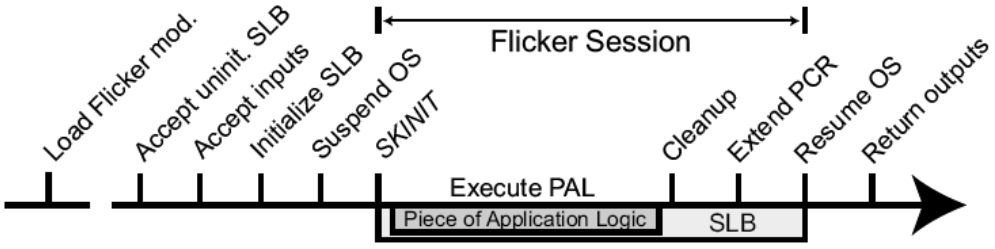


Figure 6: Timeline showing the steps necessary to execute a PAL [44]

Flicker module then initializes the SLB. This includes some kind of actions such as enabling the processor's segmentation support and creating segments that start at the base of the PAL code, detecting the starting address of PAL during the memory allocation for SLB and inserting appropriate entries in the SLB Core.

SKINIT does not save existing state when it executes. However, untrusted OS should be resumed after Flicker session. In order to do this only the bootstrap processor should be running in a multi processors system. So by using CPU Hotplug support available in recent Linux kernels, Flicker deschedules all application processors and when they are idle, Flicker module sends an INIT IPI by writing to the system's Advanced Programmable Interrupt Controller. Then the system becomes ready to execute SKINIT. Before this, Flicker saves information about the Linux kernel's page tables so the SLB Core can restore paging and resume the OS after the PAL exits.

After invoking SKINIT command, hardware protections are enables and the SLB Core is started to execute. Hardware protections includes disabling DMA to the memory region containing SLB, disabling interrupts and debugging support. Once the environment has been prepared, the PAL executes its application-specific logic. During PAL execution, output parameters are written to a well-known location beyond the end of the SLB. When the PAL exits, the SLB Core regains control. The PAL's exit triggers the cleanup and exit code at the end of the SLB Core. The cleanup code erases any sensitive data left in memory by the PAL.

After the required operations are carried out to restore the kernel's page tables using the values saved during the Suspend OS phase, the control is transferred back to the flicker-module. The flicker-module restores the execution state saved during the Suspend OS phase and fully restores control to the Linux kernel by re-enabling interrupts. If the PAL outputs any values, the flicker-module makes them available through the SYSFS outputs entry.

CHAPTER III

INCREASING TRUSTWORTHINESS OF AUTHENTICATION PROXIES

Especially in corporate settings, proxy systems are in use for variety of purposes such as caching, access control, content filtering, logging, etc. An application area for proxies that is not as popular as others but has received significant attention (e.g., [4-9]) in the academic literature is to use them as agents for user authentication. With an authentication proxy, user first establishes a secure session with the proxy. Then, in each time user wants to login to a server, the proxy intercepts the connection, inserts the user credentials into the page and then submits it to the target server.

Two prominent advantages that authentication proxies can provide, improving the usability of credential management and increasing the security of user authentication, are described briefly as follows:

(i) Remembering and using large and continuously growing number of credentials (e.g., passwords, PINs and even usernames) becomes a real burden for users. Due to usability problems, users may prefer insecure options such as reusing the same password for different web sites. In this sense, authentication proxies enable users to store their credentials on the proxy and use them by entering just a single password shared with the proxy.

(ii) Authentication proxies can also improve security by making it possible to use more secure alternatives such as one-time passwords even when the server itself does not support it [8].

In the literature, there is a considerable amount of work on authentication proxy systems (e.g., [4-9]). Although these proxy systems offer benefits with respect to security and usability, two of their problems are noteworthy; Firstly, authenticating users to the proxy system in a secure and usable fashion is still a serious problem. One may argue that the right balance for using more secure but less usable solutions like one-time passwords could be achieved by limiting their use only once per session opened with the proxy and only when an untrusted machine other than the user's primary computer is to be used. A one-time password based solution proposed in the earlier work on authentication proxy systems is also adopted in our proposed framework but we note that our contribution is not on this first problem.

The second problem, which is the central focus in our work, is less spoken but at least as serious as the first one; increasing the trustworthiness of the proxy system so that users would accept to hand over their sensitive credentials such as e-banking passwords to proxies without worrying about possible security breaches, intended or by mistake. This may be the main reason why proxy systems have not found a wide adoption among users for authentication purposes[1]. In our literature survey, we see that previous work on proxy systems have made trust assumptions and this problem has not been studied in detail before.

In this thesis work, we make a first attempt towards establishing the trustworthiness of authentication proxies. For this purpose, we propose Trust-in-the-Middle, a proxy

[1] A recent usability study confirms that users are not comfortable with giving control of their passwords to an online entity [45].

system based on trusted computing technology and its core component TPM (Trusted Platform Module). With the TPM Dynamic Root of Trust for Measurement (DRTM) functionality, we securely store and submit sensitive credentials to the target servers without disclosing them even if the proxy server is compromised. All the security critical operations are carried out in the modules whose integrity is protected by TPM. The credentials are never put out of DRTM protection without the cryptographic protection. Therefore, any malicious entity cannot intervene the operation and access the credentials in plaintext. We use remote attestation to verify the security of the software modules on the proxy. Sensitive data is sent to proxy only after the attestation result is checked.

III.1 Related Work

Previous work is discussed under three headings: proxy-based systems, TPM-based systems and password managers and identity management systems.

III.1.1 Proxy Based Systems

A proxy-based system called Impostor for use from untrusted devices was proposed by Pashalidis and Mitchell [4]. Impostor, the proxy, keeps a copy of user credentials for different web sites in this system. Whenever a user wants to connect to a site, Impostor intercepts the connection and sends a special login screen to the user. The login screen involves a challenge/response mechanism which requires users to share passphrases (at least eight characters) with the proxy server. The challenge asks user to provide three randomly chosen characters from the passphrase. If the response is correct, then Impostor sends the user's credential to the site and completes the authentication. By this way, if the user's machine is compromised, only a small portion of the secret i.e., the passphrase is revealed. As the challenge changes each time the user connects to the proxy, a replay attack is rendered to be more difficult. However, since the secret used for responses is same, an adversary obtaining an

adequate number of responses is able to build the entire secret. The security of the proxy system is also not discussed for Impostor and an inherent trust assumption is made.

Wu et al. [5] propose another similar architecture where a proxy stores credentials and asks the user to respond to a challenge before submitting them. The challenge is also sent as an SMS message to the user's mobile phone. The SMS message includes a link which directs user to a WAP page to let him accept or deny the connection. By comparing challenges on two pages, user could avoid phishing attacks. In this system, the proxy is again assumed to be trusted.

Delegate [6] is another proxy based authentication system. As the main difference to the other systems we discuss, Delegate implements rule-based policies and requests additional credentials via the mobile phone of the user whenever a sensitive operation is to be carried out.

KLASSP [7] proposed by Florencio and Herley is a proxy-based system which differs from other similar systems by not storing user passwords in the proxy. Although passwords are not stored on the proxy system, this does not eliminate the proxy trust problem because the proxy now holds other secrets (i.e., mapping table) to recover the password. Besides, a malicious software on the proxy system can access to the plaintext password while it is being submitted to the target server.

Florencio and Herley proposed another system based on the proxy idea [8]. This time, one time passwords are also incorporated into the proposed solution. Before using the system, users register to the proxy called URRSA and provide the credentials (passwords) of the target servers. URRSA generates the one-time passwords from the passwords using an encryption algorithm.

Martineau and Kodeswaran proposed a very similar system to URRSA called SecurePass [9]. However, the system has the same drawback with respect to trustworthiness of the proxy.

III.1.2 TPM Based Systems

Up to our best knowledge, there is no earlier work on applying trusted computing technology on a third party authentication proxy system which is not in control of user. Below, we review previous work on using this technology for a more general problem, the problem of protecting credentials on untrusted environments.

One previous work that addresses the problem of protecting sensitive input on untrusted environments using TPM is Bumpy, proposed by McCune et al. [46]. The system is based mainly on Flicker [44]. Bumpy allows the user to specify strings of input as sensitive while entering them and ensures that these input reach the desired endpoint in a protected state. The sensitive input are processed in an isolated code module (Flicker) on the user's system where they are encrypted or otherwise processed for a remote web server. Bumpy requires special equipment like an encrypting keyboard and may require change on the server side.

Li et al. proposed a secure user interface for web applications running on an untrusted operating system [47]. With a small portion of code included in the user interface a secure path from the user directly to the remote server is built. After the interface attests itself to the remote server, sensitive inputs are handled in this interface and transferred back to the OS with cryptographic protection. The proposed system utilizes TPM DRTM and Intel TXT technology to create an isolated environment as Flicker does [44]. The difference is that a simple VGA and a keyboard driver are also added in the measured launch environment (MLE) (the isolated environment of Intel

TXT technology created by TPM DRTM operation). By this way, users are able to interact with the MLE directly. When the user is required to input sensitive information, the browser places the MLE in the memory and invokes it to handle secure input and output.

Borders and Parkash proposed a virtualization based architecture [48]. On the client machine, the keyboard inputs are directed to a trusted input-proxy (TIP) which executes in another virtual machine on the same machine. TIP replaces the real inputs with placeholders and sends them back to the primary OS. When the primary OS sends the packet to the network, the TIP searches for these placeholders and replaces them with the real inputs. T_PIM [49] is another similar solution which uses an activation password instead of placeholders and also uses TPM to verify the integrity of the trusted virtual machine.

Gajek et al. proposed two similar wallet based authentication systems in [50] and in [51] utilizing virtualization and trusted computing technologies also on client machine. The systems consist of a trusted wallet acting as web proxy to perform the user login at web sites and a security kernel that provides a secure environment for the wallet. Truwallet provides protection for users' credentials and sensitive data by binding them to the user's platform configuration based on trusted computing technologies. The latter work, Truwallet [51], extends the previous work [50] by adding secure migration of the wallet data to another machine and implementing an automated login procedure where server is authenticated independently from (SSL) certificates.

TIP, T_PIM and the wallet systems mentioned are virtual machine based systems deployed solely on client machines hence they are not proxy-based systems as the ones discussed in this thesis. Truwallet in [51] also requires change on server side in

order to establish secret between wallet and server during registration. We also note that using virtual machines may bring additional security problems [52].

In [53], Kostiainen et al. describes how general purpose secure hardware can be used to develop an inexpensive, secure and open architecture for credentials which they call On-Board Credentials (ObC). Although their implementation is based on M-Shield and used in mobile phones, their architecture can also be implemented on TPM based secure environments. Bugiel et al. also introduces a framework for application-specific credentials and provides a prototype implementation using mobile trusted module and DRTM technologies in [54]. Both work provide an infrastructure for credentials in user's computer. However; they do not include how this infrastructure can be used in credential input on third party systems securely.

Bugiel et al. proposed TruWalletM another wallet-like password manager and authentication agent based on trusted execution environment in [55]. However, their proposal is for mobile platforms.

Pashalidis and Mitchell proposed a single sign-on system based on trusted computing in [56]. In their system, client computer is used as authentication service provider (ASP). Any service provider can perform attestation in order to check the integrity of the ASP. The system is a complex system requiring domain knowledge in order to install and operate it on client side. Since single sign-on identities are trusted computing identities specific to the client system's TPM, the system is not portable.

Trusted computing has also been used in cloud computing for storing and securing sensitive information. Trusted computing group gives an overview how trusted computing can be useful for cloud computing in [57]. Patidar et al. mention same techniques in their paper [58]. Li et al. proposed C- MAS: The Cloud Mutual Authentication Scheme for cloud authentication using TPM and smart cards [59].

Senthil et al. examine how trusted computing can improve the security of cloud computing in [60]. Naruchitparames et al. proposed a blind processing scheme in [61] where user exchanges sensitive information with a remote cloud system via isolated processes built with the help of TPM and virtualization. Shen et al. proposed a cloud computing system based on trusted computing in [62]. Santos et.al. presented Excalibur in [63] which provides a trusted computing abstraction in the cloud called policy-sealed data that lets data sealed and then unsealed only by nodes whose configuration match the policy. When we examine all those previous work, we see that trusted storage functionality of trusted computing is used for securing sensitive data during storage. For verifying the trustworthiness of either a server on the cloud or a client system, remote attestation is used. A virtual trusted execution environment executed after a TPM based authenticated boot is also one of the popular mechanisms preferred in cloud computing. For authentication, TNC (Trusted Network Connect) functionality of trusted computing is preferred to provide authorization to the cloud resources. However; none of the previous work proposed a solution implementing trusted computing based authentication in the cloud to access a service outside the cloud without requiring change on server side.

III.1.3 Password Managers and Identity Management Systems

Password managers can exist in different formats: web based password managers such as Microsoft's Windows Live ID scheme (formerly Passport) [64] or OpenID [65], browser plug-ins such as Password Maker [66], PwdHash [67] or Password Multiplier [68], stand-alone applications such as Site Password [69] and bookmarklets such as Password Generator [70]. While some of these password managers just let users manage their existing passwords, the others may make them use high entropy passwords generated from a single easy-to-remember password. So that both the security of the passwords are improved and the burden on users to memorize all different passwords are removed.

According to the location of password manager, two classifications can be done: client-side password managers, online password managers. While the former runs on user's client device, the latter provides remote service on the internet. Both of these classifications can have their own advantages and disadvantages; Client-side password managers have portability problems and more prone to failure (i.e. damage of PC or theft). On the other side online password managers suffer from trust problems that we focus in this thesis work.

There are other more complex identity management systems which are not limited with identities such as username and passwords but manage authorization, roles , privileges of the user as well. We refer interested readers to some popular applications, protocols, standards and initiatives such as SAML [71], Yadis [72], OAuth [73], OpenID [65] and Liberty Alliance [74].

Trusted computing has also been utilized in some of those identity management systems such as [75] and [76]. However; our focus in this thesis work is not on complex identity management systems, but on the authentication proxy systems which basically carry out credential input operation on behalf of the user. The basic difference is that we do not want any modification on the available authentication protocols and want to be transparent to the service providers, which means that we do not insist any change on server side.

III.2 Proposed System

Our proposed system Trust-in-the-Middle is explained in the following sections:

III.2.1 Model, Objectives and Assumptions

Before giving the technical details of Trust-in-the-Middle system, we present our system model. We also describe the assumed attacker model informally and explain our objectives, limitations and assumptions:

System Model. Figure 7 depicts the system model of an authentication proxy system. There are three entities: (i) user, (ii) proxy, (iii) target servers. The user, who wants to authenticate himself to several target servers, uses the authentication proxy system which is in a relay agent position intercepting the communication and carrying out the tasks required for authentication on behalf of him.

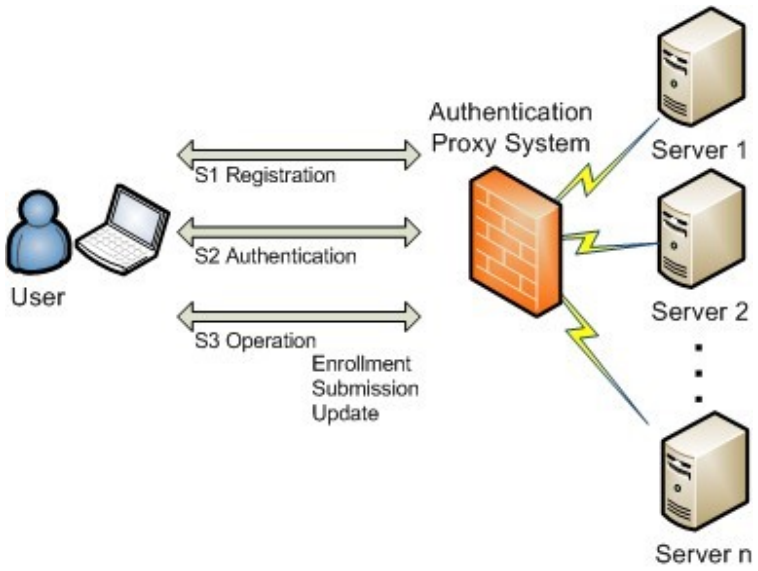


Figure 7: System Model of our Authentication Proxy System

Within this system model, there are three main services provided by the proxy: (S1) Registration, (S2) Authentication and (S3) Operation.

S1 Registration. The user registers to the system and shares with the proxy a master password and optionally a one-time password list

S2 Authentication. User's identity is verified first by the proxy before serving him for further operations.

S3 Operation. The following operations are made available by the authentication proxy system:

- Enrollment: Users could enroll their credentials for the servers on the authentication proxy at any time.
- Submission: Credentials, stored on the proxy, could be submitted to the servers on behalf of their authenticated owners.
- Update: Users could also update their stored credentials on the proxy any time they wish.

Attacker Model. We assume that the main goal of an attacker is to obtain user credentials. We can list four places where credentials are under threat: (i) Client machine, (ii) Network between clients and the proxy as well as between the proxy and target servers, (iii) Servers, (iv) Authentication Proxy.

Client machine. The attacker may try to capture user credentials while they are being entered on the client machine by using methods such as keystroke logging, screen-scraping, malicious code injection on the operating system, browser or any other software on the client.

Network between client-proxy-servers. The attacker may conduct several types of attacks in order to capture the credentials on the network i.e. network sniffing, pharming, man-in-the-middle attacks, etc.

Target Servers. The attacker can perform an attack on server machines to capture the stored credentials.

Authentication Proxy. The attacker may try the following methods to obtain credentials while they are being processed or stored on the proxy; He can attack directly to the database where the credentials are stored and try password guessing or dictionary attacks if the credentials are stored in hashed form. He can try to capture the credentials while they are being enrolled, submitted or updated. He can inject malicious code on the software modules running the enrollment, submission and update protocols.

Security Objectives and Limitations. Our main objective in this thesis work is to focus on authentication proxy systems and address the concerns with respect to security of user credentials while they are being stored and processed on these systems. Our threat model does not include client side or network threats but the threats on proxy system. We take into consideration each services provided by the proxy and provide an overall security architecture.

Despite being not our focus, we note that our proposed framework also provides protection against some of the client side attacks (due to its support for one-time passwords) and network attacks (due to use of encrypted tunneling). See the discussion in Section IV.1 for more details.

A limitation of the proposed work is that we are concerned with the security of the authentication prior to proceeding with the online transaction, not with the security of the transaction as a whole. Hence; sophisticated attacks such as session hijacking attacks, transaction generators, etc. are not addressed. Furthermore, attacks to target servers, denial of service attacks, physical attacks, and social engineering attacks are

also out of scope in our work. We also do not discuss the privacy implications of using proxies.

Usability Objectives and Limitations. Our objectives are listed as follows:

- To enable users to access different password protected web sites using only one (master) password. As a result, users do not need to memorize multitudes of passwords.

- To provide users an easy access to the password protected web sites as long as their authentication session with the proxy system is alive. After authenticated to the proxy, users could feel as if they were not using passwords at all.

- To provide a smooth user experience not only during login but also for password update i.e., by filling the current password fields automatically while user is changing password and by updating stored passwords on the proxy at the background without requiring further user action.

- The proposed system should be transparent; not requiring any change on server side

- The proposed system should not require a specific architecture or operating system on user side.

The usability limitations of the proposed system are as follows:

- There is a need for registration to the proxy before using the system (but the registration does not need to be offline).

- Users should install a special client software on their machines to use the authentication proxy service.

- Users should make relevant proxy settings in their browsers[2].

- Trust-in-the-Middle does not support advanced authentication or identity management services. It only works with the target sites requiring a basic authentication with username and passwords

Additional Assumptions. We also make the following assumptions:

- Proxy machine has a TPM v1.2 chip and supports TPM DRTM.

- Source codes, binary files and hash values of software modules and PAL used in the proxy system are publicly available (i.e. on the official web site of the proxy) and verified as being secure. When an update on either software modules or PAL is of concern, new values are replaced and users are informed.

- The client software is not compromised.

III.2.2 Overview

In this section, we present an overview of how our proposed system implements the system model given in Figure 7. For this purpose, the interaction between a user and Trust-in-the-Middle System is summarized as follows:

[2] We present an add-on in our prototype implementation that makes the Proxy settings automatically for using Trust-in-the-Middle system.

Before using the system, the user registers to the system and obtains a proxy authentication master password and optionally a one-time password list. This is carried out through a client software on user side which runs a secure registration protocol. Registration requires to be performed on only trusted client machines.

After completing the registration and configuring the relevant proxy settings, the user can start using the system. Through the client software, he first authenticates himself to the system either using master password or one-time password. If the client machine is trusted (for instance if it is his primary machine), he can use the master password. Otherwise, one-time passwords may be preferred. During authentication, a remote attestation protocol is executed between client software and the proxy system and only if the connected module is verified to be secure, can user send his password to the proxy. If the authentication is successful, a tunnel is established between the client and the proxy system. This tunnel is used to open a local port on client system and redirect the traffic to the Trust-in-the-Middle proxy service.

The user can now visit any web site he wants to login. Since proxy settings of the user are configured to forward all the web traffic to the Trust-in-the-Middle through the established tunnel, Trust-in-the-Middle intercepts the connection. As most of the security critical login pages require SSL, Trust-in-the-Middle functions as an SSL MITM proxy and sets up two independent SSL connections, one with the user and one with the target server. While establishing the SSL connection with the server, Trust-in-the-Middle checks its SSL certificate and establishes connection only if it can verify the certificate correctly. Trust-in-the-Middle then checks whether the authenticated user has previously stored credentials for the target server or not. If it finds a match, it sends the login page to the user by filling in the credential fields with dummy credentials. Otherwise, it sends the page with empty credential fields. Trust-in-the-Middle also inserts the expression “Trust-in-the-Middle” just above the

credential input part as a visual cue indicating that Trust-in-the-Middle intercepted the connection.

Seeing that the login page is pre-filled, the user understands that Trust-in-the-Middle has filled it for himself. So the only thing he needs to do is to click on the submit button. Another option here is not sending the login page to the user if he has previously registered credentials for the target site and carrying out submission directly. However; we did not prefer to disrupt the original flow as the user may need to see other information on the login screen such as security warning messages etc.

Receiving the submission, Trust-in-the-Middle first retrieves the encrypted credentials from user database and initiates a TPM DRTM operation to securely decrypt and obtain the credentials. After that, Trust-in-the-Middle inserts the real user credentials into the correct fields and submits the page to the target server. If the login page has empty credential fields, the user understands that he has not entered these credentials using Trust-in-the-Middle before. If the user trusts the client environment, he enters the credentials so that Trust-in-the-Middle submits them on behalf of him and also enrolls them on proxy for future use.

Trust-in-the-Middle also supports credential update operation. When the user visits the credential update page of the target web site, the system is able to detect and fill in the current credential fields automatically. For the new credentials, the same process as credential enrollment is followed. Trust-in-the-Middle receives the new credentials, encrypts them with TPM protected keys and then updates its user database.

Trust-in-the Middle provides the described services by executing Main and Auxiliary protocols given in Table 2. Main protocols execute the auxiliary protocols at level 1 which may execute another auxiliary protocol at level 2.

In Table 2, there is a specific main protocol, Initial Sealing Protocol, which does not have a match with a service or an auxiliary protocol. This protocol is executed only once while the Trust-in-the-Middle system boots up. It is used to protect the integrity of the public key of the proxy module with the help of TPM till the next boot of the system. Other main protocols are responsible for carrying out the operations regarding five different services; Registration protocol is used to create and store a master password and a one-time password list to be used in subsequent proxy authentication. Authentication protocol is used to authenticate users with previously registered master or one-time passwords. Credential enrollment protocol is responsible for enrolling user credentials in Trust-in-the-Middle database encrypted with TPM protected keys. Credential submission protocol is used to decrypt and insert user credentials into the login page and perform submit operation. Credential update protocol is used to replace previously stored credentials by the new ones.

Table 2: Protocols used for implementing the services of the Trust-in-the-Middle system

Services	**Main Protocols**	**Auxiliary Protocols**	
		Level-1	**Level-2**
Registration	Registration Protocol	Secure Tunnel Protocol	Attestation Protocol
Authentication	Authentication Protocol	Secure Tunnel Protocol	Attestation Protocol
Operation(Enrollment)	Credential Enrollment Protocol	Credential Decryption Protocol	-
Operation(Submission)	Credential Submission Protocol	Credential Decryption Protocol	-
Operation (Update)	Credential Update Protocol	Credential Decryption Protocol	-
-	Initial Sealing Protocol	-	-

Main protocols use two auxiliary protocols at level 1, Secure Tunnel Protocol and Credential Decryption Protocol. The main job of secure tunnel protocol is to establish a tunnel with the security sensitive code running in TPM protected environment. By this way a direct and secure communication with the sensitive code is provided. The

Credential Decryption Protocol is used to obtain credentials securely which were previously encrypted with TPM protected keys. The only auxiliary protocol running at level 2 is attestation protocol which is executed by secure tunnel protocol in order to verify whether the correct code is executed in TPM protection.

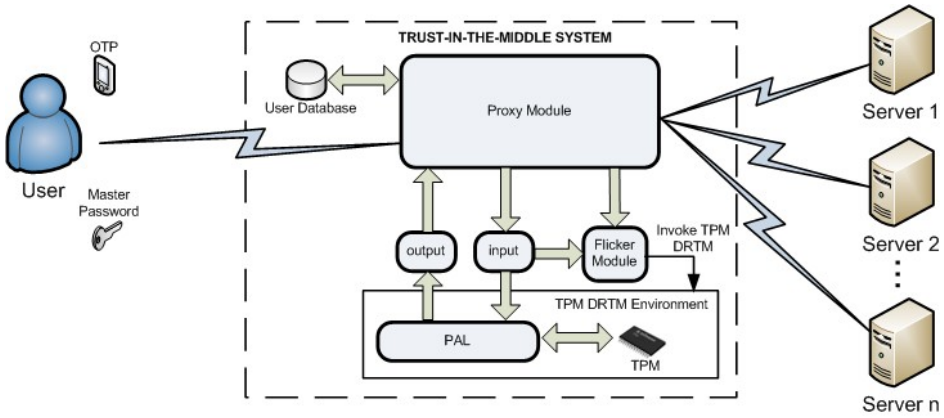


Figure 8: Trust-in-the-Middle System Architecture

In the following sections, we first give an overall architecture of the system (see Figure 8) and explain the role of each entity in this architecture. Then, we introduce the code structure of security sensitive code named as Program Application Logic (PAL) executed in TPM DRTM protections and explain functions of the code. Finally, we give the details of protocols used in the proposed system and explain their operation.

III.2.3 Architecture and Technology

In this section, we explain the architecture of Trust-in-the-Middle system together with the technologies used. The architecture of Trust-in-the-Middle is illustrated in Figure 8. There are six main components in this architecture; Proxy Module, PAL, Flicker Module, User Database, Input File and Output File.

Proxy module takes an important role in all system services. It supports SSL MITM functionality and incorporates the engines implemented to detect the login and update services of web sites and to insert user credentials into the correct fields. An important functionality of proxy module is to manage the communication with PAL by invoking TPM DRTM environment with the help of Flicker module. PAL is a piece of code responsible for performing security critical operations in TPM protection. Whenever a TPM DRTM environment is required for executing the PAL, proxy module invokes Flicker module, responsible for preparing the relevant environment and the operating system to run TPM DRTM. PAL is executed in an isolated environment provided by TPM DRTM. It is not possible to communicate directly with PAL through user level modules during its execution. The only way to communicate with PAL at this phase is using input and output files. User database holds the information about users and servers, encrypted credentials of users and proxy authentication passwords.

Two important technologies used in the architecture of Trust-in-the-Middle are described next.

TPM DRTM with Intel TXT Technology: As the platform supporting TPM DRTM operations, we use Intel Trusted Execution Technology (TXT). The platform establishes the authenticity of a measured launched environment (MLE) and protects this environment from potential corruption. MLE then establishes an isolated environment for itself and additional software it may execute [40].

In order to be able to launch MLE, first of all, an Authenticated Code (AC) Module, which is specific for the chipset and digitally signed by the chipset vendor, should be loaded into the memory. Only when AC module and MLE are in memory, the launching environment can invoke an instruction (i.e., GETSEC[SENTER]) which initiates the TPM DRTM functionality on the processor. This specific command

brings the chipset and CPU in a stable state and loads, validates and executes the AC. AC module then ensures that the platform has a proper configuration and measures and launches the MLE. These measurements are stored in specific PCRs using TPM Extend operations. In our proposed system, the security critical code (PAL) runs on this launched environment.

Flicker: Intel TXT offers capabilities to use TPM DRTM environment and run security sensitive code on it. As we mentioned, TPM DRTM is only invoked when the relevant AC Module and the MLE are in the memory. In addition, before DRTM invocation there are other requirements such as preparing the PAL and locating it on the right section in the memory and backing up the system state in order to resume it after PAL has finished its job and creating a directory structure to provide data exchange with PAL. For all these purposes, we use Flicker developed by McCune et al. [44].

Flicker is written as a SYSFS module which is a virtual file system capable of exporting information about devices and drivers from kernel to user space so that it becomes possible to make data exchange between a user level application and the Flicker module.

Whenever a DRTM operation is needed, Flicker module is loaded into the kernel and invoked with input parameters and SLB (Secure Loader Block) which includes the PAL. Then, Flicker suspends the OS by saving the existing state and gives control to the loaded SLB by executing the DRTM command. After SLB has completed its work and the relevant PCRs are extended, Flicker takes control again and resumes the operating system. PALs can use TPM-based sealed storage to maintain state across Flicker sessions, enabling more complex applications [44].

III.2.4 PAL Overview

PAL is a name we adopt from Flicker and use it to refer to the security sensitive code executed in TPM DRTM protection.

Since TPM DRTM operation provides a restricted environment where all the interrupts are disabled, the complex software programs requiring user-level operations such as Proxy Module in our application are not included directly in PAL. We prefer keeping the PAL as small as possible by including the most critical parts of the operations (i.e., Integrity Measurements, Seal/Unseal operations and Extend operations) and leave the other user level operations to other modules.

PAL is executed by Flicker module invoked from user space. After Flicker prepares the required environment, it invokes TPM DRTM and gives the control to PAL. PAL execution is carried out in a secure and isolated environment which cannot be accessed from user space. For this reason a user level application cannot communicate directly with PAL during the TPM DRTM session. The input data for PAL has to be written in a specific input file of Flicker which is then located in a known address in the memory. If PAL has any output after its execution, this is also written in a known address in the memory and made available to user level applications via Flicker output file. Since these input and output files can be accessed by any user level application, the data is encrypted before written down on these files.

An overview of PAL implemented in our system is presented in Figure 9. The operation of PAL is performed in four phases: Integrity Measurements and Extend Operations Phase, Input Phase, Main Operations Phase and Output Phase.

Integrity Measurements and Extend Operations Phase. The security of the Trust-in-the-Middle system depends on the relationship between PAL, Proxy and Flicker modules. PAL is the main software that is responsible for checking the integrity of other software modules and making the seal and unseal operations. PAL is executed in TPM DRTM protection and the hash of PAL is extended into PCR18. Therefore, the integrity measurement of PAL is directly provided by TPM. The integrity measurements and Extend operations of the other modules are performed by the PAL (see Figure 9). As a result, other modules are also added into the TCB (Trusted Computing Base). By verifying the integrity of the TCB, user credentials on Trust-in-the-Middle is protected against malicious infection.

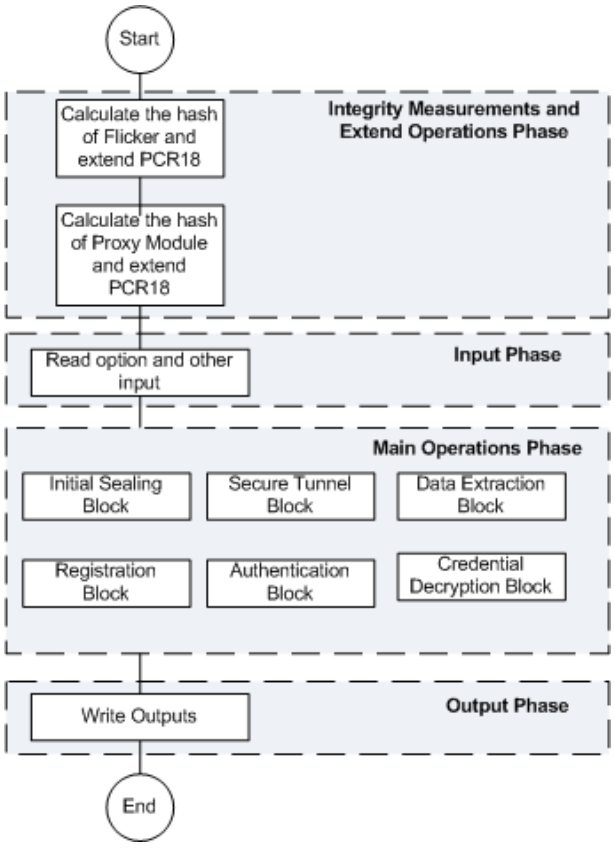


Figure 9: PAL Overview

To establish the integrity of the modules, we set up a trust chain utilizing TPM Extend operations. Furthermore, we use this chain and TPM Seal operation to protect the private portion of the key pair used in encryption and decryption operations for user credentials. The SML (stored measurement log) of our trust chain is given below:

$$SML \leftarrow \{PAL, Flicker\ Module, Proxy\ Module\}$$

Whenever a TPM Seal/Unseal operation is needed, TPM DRTM environment is invoked with the PAL. After preparing the environment and loading the MLE securely, TPM sets the value of PCR18 to ”0”, extends it with the hash of PAL and then starts executing the PAL code. All operations till this point are the standard operations of TPM DRTM functionality. PAL then performs two more Extend operations for Flicker module and Proxy Module (PM). The final value of PCR18 is determined by the following hash chain:

$$PCR18 \leftarrow H(H(PM)+H(H(Flicker)+H(H(PAL)+”0”)))$$

Using this trust chain and TPM Seal and Unseal operations, we ensure that the extraction of sealed data (i.e., sealed private keys) succeeds only if the integrity measurements of PAL, Flicker and PM are verified to be correct.

Input Phase. After Integrity Measurements and Extend Operations Phase, PAL reads the input (input file was retrieved and located to a specific memory location by Flicker Module). We note that there is a specific input called option indicating the operation block to be invoked by PAL.

Main Operations Phase. There are six main operations that can be executed by PAL according to the input option value. Initial Sealing Block is used to seal the public key of Proxy Module during the trusted boot. Secure Tunnel Block is used to establish a secure tunnel between PAL and a remote entity. Data Extraction Block is used to extract the data received through the secure tunnel. Registration Block is used to create and store proxy authentication passwords protected by TPM Seal operation. Authentication Block is used to carry out the proxy authentication with the credentials received from user. Credential Decryption Block is used to unseal the private key used in credential encryption previously and decrypt the credentials with the unsealed private key.

Table 3: Executed PAL Blocks in Main Operation Phase corresponding to Trust-in-the-Middle Services

Service	Executed PAL Blocks		
Registration	Registration Block	Secure Tunnel Block	Data Extraction Block
Authentication	Authentication Block	Secure Tunnel Block	Data Extraction Block
Operation(Enrollment)	Credential Decryption Block	-	-
Operation(Submission)	Credential Decryption Block	-	-
Operation (Update)	Credential Decryption Block	-	-

Output Phase. At this phase, if it is needed, PAL writes the output to a specific memory location which is then written into the output file.

Integrity Measurements and Extend Operations, Input and Output Phases are usually executed for every PAL invocation. However; the blocks executed in Main Operation Phase differ according to the service type. Table 3 shows executed PAL blocks corresponding to each service.

III.2.5 Auxiliary Protocols

In this section, we describe the operation of auxiliary protocols used by the main protocols of Trust-in- the-Middle system (see Table 2).

Attestation Protocol: Attestation has a crucial role in the proposed system and is used to verify that the correct PAL is executed before establishing the secure tunnel. Attestation Protocol is given in Table 4.

Table 4: Attestation Protocol

1a.Verifier	: Generate nonce
2a.Verifier→Attestor	: Attestation Request (nonce, PCRno)
3a.Attestor	: Loadkey(AIKkey)
4a.Attestor→TPM	: Execute TPM Quote Operation
5a.TPM	: Quote=sig{PCR,nonce}AIKpriv
6a.TPM→Attestor	: Quote
7a.Attestor	: Generate (SML)
8a.Attestor→Verifier	: Quote, SML and cert(AIKpub)
9a.Verifier	: validate cert(AIKpub)
9b.Verifier	: validate sig{PCR,nonce}AIKpriv
9c.Verifier	: validate SML using PCR

Attestation Protocol starts with a nonce value generation. This nonce value has an important role in providing the freshness of the attestation and preventing replay attacks. Verifier (the client) then sends an attestation request to the attestor (the proxy) with the nonce and a PCR number indicating which PCR is to be used in attestation. Receiving this request, the attestor needs to perform a TPM Quote operation. For this purpose, an AIK Key is loaded into the TPM slot first. This AIK Key is an encrypted key bound with Storage Root Key which is a non-migratable key

embedded in the TPM nonvolatile memory. Extraction of private key from AIK Key can only be done inside the TPM and cannot be accessed from outside. After loading AIK Key into the TPM, attestor performs TPM Quote operation and obtains a Quote which is formed by concatenated PCR and nonce values signed by AIK private key. Then, attestor creates the SML (the hashes of each entity creating the trust chain), and sends SML, Quote and the AIK certificate to the verifier. The verifier validates the AIK certificate, the signature and the nonce value. Then, it calculates the final hash value from the SML and validates the PCR value.

Secure Tunnel Protocol: In order to send sensitive data directly to the PAL, the client establishes a secure tunnel using the protocol given in Table 5.

Table 5: Secure Tunnel Protocol

1a.Client→PM	: Secure Tunnel Request
2a.PM→PAL	: Invoke PAL with "Secure Tunnel Block"
3a.PAL	: Generate an RSA keypair (PALpriv,PALpub)
3b.PAL	: sealedPALpriv=TPMSeal(PALpriv,PCR18)
3c.PAL	: TPMExtend(PALpub,PCR18)
4a.PAL→PM	: PALpub, sealedPALpriv
5a.PM→Client	: PALpub
6a.Client	: Execute Attestation Protocol and validate PALpub
7a.Client	: encSenData=Enc(SenData)PALpub
8b.Client→PM	: encSenData
9a.PM→PAL	: Invoke PAL with "Data Extraction Block"
9b.PM→PAL	: encSenData, sealedPALpriv
10a.PAL	: PALpriv = TPMUnseal(sealedPALpriv,PCR18)
10b.PAL	: SenData=Dec(encSenData)PALpriv

Receiving a secure tunnel request, PM invokes a PAL session and initiates a Secure Tunnel Block. PAL generates an RSA key pair. PAL then seals PAL private key with PCR18 and makes a TPM Extend operation to the PCR18 with the hash of PAL public key. PAL ends its session by providing PAL public key and sealed PAL private key as output. PM sends PAL public key to the client. Client first executes Attestation Protocol to verify that the correct PAL has been executed and the output values of the PAL are correct. If the verification is successful, it encrypts the sensitive data (e.g., the master password) with PAL public key and sends it to the PM.

PM invokes another PAL session and initiates Data Extraction Block. PM also provides sealed PAL private key and encrypted sensitive data as input. Upon receiving these input, PAL performs a TPM Unseal operation to recover the PAL private key. This operation succeeds only if the value of PCR18 is as same as the value in the previous PAL session ensuring the integrity of the PAL. After the unseal operation, PAL decrypts encrypted sensitive data with its PAL private key. With this Secure Tunnel Protocol, we prevent any malicious entity between the client and the PAL to access the sensitive data in plaintext.

Credential Decryption Protocol: Credential Decryption Protocol, given in Table 6, is an auxiliary protocol called by Credential Enrollment, Submission and Update Protocols to decrypt previously encrypted user credentials using a TPM protected (sealed) private key. In the protocol, Proxy Module first generates a nonce value and invokes a PAL session by initiating the Credential Decryption Block. It then sends previously encrypted credentials, sealed PAL private key, sealed public key of proxy module and the nonce value as input. Nonce value is used in the encryption of data sent to Proxy Module with the goal of preventing replay attacks. Receiving the input, PAL first unseals the sealed public key of Proxy Module (the public key of proxy is sealed by PAL during the trusted boot process discussed in Section III.2.6). Then, PAL unseals the sealed PAL private key and uses it to decrypt the encrypted

credentials. After obtaining the credentials in plaintext, PAL first concatenates the credentials with the nonce value and encrypts them with the unsealed public key of proxy module. PAL then sends them to Proxy Module. Receiving this, Proxy Module performs a decryption operation by using its private key and accesses the credentials and the nonce in plaintext. It validates nonce before using the plaintext credentials.

Table 6: Credential Decryption Protocol

1a.PM	: Generate nonce
2a.PM→PAL	: Invoke PAL with "Credential Decryption Block"
2b.PM→PAL	: encCredwithPAL, sealedPALpriv, sealedPMPub, nonce
3a.PAL	: PMPub=TPMUnseal(sealedPMPub,PCR18)
3b.PAL	: PALpriv=TPMUnseal(sealedPALpriv, PCR18)
3c.PAL	: Credentials=Dec(encCredwithPAL)PALpriv
3d.PAL	: encCredwithPM=Enc(Credentials, nonce)PMPub
4a.PAL→PM	: encCredwithPM
5a.PM	: {Credentials, nonce}=Dec(encCredwithPM)PMpriv
5b.PM	: Validate nonce value

III.2.6 Main Protocols

In this section, we explain the main protocols used in registration, authentication, credential enrollment, update and submission services (see Table 2). We describe the protocol used in initial sealing, first.

Trusted Boot and Initial Sealing: In Trust-in-the-Middle system, Proxy Module and Flicker Module start running as a service when the system is booted. However, PAL is not a service running continuously. The integrity of the modules running as a service is provided by a trusted boot operation. With the help of Intel TXT, LCP (Launch Control Policies) [77] and tboot [78], we carry out a trusted boot process and

prevent booting when one of the hash values in the boot chain is changed. Trusted Boot (tboot) is an open source, pre- kernel/VMM module that uses Intel(R) Trusted Execution Technology (Intel(R) TXT) to perform a measured and verified launch of an OS kernel/VMM [42]. Launch Control Policy (LCP) is the verification mechanism for the Intel TXT verified launch process. LCP is used determine whether the current platform configuration or the environment to be launched meets a specified criteria. Policies may be defined by the Platform Owner, and/or, as a default set by the Platform Supplier (Please see [77] for details).

Table 7: Initial Sealing Protocol

1a.PM	: Generate an RSA keypair (PMpriv,PMpub)
1b.PM	: TPMExtend(PMpub,PCR15)
1c.PM	: Validate the hash of PAL
2a.PM→PAL	: Invoke PAL with "Initial Sealing Block"
2b.PM→PAL	: PMPub
3a.PAL	: hash=Hash("0"+Hash(PMPub))
3b.PAL	: validate PMPub by checking hash=PCR15
3c.PAL	: sealedPMPub=TPMSeal(PMPub,PCR18)
4a.PAL→PM	: sealedPMPub
5a.PM	: Store sealedPMPub

During the boot process, Proxy Module starts its operation by executing the initial sealing protocol given in Table 7. In this protocol, first an RSA key pair is generated and the hash of public portion is extended into one of the empty PCRs (PCR15 in our implementation) which has a default value"0". This PCR is used by PAL in order to verify the public key of the Proxy Module.

The hash of PAL is validated by Proxy Module and if it is correct, PAL is invoked for an initial sealing operation using the proxy public key. Receiving the public key, PAL first calculates the hash of public key, then concatenates it with ”0” and again performs the final hash operation. If the calculated value matches with the PCR15 value, it ensures that public key belongs to the proxy module. It performs TPM Seal operation on the public key and the sealed public key is stored by the Proxy Module for later use. The private key of the Proxy Module is not written to a file and kept in the memory as long the Proxy Module runs as a service (see Section IV.1 for the security issues herein).

Proxy Registration: If proxy authentication password is compromised, all credentials enrolled in the Trust-in-the-Middle becomes vulnerable. Therefore, we assume users perform registration only using secure platforms. The protocol in Table 8 is executed for proxy registration.

Table 8: Registration Protocol

1a.Client→PM	: Registration Request
2a.PM→PAL	: Invoke PAL with ”Registration Block”
2b.PM→PAL	: sealedPassList
3a.Client↔PAL	: Execute Secure Tunnel Protocol
3b.Client→PAL	: Userid, MasPass and SecPhrase
4a.PAL	: Generate OTP passwords (OTP[list])
4b.PAL	: PassList= TPMUnseal(sealedPassList,PCR18)
4c.PAL	: Update PassList with Userid, MasPass and OTP[list]
4d.PAL	: sealedPassList=TPMSeal(PassList,PCR18)
5a.PAL→PM	: OTP Parameters, sealedPassList
6a.PM→DB	: sealedPassList
7a.PM→Client	: OTP Parameters
8a.Client	: Generate OTP[list] by using OTP Parameters and SecPhrase

Upon receipt of a registration request, Proxy Module invokes a PAL session, initiates a Registration Block and gives sealed password list to the PAL as input. The Secure Tunnel Protocol is executed between PAL and the client in order to establish a secure tunnel. User determines a master password and a secret phrase which will be used in the generation of one time passwords. As we mentioned before, Trust-in-the-Middle offers two password options for the user, master password and one-time passwords. PAL runs an OTP generation algorithm with the given secret phrase to generate a list of one-time passwords.

The sealed password list, including all user IDs and passwords of all registered users, is TPM protected. Hence, for a new registration, the list is unsealed first. Then, the new record is added and the list is sealed again. PAL passes the OTP parameters and sealed password list to the Proxy Module which stores sealed password list and sends the OTP parameters to the client. Client generates the same OTP List using the secret phrase and the parameters and outputs the parameters and the OTP list to the user. User can use an OTP application (e.g., mobile phone application) loaded with the parameters provided. Alternatively, he can print the list for manual use.

Proxy Authentication: For proxy authentication, the protocol presented in Table 9 is executed. Receiving an authentication request from client, Proxy Module invokes a PAL session, initiates Authentication Block and provides the sealed password list as input. Before password is sent, a secure tunnel between client and the PAL is established. Client stores the received PAL public key used in secure tunnel establishment for later use. So that during the user's session only one key generation operation is performed. User enters his user ID and password which is then sent to the PAL through the established secure tunnel. PAL performs an unseal operation and validates the password. It extends PCR18 with "1" indicating the success of the operation, otherwise it extends PCR18 with"0". It also writes the validation result (fail or success) into the output file. Receiving this output, Proxy Module also checks

the value of PCR18 for validation. If the validation result is fail, it sends an error message to the client and aborts. Otherwise, authentication is successfully achieved.

Table 9: Authentication Protocol

1a.Client→PM	: Authentication Request
2a.PM→PAL	: Invoke PAL with "Authentication Block"
2b.PM→PAL	: sealedPassList
3a.Client↔PAL	: Execute Secure Tunnel Protocol and send User ID and Password (master or OTP[n])
4a.PAL	: PassList= TPMUnSeal(sealedPassList,PCR18)
4b.PAL	: validate User ID and Password from PassList
4c.PAL	: If (valid) TPMExtend("1",PCR18) else TPMExtend("0",PCR18)
5a.PAL→PM	: Validation Result (fail or success)
6a.PM	: Check PCR18 Value
6b.PM→Client	: If validation = "fail", send error and abort

Credential Enrollment: For credential enrollment, the protocol presented in Table 10 is executed. As we mentioned previously, Proxy Module runs as an SSL MITM and intercepts the SSL connection of the user. First, PM checks whether the user is authenticated. Then, it sets up SSL connections, one for client and one for the target server. Before the SSL connection with the target server, PM validates the target server's certificate. If the visited web page is a login page, PM runs a query in User DB to understand whether the user has previously enrolled credentials for the target web site. If not, it forwards the login page to the user with empty credential fields. Receiving this login page the user enters credentials and clicks the submit button. The browser add-on of client software recognizes that user has filled credential fields and encrypts them with PAL public key which has been stored during the proxy authentication.

Receiving the encrypted credentials, PM first executes Credential Decryption Protocol in order to obtain the plaintext credentials. Then, it inserts the credentials

into the relevant fields on the login page and performs submission. If the user is successfully authenticated to the server, PM completes the enrollment protocol by storing the encrypted credentials, sealed PAL private key, and the other user information into the user database

Table 10: Credential Enrollment Protocol

1a.User→PM	: SSL web site connection request
2a.PM	: Check whether user is authenticated
3a.PM↔User	: set up SSL connection with the User
4a.PM↔Server	: validate Target Certificate and set up SSL connection with the Target Server
5a.PM	: is login page?
6a.PM↔DB	: does user have enrolled credentials?
7a.PM→User	: If not, send login page with empty fields
8a.User	: enter credentials
9a.Browser	: encCredwithPal= Enc(credentials)PALpub
10a.Browser→PM	: encCredwithPal
11a.PM↔PAL	: Execute Credential Decryption Protocol
12a.PM	: Insert original credentials
13a.PM→Server	: Submit credentials
14a.Server→PM	: User is authenticated
15a.PM→DB	: store encCredwithPal, sealedPALpriv
16a.PM→User	: User is authenticated

Credential Submission: Credential submission protocol is given in Table 11. Credential Submission Protocol starts as same as the Credential Enrollment Protocol. But, if the credential of the authenticated user for the target web site has already been enrolled in user DB, PM generates dummy credentials and inserts them into the credential fields of login page and sends it to the user. We note that the original credentials are not sent for security reasons. Upon user's click on the Submit button, PM retrieves the encrypted credentials of the user and sealed PAL private key from user database, executes Credential Decryption Protocol and obtains the plaintext

credentials. Then, it inserts the credentials and completes the submission operation by submitting the login page to the target server.

Table 11: Credential Submission Protocol

1a.User→PM	: SSL web site connection request
2a.PM	: Check whether user is authenticated
3a.PM↔User	: set up SSL connection with the User
4a.PM↔Server	: validate Target Certificate and set up SSL connection with the Target Server
5a.PM	: is login page?
6a.PM↔DB	: does user have enrolled credentials?
7a.PM	: If yes, create dummy credentials (DumCred)
8a.PM→User	: insert DumCred and send login page
9a.User→PM	: submit login page
10a.PM↔DB	: Retrieve user's encCredwithPAL and sealedPALpriv
11a.PM	: Execute Credential Decryption Protocol
11b.PM	: Insert original credentials
12a.PM→Server	: submit credentials
13a.Server→PM	: User is authenticated
14a.PM→User	: User is authenticated

Credential Update: The protocol for credential update is given in Table 12. Credential Update Protocol is similar to Credential Submission Protocol. If the visited web site is a password update page, PM checks the User DB to understand whether the user has previously enrolled credentials for the target site. If so, it sends the update page to the user by filling in the old credential fields with the dummy credentials. Then, user fills in only the new credentials part on the update page. The new credentials are encrypted with the PAL public key. Receiving the encrypted credentials, PM first executes Credential Decryption Protocol and obtains the new credentials in plaintext. Then, it retrieves the encrypted old credentials and executes Credential Decryption Protocol to obtain the plaintext old credentials of the user. PM

inserts the old and new credentials into the relevant fields on update page and submits the page to the target server. If the update operation on server side is successfully completed, PM updates the user database with the new credentials and new sealed PAL private key.

Table 12: Credential Update Protocol

1a.User→PM	: SSL web site connection request
2a.PM	: Check whether user is authenticated
3a.PM↔User	: set up SSL connection with the User
4a.PM↔Server	: validate Target Certificate and set up SSL connection with the Target Server
5a.PM	: is update page?
6a.PM↔DB	: does user have enrolled credentials?
7a.PM	: If yes, create dummy credentials (DumCred)
8a.PM→User	: insert DumCred in old credentials field and send update page
9a.User	: enter new credentials (NewCred)
10a.Browser	: encNewCredwithPal=Enc(NewCred)PALpub
11a.Browser→PM	: encNewCredwithPal
12a.PM	: Execute Credential Decryption Protocol and get NewCred
13a.PM↔DB	: Retrieve user's encCredwithPAL and sealedPALpriv
14a.PM	: Execute Credential Decryption Protocol and get Old Credentials (OldCred)
14b.PM	: Insert OldCred and NewCred
15a.PM→Server	: Submit Update Page
16a.Server→PM	: User credentials are updated
17a.PM→DB	: Update user's record with the new encNewCredwithPal and sealedPALpriv
18a.PM→User	: User credentials are updated

III.3 Implementation Details

In this section, we give brief information on our prototype implementation of Trust-in-the-Middle system. In our prototype system, we modify and use an SSL MITM

Proxy software publicly available [79]. We use OpenSuse 11.2 operating system on an HP DC7800 having a TPM v1.2 chip and Intel TXT support [77]. For TPM DRTM operations, we use Flicker v.0.2 [44]. For trusted boot we use tboot [93]. For the client software which runs registration, attestation and tunnel (SOCKS) establishment, we use a modified Putty SSH client software [80] which is installed together with the Trust-in-the-Middle SSH certificates in a flash disk whose read only feature is activated. This modified Putty SSH software does not let the user to continue the SSH session if the certificate of the connected server does not match with the one previously stored into the read only flash disk. The reason of using a second tunnel besides SSL tunnels in our proposed solution is the need to set up browser independent tunnels where all the web traffic is forwarded through. In order to open an SSH SOCKS tunnel, we configure specific settings on Putty software [81].

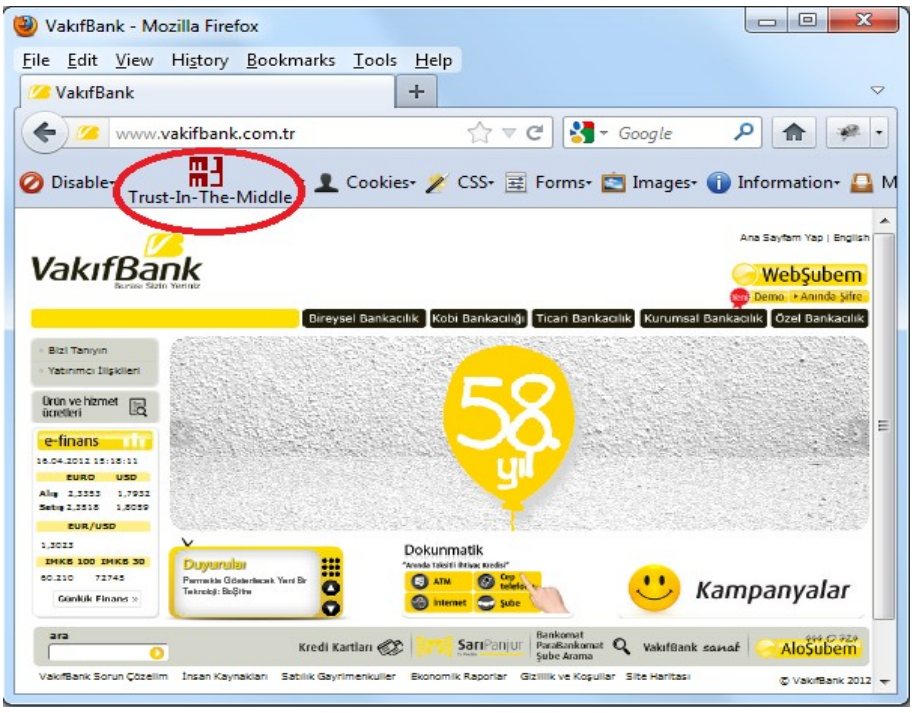


Figure 10: Trust-in-the-Middle Browser Add-on

For one time passwords, we use OPIE (One-time Passwords in Everything) [82] which is an S/Key One-Time Password implementation. On client side, we use 1Key which is an OPIE iPhone application [83] for one-time passwords. As the browser add-on, we modify and use a Firefox add-on, Proxy Switch [84]. With this add-on users could configure the proxy settings automatically (see Figure 10). This add-on is also responsible for performing encryption operations using the PAL public key.

III.4 Performance Evaluation

In this section, we give our performance evaluation results of our prototype system.

III.4.1 Methodology

We examine the performance of the Trust-in-the-Middle in two perspectives; system perspective and user perspective.

From system perspective, we first give the measurements of core TPM operations and the measurements of auxiliary and main protocols. In our time calculations on Trust-in-the-Middle system, we used RDTSC [85] instruction which gives us the number CPU cycles since the last reset. Then we convert the CPU cycles into milliseconds by using CPU Speed. We note that this instruction does not make sense in today's multi core CPUs. However; since our prototype runs on single CPU, this method still gives us accurate results.

From user perspective, we have tried to measure average time user spends to complete his process when Trust-in-the-Middle is used or not used. There are 5 main user operations; Registration, Authentication, Enrollment, Submission and Update. We give the time needed for each operation in section III.4.3. However; we have only put on the table submission and update operations from user's perspective in order to

provide a comparative analysis. Since registration and authentication operations are not executed continuously and enrollment operation does not show difference from user's normal login behavior.

When analyzing the performance of submission and update operations, we have carried out our experiments for three different login types; standard keyboard input, input from a virtual keyboard which has a fixed character sequence and input from a virtual keyboard whose character sequence changes at each click. We call the last process as dynamic virtual keyboard input. By using these three different methodologies, we performed one successful login without any input error and one login with one input error. So for the latter, we had to enter some credentials twice.

All the performance measurements were carried out getting average value of 100 runs of different executions. For the submission and update operations, 20 different average internet users are used. The measurements are recorded after each user carried out 30 exercise.

III.4.2 Experimental Environment

Our prototype Trust-in-the-Middle system is built on HP DC7800 with Intel Core2 Quad CPU Q9300 2.50 GHz and v.1.2 Infineon TPM. As client machine, we use a Sony Vaio laptop equipped with Intel Core i7-3520M CPU 2.90 GHz and 8.0 GB RAM, which is 3 hops away from the server with an average 27ms ping rate. We use OpenSUSE 11.2 on server and Windows 7 on client machine.

We have tested our proxy in top 10 e-banking sites in Turkey (Türkiye İş Bankası, Ziraat Bankası, Garanti, Akbank, Yapı ve kredi, Halk Bankası, Vakıfbank, Finansbank, Türk Ekonomi Bankası, Denizbank) and successfully used Trust-in-the-Middle.

For detailed user performance analysis, we have imitated the login system of Vakıfbank e-banking service. In this system there are 3 steps for successful login. At first step, a 12 Digit Customer ID and at least an 8 digit password including letters and numbers are required. Virtual keyboard can be used only at password input field. At second step, customer is required to enter at least 6 numeric digits PIN number. Virtual keyboard can also be used at this step. Virtual keyboards can be used with either fixed key sequence or dynamic key sequence preferably. If customer successfully passes the first two steps, a one-time password is sent to his mobile phone as a last step.

In our test server, we implement a copy of first two steps using PHP, Apache and SSL on an OpenSuse machine. For test credentials we have used a 12 numeric digits customer ID, 8 digits (4 numeric, 4 alphabetic) password and 6 numeric digits PIN number. Each digit of Password and PIN number is selected different from each other. We have used "(microtime(true) * 1000)" PHP expression for time measurements in milliseconds. We have calculated the time at the beginning and at the end of the operations and found the difference by subtracting the two measurements. We have used Mozilla as browser.

III.4.3 Server-Side Measurements

In this section, we take into consideration of each operation executed on proxy system. We do not consider the time elapsed on user side or on network. We first give below the measured performance results of the core TPM operations used in different protocols:

Table 13: Core TPM Operations

Operation	Time (ms)
Quote	352
SENTER	32
Seal	248
Unseal	397
Extend	2
Nonce	1,3
Encryption	4,6
Decryption	47,6
RSA Key	196,8

We see that, Unseal and Quote operations are the most expensive TPM operations in our prototype system.

Table 14: Measurements of Auxiliary Protocols

Protocol	Time (ms)
Attestation	422
Secure Tunnel Protocol	1391,3
Credential Decryption	952,3
Initial Sealing Protocol	508,92

Table 15: Measurements of Main Protocols

Protocol	Time (ms)
Registration Protocol	2.105,3
Authentication Protocol	1.848,3
Credential Enrollment	1.013,5
Credential Submission	1.004,8
Credential Update	1.962,6

At table 14, the measurements of auxiliary protocols are displayed and table 15 shows the total system time required for main protocols which use auxiliary protocols. We see that most expensive operations are Registration, Credential Update and Authentication Protocols. Credential update executes two credential decryption operation. Registration and Authentication protocols execute secure tunnel protocol which is the most expensive auxiliary protocol and both of them also incorporate an expensive TPM operation, Unseal.

III.4.4 User-Side Measurements

In this section, we give user side experiment results in the tables 16, 17, 18, 19:

Table 16: Submission-Keyboard Input without Error

	Normal (ms)	Trust-in-the-Middle (ms)	Gain (ms)	Gain (%)
Keyboard Input	24.874,3	3.271,9	21.602,5	86,85
Fixed Virtual Keyboard Input	27.064,1	3.271,9	23.792,2	87,91
Dynamic Virtual Keyboard Input	35.667,1	3.271,9	32.395,2	90,83

Table 17: Submission-Keyboard Input with Error

	Normal (ms)	Trust-in-the-Middle (ms)	Gain (ms)	Gain (%)
Keyboard Input	36.104,5	3.271,9	32.832,7	90,94
Fixed Virtual Keyboard Input	47.455,4	3.271,9	44.183,5	93,11
Dynamic Virtual Keyboard Input	63.417,0	3.271,9	60.145,1	94,84

When we analyze the measurements of submission process according to different input methodologies, we see that Trust-in-the-Middle offers considerable gain on user side. At standard keyboard input Trust- in-the-Middle improves the performance at least 21 s. When we examine dynamic virtual keyboard input which is one of the most popular methods used especially at e-banking sites, Trust-in-the-Middle offers around 32 s in input without error and 60 s in input with error. The main reason of

these scores is that whatever type is used, the only thing user is expected to do while using Trust-in-the-Middle is to click the login button without entering any credentials. So the user completes the submission process with two clicks effort.

Table 18: Update-Keyboard Input without Error

	Normal (ms)	Trust-in-the-Middle (ms)	Gain (ms)	Gain (%)
Keyboard Input	16.402,8	11.127,3	5.275,4	32,16
Fixed Virtual Keyboard Input	23.825,2	17.185,4	6.639,8	27,87
Dynamic Virtual Keyboard Input	33.943,3	24.938,9	9.004,3	26,53

When we analyze the measurements of the update process, we see that the total gain of Trust-in-the-Middle is not as much as the submission process. The first reason is that, the update policy of our prototype implementation only requires the PIN update which means Trust-in-the-Middle update operation differs from the normal operation only in entering 6 digit current PIN number. The user should determine new PIN and enter it twice in all conditions. The second reason is that credential decryption operation adds nearly two seconds system delay when using Trust-in-the-Middle.

Table 19: Update-Keyboard Input with Error

	Normal (ms)	Trust-in-the-Middle (ms)	Gain (ms)	Gain (%)
Keyboard Input	29.534,0	28.044,6	1.489,4	5,04
Fixed Virtual Keyboard Input	40.256,6	35.174,9	5.081,7	12,62
Dynamic Virtual Keyboard Input	65.473,1	54.324,7	11.148,4	17,03

III.4.5 Final Remarks

Our proposed system Trust-in-the-Middle creates 1.000,4 ms system delay in submission and 1.962,6 ms system delay in update operations on the proxy server. For the first look, we can say that these scores are significant for a heavy load server and does not meet our performance expectations. When we compare our system with the SSH Server implementation of Mc.Cune et.al. in [44], the total time elapsed on

the client between the establishment of the TCP connection with the server, and the display of the password prompt for the user is 1.221 ms. compared with 210 ms for an unmodified server in [44] . So we understand that 1.011 ms has been added because of the TPM. According to these results, we see that although the submission operation of Trust-in-the-Middle takes almost similar time with this SSH login, update operation takes much more milliseconds.

Furthermore, in Mc. Cune et al. work [59], we see that the performance of TPM core operations changes significantly according to the used TPM chip. For example, in [44] Broadcom TPM's performance in Quote operation (972,7 ms) and Unseal operation (898,3 ms) is worse than Infineon's Quote (352 ms) and Unseal operations(397 ms) measured in Trust-in-the-Middle. On the other hand Broadcom TPM is faster than Infineon on seal operation with 10,2 ms compared to Infineon's seal operation with 248 ms in Trust-in-the-Middle.

On the other hand, when we look at the whole picture, we see that Trust-in-the-Middle offers significant gain on user's side. Especially for the applications requiring complex login procedures such as e-banking systems, Trust-in-the-Middle improves the performance of submission on user side more than 90%. Although registration and authentication operations creates a significant overload (totally 3.953,6), we see that it is acceptable since they are not continuously executed operations.

From system's perspective, we see that most of the latency is created by TPM. As indicated in [44], TPM devices are new devices in the market and have not yet proven themselves. Although the results of our experiments are not so promising on server side, we believe that performance improvement will be provided by the vendors in the future, as long as the TPM technology continues to spread in the market.

CHAPTER IV

SECURITY AND USABILITY ANALYSIS

This chapter consists of two main sections. In first section, we focus on possible threats against the given architecture of Trust-in-the-Middle. We classify threats according to its location; client based threats, network based threats, proxy based threats and verifier threats. We also add one more category which is specific threats against Trust-in-the-Middle to discuss threats that are valid for specific architectural functionalities of Trust-in-the-Middle. Then we discuss how those threats are addressed in Trust-in-the-Middle and explain defense mechanisms in detail. In the second section, we perform a very detailed usability, deployability and security analysis to compare Trust-in-the-Middle with the previous 20 systems and discuss at which points Trust-in-the-Middle has better and worse functionalities.

IV. 1 Security Analysis of Trust-in-the-Middle

In this section, we first give a threat table and define our threats in five categories; client based threats, network based threats, proxy based threats, verifier threats and special threats against Trust-in-the-Middle. Then we perform a security analysis for each threat in order to discuss whether Trust-in-the-Middle can offer protection or not.

While determining our threat list, we examined the threats in related work and try to address all of them in our threat table (see Table 20):

Table 20: Threats Mapping Table

No	**Threat Category**	**Mapping Attacks in Related Work**
T1	Keystroke Logging	Keystroke Logging [6] Key loggers [7] [49] [48] [46] [51]
T2	Screen Scraping	Screen loggers [49] Screen Scrapers [47] [46]
T3	Malicious Codes	Session Hijacking [8] [6] Spyware Infection [7] Malware [49] Malicious Software [48] Malicious Code [46] [51]
T4	Malicious Browsers	Compromised Browser [46] Browser Scripts [51]
T5	Phishing	Phishing [7] [49] [48] [47] [46] [50] [51]
T6	Transaction Generators	Transaction Generators [47] [51]
T7	Physical Observation	Shoulder Surfing [6]
T8	Password Attacks	Brute-force [8] Entropy Attacks [7]
T9	Lost or Stolen Physical Objects	Lost or Stolen OTP List [8] Stolen or Lost Cell Phone [6] [5]
T10	Interception Attacks	Network Sniffing [6] Passive Eavesdropping [6] Eavesdropping Channel [50]
T11	Passive SSL MITM Attacks	Replay Attacks [5] Man-in-the-Middle Attacks [7] [56] [51]
T12	Active and Real-time SSL MITM	Session Hijacking [8] [6]

	Attacks	Man-in-the-Middle Attacks [7] [56] [51]
T13	Pharming	DNS Redirection [7] Pharming [50] [51]
T14	Threats against Proxy Services	Threats against Proxy Services [16]
T15	Threats against User Database	Threats against User Database [16]
T16	Malicious Modification of Proxy Software	Malicious Modification of Proxy Software [16]
T17	Run-time and Memory based Attacks	Run-time and Memory based Attacks [16]
T18	Leaks from other Verifiers	Web Server's Disclosing Information [6]
T19	Collusion Attacks	Collusion [56]
T20	Threats against secure tunnel	Replay Attacks [5] RSA Attacks [98]
T21	Threats against SSH tunneling	Man-in-the-Middle Attacks [7] [56] Session Hijacking [8] [6]
T22	Malicious modification of Flicker Module or PAL	Malicious modification of Flicker Module or PAL [16]
T23	Modification of TPM PCRs	Modification of TPM PCRs [16]
T24	Modification of Input/Output of Flicker	Modification of Input/Output of Flicker [16]
T25	Physical Attacks to TPM	TPM Attacks [31]
T26	Attacks to Trust-in-the-Middle Encryption Schema	RSA Attacks [98]

IV.1.1 Analysis of Client Based Threats

Client based threats are examined under six categories; keystroke logging, screen scrapping, malicious codes, malicious browser, phishing and transaction generators.

T1 Keystroke Logging: Keystroke logging (often called keylogging) is the action of tracking (or logging) the keys struck on a keyboard, typically in a covert manner so that the person using the keyboard is unaware that their actions are being monitored. There are numerous keylogging methods, ranging from hardware and software-based approaches to electromagnetic and acoustic analysis [86].

Software keylogger is a piece of software code that runs on the user's machine and basically records all the keystrokes. These keyloggers can reside in computer systems in different forms; they can be hidden on operating system kernel as keyboard drivers for example, or they can be injected in keyboard APIs, hypervisors or in browsers.

Hardware keylogger is a kind of physical device capable of capturing and storing the keystrokes. Keyboard hardware type keyloggers are the most well-known ones which are located either as a separate apparatus between the keyboard and the input port of the computer or integrated in the keyboard itself. These devices often have an internal storage to record the keystrokes and can be accessed via a special key sequence.

Acoustic and electromagnetic keystroke logging are the most dangerous and hard-to-detect keylogging attacks. Acoustic Keyloggers work on the basis of converting sound into data. The idea is that each key on the keyboard makes a slightly different sound and a listening device can detect the subtle variations between the sounds of each keystroke and use this information to record what is being typed [87]. Wired and wireless keyboards also emit electromagnetic waves, because they contain electronic components. This electromagnetic radiation could reveal keystrokes. In [88], Vuagnoux and Passini found 4 different ways to fully or partially recover keystrokes from wired keyboards at a distance up to 20 meters, even through walls. They tested 12 different wired and wireless keyboard models bought between 2001 and 2008

(PS/2, USB and laptop). They found out that all of those keyboards are vulnerable to their attacks.

D1: At the time of proxy authentication, a keystroke logging software or device can capture proxy authentication password of the user. However; since one time password is implemented in Trust-in-the-Middle, this captured password will not be used second time after user has successfully authenticated himself. During registration, enrollment and credential update phases, user's secret data can be obtained by keystroke logging software. For this reason, Trust-in-the-Middle assumes that those operations are carried out only from secure computers. If user securely registers himself and carries out enrollments for his target web services, user's secret credentials are never entered on client side during normal operation. Hence; capturing user credentials during submission is not possible.

T2 Screen-Scraping: Screen scraping referred to the practice of reading text data from a computer display terminal's screen. Modern screen scraping techniques include capturing the bitmap data from the screen and running it through an OCR engine [89]. The attacks using screen-scraping are effective against virtual keyboard defense by recording the mouse movements on the screen and disclosing the characters of the password.

D2: Since user passwords are not displayed on screen and virtual keyboard is not used, screen-scraper attack is not valid for Trust-in-the-Middle schema.

T3 Malicious Codes: Malicious codes such as viruses, Trojan horses, backdoors and other similar spyware programs can have access all the resources on the client computer and are able to acquire user's secure credentials.

D3: Any malicious software that has the capability of using all system resources on client side can mount several attacks against user's credentials. If it performs only keystroke logging or screen scraping attacks, the same defense mechanisms with D1 and D2 are valid. However; a malicious software having control of all system resource can perform both keystroke logging and denial-of-service attack. So that after user's one time proxy authentication password is captured, denial of service attack can prevent user from logging into the proxy with the captured OTP. Hence; the captured one time password becomes valid for the malicious entity.

Possible Solution (PS3): A possible solution to this attack can be implementing one more verification through a second channel such as mobile phone data connection to the proxy in order to complete authentication procedure. By this way, without giving approval through the second channel, any malicious entity having captured the one time password cannot complete the authentication.

T4 Malicious Browsers: Web browsers, as one of the indispensable applications of the internet, integrate many complex applications such as ActiveX, Cookies, Plug-Ins, Flash Player, Java, Acrobat Reader and so on, which extend the browsers' functionality in order to increase usability and let users display the web pages including different graphics and animations without problem. Many web based applications require the user to install additional software to enable these functionalities. It is a known fact that most of those functionalities is enabled in the browsers' default settings as well. So any flaw or vulnerability in these applications in addition to the web browser's vulnerabilities itself, increases the security risks of the browsers. If one of those components is malicious and enabled by the user, it can have access to the computer resources or can acquire the sensitive data such as passwords on the web page before they are submitted to the web server.

D4: A malicious browser can capture user's credentials during enrollment and update operations if Trust-in-the-Middle assumptions for these operations do not hold. For the other operations, malicious browser cannot access any secret data, since they are inserted by Trust-in-the-Middle.

T5 Phishing Attacks: Phishing attacks are the attacks that tempt victim to connect to a forged link that has a connection to the attacker's fake web system. This link is mostly sent to the victim's email address. The goal of this attack is to steal user's secret credentials.

D5: During normal operation, since user does not enter his credentials, it is impossible to capture user's credentials using phishing attacks. However; as it is in the other threats, phishing attacks are valid for credential enrollment and update phases.

T6 Transaction Generators: Transaction generators are the malicious software that wait until user logs into a system and complete his authentication. Then they carry out malicious transaction on behalf of user through the established authentic session [47].

D6: Transaction generators are very serious attacks that bypass all the security mechanisms for the initial authentication. So this attack is also valid for the current Trust-in-the-Middle framework.

Possible Solution (PS6): A possible solution to prevent this attack is to send user a verification for each transaction. This can be an SMS based OTP or a data link verification through a second channel data link.

T7 Physical Observation: An attacker can obtain user's sensitive credentials by physical observation. This attack includes shoulder surfing, filming the keyboard, recording keystroke sounds [99].

D7: Since original passwords are not used in Trust-in-the-Middle during submission phase, the adversary performing physical observation can only obtain one time passwords.

T8 Password Attacks: An attacker can try various attacks to guess or crack the password of the user such as brute-force attacks which include checking all possible passwords systematically until the correct one is found or dictionary attacks which include trying a huge number of likely possibilities until the correct one is found, such as words in a dictionary or popular keyboard input i.e. QWERTY, ASDFG, 123456 and etc.

D8: Since original passwords are not used on client side, password attacks are not valid for Trust-in-the-Middle.

T9 Lost or Stolen Physical Object: If user uses a physical object in the authentication process such as a mobile phone or a piece of paper including user's passwords, an adversary having obtained this object can impersonate the user.

D9: Trust-in-the-Middle uses a mobile phone for storing one time passwords. However; if this mobile phone is obtained by an adversary, he cannot access one time passwords of the user as both the mobile phone and the one time password application are PIN protected.

IV.1.2 Analysis of Network Based Threats

In broad terms, network based threats can be categorized according to 3 main targets of security; confidentiality, integrity and availability. To define briefly, confidentially is the prevention of unauthorized disclosure of information, integrity is the prevention of unauthorized modification and availability is the prevention of unauthorized withholding of information.

In the scope of this thesis, we are focusing on the confidentiality and integrity issues than the availability. Possible network based attacks that target confidentiality and the integrity are given below:

T10: Interception Attacks: Interception attack is a type of passive attack where an unauthorized party has gained access to a service or data. In this attack, the adversary may either have an access to the network link between the user and the target server or can mount attack to the network devices to redirect or clone the traffic towards himself. After that, adversary listens the traffic using one of the packet sniffing tools. The aim of these attacks is mostly capturing sensitive data flowing on the network without making any modification on the original data.

D10: Since all the traffic between client and Trust-in-the-Middle is encrypted, interception attacks are not valid.

Man-in-the-Middle (MITM) Attacks: MITM attack is a form of attack where the attacker intercepts the traffic between victims and relays all the messages through itself making the victims believe as if they are talking directly to each other although all the traffic has been redirected by the attacker. In MITM attack, attacker establishes independent connections with the victims and relays the messages to each other.

During this operation, attacker may not only target the confidentiality of the sensitive data but also compromise the integrity by modifying the data before forwarding them to their destination.

Since most of the web based security critical systems require SSL, the following types of MITM are taken into account in our threat scenario:

T11: Passive SSL MITM Attacks: These attacks target the confidentiality of the user's network traffic in order to capture user's secret data to use it later on, i.e. a replay attack. Since the goal of this attack is just to eavesdrop the secret data without modifying the packet contents relayed, it is called as passive SSL MITM Attack.

T12: Active and Real-time SSL MITM Attacks: If MITM attacker performs a real-time session hijacking attack, this type of MITM attack is called as active and real-time MITM attack which is much more dangerous than passive MITM attacks. For example in passive SSL MITM, the attacker can obtain the user's password and if this password is a one-time password it cannot be used later. However, in active and real-time SSL MITM Attacks, attacker not only captures all the packet content but also hijacks session after the authentication completed. This means that whichever authentication scheme has been adopted, the last authenticated session sent to the victim's machine can be hijacked making all the previous defense mechanism obsolete. This is a very serious threat still valid in today's systems and networks.

D11-D12: Since SSL traffic is tunneled in SSH connection, Passive and Active SSL MITM attacks are not valid for Trust-in-the-Middle.

T13 Pharming Attacks: Pharming is another network based attack which aims at redirecting a website's traffic to another, bogus website. Pharming can be conducted

either by changing the hosts file on a victim's computer or by exploitation of a vulnerability in DNS server software. The main goal of this attack is to obtain victim's password after he has entered it on the attacker's fake system which is designed very similar to the original one. An MITM attack is also possible after redirection is performed by pharming attack.

D13: If Trust-in-the-Middle assumptions do not hold for credential enrollment and update phases, user's secret credentials can be captured by redirecting him to a fake site during enrollment and update phases. However; in normal credential submission phase, since user does not enter any credentials, pharming attacks are not valid.

Pharming attacks can also be used to redirect user's SSH connection to a malicious web site. In this case, user's SSH client software will warn user because of the incorrect certificate. However; if user ignores this warning message, he can connect to a forged system and give his one-time proxy authentication password. Capturing the OTP, malicious user can just break the connection of the user and connect Trust-in-the-Middle on behalf of him.

Possible Solution (PS13): One solution to this problem is to configure SSH client to check the digital signatures of the Trust-in-the-Middle from a safe resource (i.e. read only flash disk or CD) and directly prevent user to connect if the digital signature of target system does not hold without providing him any chance to proceed. We have used a similar system in our implementation using a modified Putty SSH client software [81].

IV.1.3 Analysis of Proxy based Threats

Since the architecture of Trust-in-the-Middle requires a third party authentication proxy system, we, in this section, give specific attacks to the system's proxy functionalities.

T14 Threats against Proxy Services: An adversary can mount attacks to capture user's credentials during registration, authentication, enrollment, submission and update operations of proxy. He can attack configuration and data files of the services. He can try to intercept the communication of services and try to obtain credentials in plain text. If the credentials are transferred encrypted, the adversary can mount replay attacks or can try attacks on encrypted passwords such as dictionary attacks, brute force attacks etc.

D14: Preventing disclosure of user credentials to a malicious code running on the proxy system is mandatory to establish the trustworthiness of the system. Credentials are under threat during (i) registration (ii) authentication (iii) enrollment, (iv) update and (v) submission services of Trust-in-the-Middle:

*Registration***.** During registration, master password and the secret phrase used in OTP generation are sent to PAL through the secure tunnel. OTP generation is performed by PAL in TPM DRTM protection and the passwords are stored after being sealed in PAL session. Only in TPM DRTM protection and in a correct PAL session, passwords can be unsealed.

*Authentication***.** Passwords for proxy authentication is sent to PAL through the secure tunnel. The authentication is performed in PAL and in TPM DRTM protection. Therefore, authentication process cannot be intervened by any other entity. The

authentication result is not only written in output file but also extended into PCR18. Hence, proxy module can verify the authentication result by checking PCR18.

Enrollment. The credentials are encrypted with PAL public key. Since the public key has been verified, we ensure that the encrypted credentials can only be accessed on the proxy by PAL. PM runs Credential Decryption protocol which outputs the credentials by encrypting them with PM's public key obtained after a TPM Unseal operation. So this public key is verified to be the one created during trusted boot with the Initial Sealing protocol. A malicious software cannot obtain the plaintext credentials because they are encrypted with the public key of the proxy module.

Submission. Submission operation is performed only if the user is authenticated to the proxy and the target certificate is verified. Encrypted credentials can only be decrypted with the unsealed PAL private key which is available to the correct PAL in TPM DRTM environment. User credentials are sent to proxy module by encrypting them with the public key of proxy which has been created during trusted boot and protected by initial sealing operation.

Update. Security of credential update is achieved similarly as in submission and enrollment operations.

T15 Threats against User Database: An adversary can mount attacks against user database holding users' credentials. He can try to obtain database administrator credentials and access the database. If the credentials are kept encrypted, user again can try to decrypt the credentials using several methods including dictionary attacks, brute force attacks.

D15: User database is sealed by PAL in TPM DRTM protections and kept encrypted on the system. The database can only be unsealed in TPM DRTM protections by PAL and only when the integrity measurements of PAL, Flicker and Proxy Modules hold. So it is not possible for a malicious entity on the proxy system to access user's credentials in the user database.

T16 Malicious Modification of Proxy Software: If an adversary can maliciously modify proxy software, he can not only obtain user's secret credentials, but also initiate an authentication session on behalf of the user. If there is another authentication such as sending one time password to the user's mobile phone, adversary can hijack the user's transaction and redirect the session to the attacker's side after user's authentication is completed.

D16: Modification of Proxy Module is a serious threat. We give below different attack scenarios and analyze how Trust-in-the-Middle provides protection.

Malicious code infects proxy module just before system reboot. If proxy module has been maliciously modified before system reboot, trusted boot can detect that the hash of the proxy module has changed and aborts the booting (See Section III.2.6).

Malicious code stops proxy service and infects proxy module. If proxy module has been modified after system boot, the final hash value of PCR18 would be different after Extend operations in PAL execution. As a result, the attestation fails and sensitive data of the user is not conveyed to the proxy during registration or authentication. Without the authentication, the tunnel cannot be established and the user cannot proceed using the Trust-in-the-Middle system.

Malicious code stops proxy service and runs a malicious copy of the proxy module. In this scenario, we assume that malicious code does not change the original code of proxy therefore the attestation may be successful during registration or authentication. However, malicious code cannot access user credentials in storage as they are encrypted with PAL public key attested by the client. It cannot access the plaintext credentials during credential enrollment, submission or update because the credentials conveyed between client and proxy are encrypted with PAL public key. After credentials are decrypted in PAL session, they are encrypted with the public key of the genuine proxy which was created and sealed during trusted boot. So malicious proxy module again cannot have access to the plaintext credentials.

Malicious code trying to modify sealed public key of proxy module. Trusted boot ensures the integrity of proxy module. While starting-up, PM executes the initial sealing protocol. During this protocol, PM generates an RSA key pair, extends the public portion into PCR15. It then sends the public portion to PAL to seal it. Since the PCR15 value is firstly extended by the genuine proxy module during trusted boot, any malicious module extending the value of PCR15 cannot make PAL to use its own public key for sealing due to failure in PCR15 verification (note that PCR values cannot be set to a specific value, they can only be extended).

T17 Run-time and Memory Based Attacks: An adversary can mount attacks against the run time files and the memory location of the software while it is being executed on the proxy and can capture user's secret credentials.

D17: TPM DRTM environment guarantees that PAL is executed in an isolated environment which cannot be intervened by any malicious entity. When PAL quits, Intel TXT ensures that all relevant memory locations are cleaned before exiting.

However, run-time memory based attacks to the proxy module capturing its private key or user credentials are possible.

Possible Solution (PS17): In order to prevent this attack, the credentials can be encrypted in the PAL session with public key of the target server. However, this protection violates the transparency requirements as it requires change on server side. Run-time software integrity problems were studied previously and several solutions were available in the literature [91-96]. We plan to incorporate these solutions into the Trust-in-the-Middle system in our future work.

IV.1.4 Analysis of Verifier Threats

We, in this section, examine the following two threats on verifier which checks the credentials of the user and performs authentication:

T18 Leaks from Other Verifiers: The information that a verifier can possibly leak can help an adversary to impersonate the user [99]. For example if multiple usage of the same password is of concern, then an adversary, having obtained the user's password from a malicious verifier, can easily use it on the other verifier's side.

D18: Since we do not make any modification on the existing credentials on server side, any malicious server can disclose user's secrets. So this attack is valid for Trust-in-the-Middle.

T19 Collusion Attacks: Colluding verifiers can disclose the user's identity. For example in federated single sign-on systems, if verifiers use the same identity for different target servers, then the target servers can predict the user's identity.

D19: Since Trust-in-the-Middle only replaces the credential fields with user's original credentials and does not use a specific data that may violate user's privacy, collusion attacks are not valid for Trust-in-the-Middle.

IV.1.5 Analysis of Specific Threats against Trust-in-the-Middle

There are some kind of threats that cannot be addressed as a client, network, proxy or verifier threats and specific to the deployed systems or utilized hardware in the architecture of Trust-in-the-Middle. In order to also cover those threats, we leave a separate session for specific threats against Trust-in-the-Middle in this section.

T20 Threats against secure tunnel: An adversary may attack to secure tunnel and try to either obtain user's credentials by breaking the encryption or replay the encrypted credentials.

D20: Before a sensitive data is sent to PAL, a Secure Tunnel protocol is executed. This protocol uses TPM DRTM functionality and the attestation protocol. During the PAL session, the generated output including the public key of PAL is extended into PCR18 and the private key of PAL is sealed. By verifying the PCR 18 value, the attestation operation ensures that the correct PAL has been executed and the received public key belongs to this PAL. After this verification, sensitive data is sent encrypted by PAL public key. This encrypted data can only be decrypted by PAL private key in TPM DRTM protection. Since PAL private key is protected by TPM seal operation, it is ensured that only the same PAL, running in TPM DRTM protection, can unseal the private key. As a result, it is ensured that sensitive data cannot be accessed either on network or in proxy once it has been encrypted on client side.

T21 Threats against SSH tunneling: An adversary can mount passive and active SSH Man-in-the-Middle Attacks and can intercept and modify the data sent through SSH tunnel.

D21: Since all the traffic through Trust-in-the-Middle is tunneled with SSH, the following attacks can be considered:

- Malicious SSH Client Software on client side: The same mechanisms with D3 is also valid for this attack.
- SSH Man-in-the-Middle Attack on Network: This case is taken into consideration in D13 and PS13 is also valid for possible solution. If we want more secure solution (PS21), we can locate both a special SSH client software which does not let user to connect the server whose signature cannot be verified and the file including the Trust-in-the-Middle's digital certificate into a read only media such as read only configured flash disk, CD or DVD and run the SSH Client software from this read only media whenever Trust-in-the-Middle will be used.

Since Trust-in-the-Middle implements PS21, it is not possible for an adversary to perform SSH MITM attacks.

T22 Malicious modification of Flicker Module or PAL: Malicious modification of Flicker may result in wrong execution of TPM DRTM or tamper with the loaded PAL binary file, input and output files. Malicious modification of PAL may reveal user's secret credentials to operating system without encryption protection or may encrypt user's credentials with the attacker's keys. A Malicious PAL can also reveal all the private keys used in critical encryption operations such as secure tunnel establishment without seal protection which will result in disclosure of user's secret data transferred into the secure tunnel.

D22: Boot time modification of Flicker module can be detected by the trusted boot. Load time modification of Flicker Module breaks the trust chain and leads to a failure in unseal operation. It can also be detected by the attestation.

If Flicker module loads a modified PAL, PCR18 would have a different hash value after TPM DRTM operation, which results in failure in seal/unseal operations and attestation.

T23 Modification of TPM PCRs: If an adversary is able to change PCR values used in TPM DRTM operation, he can write wrong hash values into PCRs and it will not be possible for the user to verify whether his secret credentials are processed securely in TPM DRTM protections on proxy or not.

D23: TPM DRTM ensures that no other operation can reset the value of PCRs to zero and TPM guarantees that PCR values cannot be set to a default value and can be written only by the TPM Extend operation.

T24 Modification of Input/Output of Flicker: An adversary can tamper with input and output files and can try to capture secret credentials from there or can input or output wrong values or files.

D24: The modification of input leads to a denial of service attack but does not reveal the credentials. If the input value is wrong, seal/unseal operations would fail. PAL cannot recover the required data and aborts. Critical output values are extended into PCR18 by PAL and are sent in the SML. Hence, the output values are verified in attestation.

T25 Physical Attacks to TPM: Integrity measurement of software modules written in special PCRs of TPM is a security critical operation which builds the trust chain used in Trust-in-the-Middle. If an adversary can find a way to reset the PCR values by performing physical attacks to TPM and extend its own hash values into the PCR, this will break the trust chain and enable adversary to easily tamper with the critical software modules used.

TPM is attached to LPC bus which has a 4-bit address/data bus, 33 Mhz clock, frame, and reset lines. In [31], the Dartmouth researchers have performed a physical attack to TPM chip and could simply grounded the LPC reset line with a short wire while the system was running. At that point, the PCRs are clear just like at boot [90].

D25: Since Trust-in-the-Middle uses a third party authentication proxy system, physical attacks to TPM chip is of concern. If PCRs used by Trust-in-the-Middle can be resetted by a physical attack as it is declared in [31, 90], we can say that the adversary can easily tamper with the software modules without being detected and hence acquire user's credentials. However; we know that TPM Reset Attack declared in [31] was effective for TPM v.1.1 and it was patched in TPM v.1.2. As it is declared in [97], with v.1.2 a new locality message was integrated in the system in order to set certain PCRs. Trust-in-the-Middle uses TPM v.1.2 so the relevant PCRs can only be resetted in Locality 4 which is only active in TPM DRTM protections. Hence we can say that this attack is not valid for Trust-in-the-Middle.

In another attack mentioned in [97], it is claimed that a man-in-the-middle device with a simple microcontroller attached to the clock, frame and 4 bit address/data bus, 6 lines in total, could drive the frame and A/D lines to insert a locality 4 "reset PCR" message. However; since this attack has not been implemented yet, we cannot be sure whether it will be successful or not.

T26 Attacks to Trust-in-the-Middle Encryption Schema: If an adversary can succeed to break the encryption schema (RSA) used by Trust-in-the-Middle, encrypted credentials can be obtained.

D26: In Trust-in-the-Middle, TPM uses RSA 2048 algorithm to encrypt user's credentials and establishes secure tunnel with the client. RSA is accepted as the de facto standard of the public key encryption and signatures. It is widely deployed worldwide and it has been used in many applications today. It can be considered as the basis of secure communication in the internet. Since its invention in 1977, many mathematicians and security experts have been examining the protocol all its underlying functions. However; no devastating attack could be found and its security has never been under doubt.

We see in the literature that the problems mentioned as RSA vulnerabilities are mostly because of misuse of the system, bad choice of parameters or flaws in implementations i.e. factoring problems such as trial division, Pollard's p-1 Method, Pollard's rho Method, Elliptic Curve Method, RSA function attacks such as Low Private Exponent Attack, Partial Key Exposure Attack, Broadcast and Related Message Attacks, Short Pad Attacks and implementation attacks such as Timing Attack, Power Analysis, Fault Analysis, Failure Analysis. All these attacks and possible countermeasures can be reached in [98].

As a conclusion, since we are unaware of any those attacks to a TCG compliant RSA 2048 encryption implementation used in Trust-in-the-Middle, we can say that Trust-in-the-Middle is resistant to the attacks to its encryption schema.

IV.1.6 Comparison of Trust-in-the-Middle with Other Proxy Based Systems

In this section, we compare the security of Trust-in-the-Middle with the similar previous systems (Imposter [4], Wu et al. [5], Delegate [6], KLASSP [7], URRSA [8], SecurePass [9]) which utilize third party authentication proxies.

In the threat table (see Table 21), each system is evaluated as either resilient to threat, or non-resilient to the threat. If a schema is almost resilient to threat, but not quite, it is indicated with "Quasi-" prefix.

Since all the systems implement one-time passwords or one-time secret in proxy authentication, they are Resilient-to-Keystroke-Logging and Resilient-to-Phishing. They also do not use virtual keyboard or any other mechanism that can reveal the user's password on the screen. Therefore, we rate all the systems as Resilient-to-Screen-Scraping.

Among the proxy based systems, only the Delegate is Resilient-to-Transaction-Generators as it sends verification request to user's mobile phone for each transaction and only the Imposter is vulnerable to standard password attacks (i.e. dictionary and brute force attacks) as user determines an 8 character secret phrase and use the same phrase in every proxy authentication.

Table 21: Security Comparison Table of Proxy Based Systems

	Proxy Based Systems	Imposter [4]	Wu et al. [5]	Delegate [6]	KLASSP [7]	URRSA [8]	SecurePass [9]	Trust-in-the-Middle
Client-Side Threats	T1:Keystroke Logging	*	*	*	*	*	*	*
	T2:Screen-Scraping	*	*	*	*	*	*	*
	T3: Malicious Codes	+	*	*	+	+	+	+
	T4: Malicious Browser	+	*	*	+	+	+	*
	T5: Phishing	*	*	*	*	*	*	*
	T6:Transaction Generators	-	-	*	-	-	-	-
	T7: Physical Observation	+	*	*	+	*	*	*
	T8: Password Attacks	-	*	*	*	*	*	*
	T9: Lost or Stolen Physical Object	*	+	+	-	-	+	*
Network Threats	T10: Interception Attacks	*	*	*	*	*	*	*
	T11: Passive SSL MITM Attacks	*	*	*	*	*	*	*
	T12: Active and Real-time SSL MITM Attacks	-	+	+	-	-	-	*
	T13: Pharming	+	+	+	+	*	+	*
Proxy Threats	T14: Threats against proxy services	-	-	-	-	-	-	*
	T15: Threats against User Database	-	-	-	-	-	-	*
	T16: Malicious Modification of Proxy Software	-	-	-	-	-	-	*
	T17: Run-time and Memory Based Attacks	-	-	-	-	-	-	-
Verifier Threats	T18: Leaks from Other Verifiers	-	-	-	-	-	-	-
	T19: Collusion Attacks	*	*	*	*	*	*	*

* Resilient to the threat

+ Quasi-Resilient to the threat

- Non-Resilient to the threat

Malicious codes on client computer can mount both passive eavesdropping attacks and real time attacks that can perform both eavesdropping and denial-of-service attacks at the same time. In real time attacks, malicious code can perform denial of service attack just after capturing user's one-time-password. By this way, the OTP can still be valid, if it is used on time. So the systems requiring only one-time password input from the client computer for authentication cannot provide protection to real time attacks although they are resilient to passive eavesdropping attacks of malicious codes. For this reason, we grant Imposter, KLASSP, URRSA, SecurePass and Trust-in-the-Middle Quasi-Resilient-to-Malicious-Codes. Since Wu et al. and Delegate implements a second channel verification, they are Resilient-to-Malicious-Codes. The same attacks can also be performed by a malicious browser. For this reason, we give almost the same rates to the systems. Only the rate of the Trust-in-the-Middle is different because one-time passwords are used in SSH authentication not in browsers.

According to Bonneau et al. UDS (Usability-Deployability-Security) framework [99] detailed in the next section, the schemas that can be broken only by repeating the observation more than 10-20 times are granted as Quasi-Resilient-to-Physical-Observation. If we follow up the same evaluation method, we can say that Imposter and KLASSP are Quasi-Resilient-to-Physical-Observation as the entered secrets might be revealed after observing the user's password input more than 10-20 times. Since the other schemas do not require users to enter their original passwords, they are all Resilient-to-Physical-Observation.

In Imposter, users are not required to carry a physical object and in Trust-in-the-Middle user's mobile phone and the OTP application in it are PIN protected. For these reasons, both schemas are secure against lost or stolen physical objects. In

KLASSP, mapping table can be stolen. In URRSA, the paper including one time passwords can be stolen. Therefore; both schemas are vulnerable to lost or stolen physical objects. Since the mobile phones are PIN protected but there is not any PIN protection in accessing the OTPs in mobile phones, the other schemas are granted as Quasi-Resilient-to-Lost-or-Stolen-Physical objects.

In terms of Client-side threats, Delegate has the best figures by providing protection almost all client side threats. Wu et al. and Trust-in-the-Middle share the second place.

When we evaluate each system in terms of network threats, we see that all the systems are Resilient-to-Interception-Attacks and Resilient-to-Passive-SSL-MITM-Attacks because of the one-time password usage. The systems implementing second channel in proxy authentication, Wu et al. and Delegate, are Quasi-Resilient-to-Active-and-Real-Time-SSL-MITM-Attacks as they do not provide protection for the network link between proxy and target server although they provide protection for the network link between client and the proxy. Since both of the links are secured by Trust-in-the-Middle, it is Resilient-to-Active-and-Real-Time-SSL-MITM-Attacks. The other schemas are vulnerable to this attack.

When we evaluate each systems in terms of pharming. We can say that all the systems except Trust-in-the-Middle and URRSA are Quasi-Resilient-to-Pharming. Since user's traffic is directed to a forged site by DNS manipulation, both passive pharming which just captures the user's credentials for later use and active pharming which both captures and use the credential in real-time without permitting user to log into the system are possible. Because those schemas only provide protection to passive pharming, we rate them as Quasi-Resilient-to-Pharming. We also assume that users ignore SSL warning messages. In Trust-in-the-Middle the user's connected

server is verified during SSH establishment and the traffic is encrypted through the SSH tunnel. In URRSA, the original URL of target site is not entered into address bar of the browser. Instead, it is entered in URRSA. So DNS manipulation attacks for target sites are not effective for URRSA. Therefore; we say that these two systems are Resilient-to-Pharming.

When we compare the systems against network threats, we see that best performance belongs to Trust-in-the-Middle.

As we have mentioned in Section IV, Trust-in-the-Middle provides protection all proxy threats except run-time and memory based attacks. All the other proxy systems do not offer any protection for third party proxy systems and just assume them as trusted. This is where Trust-in-the-Middle makes the main contribution.

When we evaluate the schemas according to verifier threats, we see that all the schemas are vulnerable to leaks from other verifiers as they do not make any change on the user's original credentials on target servers and all the systems are secure against collusion attacks as the schemas do not use a common ID or any other common information for users on target sites.

IV.2 Usability-Deployability-Security Comparison

In this section, we compare Trust-in-the-Middle with 20 previous authentication schemas including proxy based systems, TPM based systems and password managers from usability, deployability and security aspects. We use a slightly extended version of UDS (Usability, deployability, security) framework of Bonneau et al. [99]. Before going into details of our comparison table, we first introduce this framework below:

IV.2.1 Usability-Deployability-Security Framework

UDS framework of Bonneau et al. [99] defines 25 baseline properties in user authentication. Each schema is evaluated as either offering or not offering the property. If a schema almost offers the property, but not quite, it is indicated with "Quasi-" prefix. We have slightly extended the original framework by adding two more properties to deployability, "D7: Protocol Compatible" and "D8: Client Architecture Compatible" and one more property to security "S12: Resilient to SSL Man-in-the-Middle". So the total baseline properties is increased to 28.

We, now, give each baseline property below and define them:

IV.2.1.1 Usability Properties

U1 Memorywise-Effortless: If user is not required to memorize passwords at all, we grant Memorywise-Effortless. If user memorizes just one password for every service, we grant a Quasi-Memorywise-Effortless.

U2 Scalable-for-Users: If it does not create extra burden on user to connect just one service or a hundred services, we grant Scalable-for-Users. This property is evaluated from user's point of view, not system's point of view.

U3 Nothing-to-Carry: If users are not required to carry any additional physical object such as piece of paper, electronic or mechanical equipment, we grant Nothing-to-Carry. Quasi-Nothing-to-Carry is awarded, if the carried object is the one that user carries everywhere such as mobile phones but not computer or tablet.

U4 Physically-Effortless: If users are not required to do a physical action (not cognitive) such as typing, scribbling or performing a set of motions, we grant Physically-Effortless. Quasi-Physically-Effortless is awarded if the effort is limited to speaking or if user enters just only one password for accessing all the services.

U5 Easy-to-Learn: If users, who do not know the schema before, can easily learn and use the schema without so many problems, we grant Easy-to-Learn.

U6 Efficient-to-Use: If the time needed for each authentication is acceptably short, we grant Efficient-to-Use. The time required for setting up a new association with a verifier as it is in web based single sign-on systems is also accepted.

U7 Infrequent-Errors: If ordinary users do not have frequent problems while trying to authenticate themselves to the schema, we grant Infrequent-Errors.

U8 Easy-Recovery-from-Loss: If the system gives the ability of recovery of credentials without urging user physically standing in line or creating too much latency when users lose or forget their credentials, we grant Easy-Recovery-from-Loss.

IV.2.1.2 Deployability Properties

D1 Accessible: If users who can use passwords are not prevented to use the schema by disabilities or any other physical (not cognitive) conditions, we grant Accessible. If user is required to read password from somewhere, make a comparison or perform numerous actions in order to determine the password, we do not grant Accessible.

D2 Negligible-Cost-per-User: If the total cost does not increase on both user's side and the verifier's side when a new user is added to the schema, we grant Negligible-Cost-per-User.

D3 Server-Compatible: If the schema does not insist any change on verifiers' existing authentication process, we grant Server-Compatible.

D4 Browser Compatible: If the schema does not require special software or plugin on users' up-to-date and standards compliant browsers and can be executed in popular web browsers with their existing configuration, we grant Browser Compatible. A Quasi-Browser-Compatible is awarded if non-standard but very common plugins, e.g., Flash or some special settings e.g., proxy settings are needed.

D5 Mature: If the schema has been implemented and is being used on a large scale, we grant Mature.

D6 Non-Proprietary: If anyone can implement or use the schema for any purposes without having to pay royalties to anyone else, we grant Non-Proprietary.

D7 Protocol Compatible: If the schema does not insist any change on the available protocols, we grant Protocol Compatible.

D8 Client Architecture Compatible: If the schema works with commodity workstations and does not require specific architecture on client side, we grant Client-Architecture-Compatible.

IV.2.1.3 Security Properties

S1 Resilient-to-Physical-Observation: If the user's password cannot be obtained by physical observation (i.e. shoulder surfing, recording keystroke sounds, thermal imaging of keypad and etc.) during authentication, we grant Resilient-to-Physical-Observation. If it can be obtained after 10-20 physical observation, we grant Quasi-Resilient-to-Physical-Observation.

S2 Resilient-to-Targeted-Impersonation: If it is not possible for a skilled adversary to impersonate the user by exploiting the knowledge of his personal information such as birth day, the names of children etc., we grant Resilient-to-Targeted-Impersonation. For the schemas using an intermediary system such as proxy, we evaluate this property according to the used proxy authentication password differently from Bonneau et al. framework [99].

S3 Resilient-to-Throttled-Guessing: If the adversary, whose rate of prediction is limited by the verifier, cannot successfully predict the passwords of a significant fraction of users, we grant Resilient-to-Throttled-Guessing. For example if the adversary, who is limited 10 predictions per account per day, can find at most 1% of passwords in a year, this property is granted. For the schemas using an intermediary system such as proxy, we evaluate this property according to the used proxy authentication password differently from Bonneau et al. framework [99].

S4 Resilient-to-Unthrottled-Guessing: If the adversary, whose rate of prediction is limited only by the available computing resources, cannot successfully predict the passwords of a significant fraction of users, we grant Resilient-to-Unthrottled-Guessing. For example if the adversary, who can make up to 2^{64} attempts per account, can still only obtain 1% of passwords, this property is granted. For the schemas using

an intermediary system such as proxy, we evaluate this property according to the used proxy authentication password differently from Bonneau et al. framework [99].

S5 Resilient-to-Internal-Observation: If an adversary cannot impersonate the user by intercepting his authentication session from inside his device (i.e. keystroke logging) or the communication line between the prover and the verifier, we grant Resilient-to-Internal-Observation. A Quasi-Resilient-to-Internal-Observation is awarded, if the adversary can obtain the user's credentials after intercepting the input or eavesdropping the communication link more than 20 times. We also grant Quasi-Resilient-to-Internal-Observation for the schemas incorporating two factor authentication where both factors should be compromised for the attack to work.

S6 Resilient-to-Leaks-from-Other-Verifiers: If the information obtained from a malicious verifier cannot be used by an adversary to impersonate the user, we grant Resilient-to-Leaks-from-Other-Verifiers.

S7 Resilient-to-Phishing: If an adversary cannot pretend a verifier to obtain the secret credentials of the user, we grant Resilient-to-Phishing. This attack incorporates both obtaining the credentials from a lookalike site of the verifier and performing DNS manipulation attacks.

S8 Resilient-to-Theft: If an adversary obtaining the physical device used in authentication cannot impersonate the user, we grant Resilient-to-Theft. If the device is PIN protected, we grant Quasi-Resilient-to-Theft.

S9 No-Trusted-Third-Party: If the schema does not rely on a third party which can reveal user's secret credentials or violates his privacy if compromised, we grant No-Trusted-Third-Party.

S10 Requiring-Explicit-Consent: If the authentication process cannot start without the explicit consent of user, we grant Requiring-Explicit-Consent.

S11 Unlinkable: if colluding verifiers cannot determine, from the authenticator alone, whether the same user is authenticated to both systems, we grant Unlinkable.

S12 Resilient to SSL Man-in-the-Middle: There are two types of SSL Man-in-the-Middle attacks; Passive SSL Man-in-the-Middle Attacks and Active and Real-Time SSL Man-in-the-Middle Attacks (See Section IV.1). If the schema provides protection for both of them, it is awarded Resilient-to-SSL-Man-in-the-Middle. If the schema is secure against only one of them, it is awarded Quasi-Resilient-to-SSL-Man-in-the-Middle.

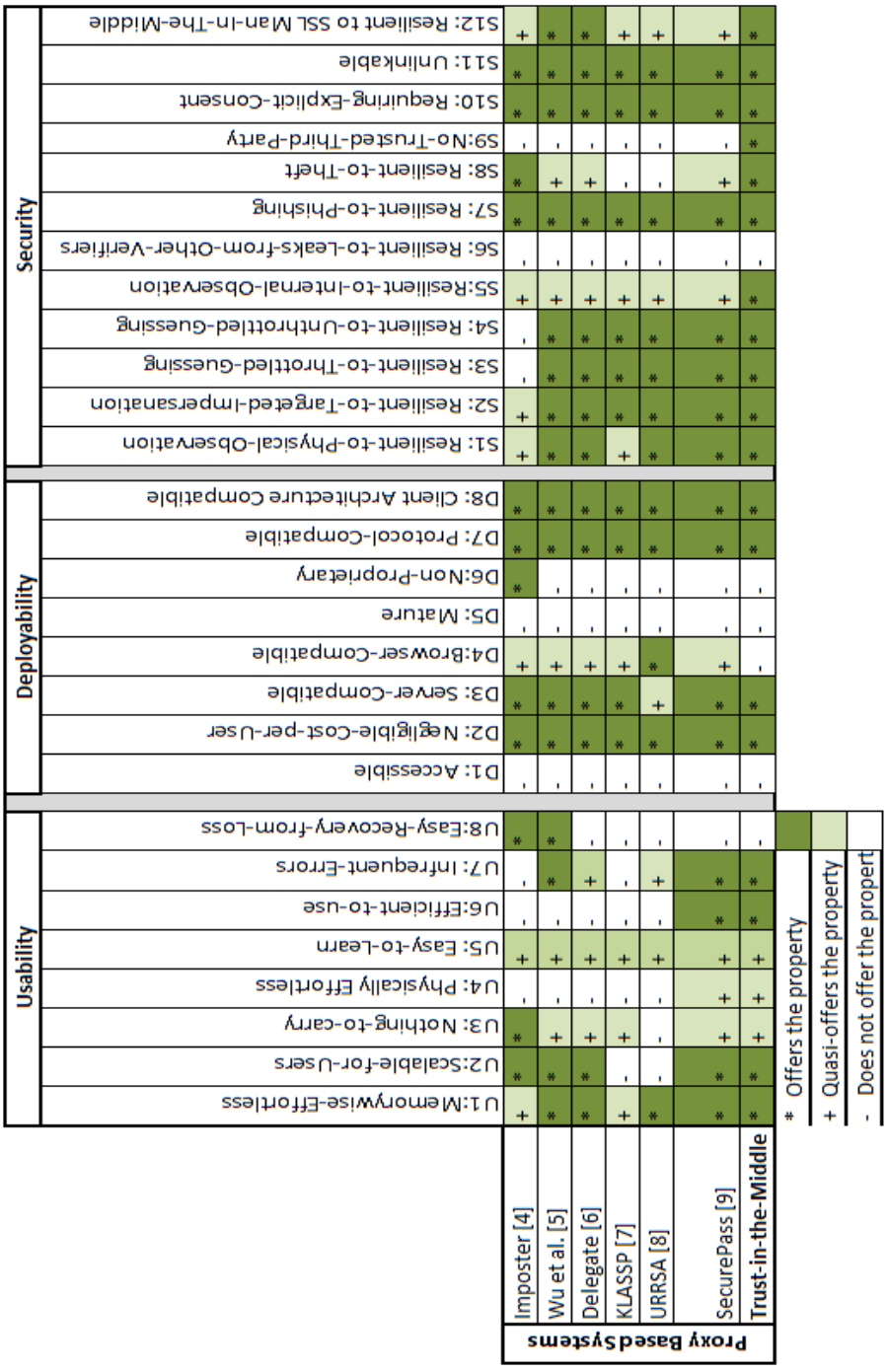
IV.2.2 Comparison of Proxy Based Systems

The results of our UDS analysis of proxy based systems are given in Table 22.

IV.2.2.1 Usability Evaluation of Proxy Based Systems

In Imposter, the users should memorize a secret phrase that they share with the proxy server in order to be able to use it for every proxy authentication. Likewise, in KLASSP users should memorize special characters in order to correctly enter the password. For these reasons, both of these schemas are granted Quasi-Memorywise-Effortless. All the other schemas including Trust-in-the-Middle are Memorywise-Effortless as they use one-time passwords that they check from a device or piece-of-paper and hence do not require memorizing the passwords.

Table 22: Usability-Deployability-Security Comparison of Proxy Based Systems



URRSA is not Scalable-for-Users, because users should require new one time passwords as the number of accounts is increased. In KLASSP, mapping table can be exhausted or the special character can be revealed as the number of authentications increases. So it is also not Scalable-for-Users. All the other schemas are Scalable-for-Users, as they do not require change on the behavior of users according to the increase in the number of accounts.

Only the Imposter is Nothing-to-Carry, all the other schemas except URRSA require users to carry mobile phones. So they are Quasi-Nothing-to-Carry. If we assume that URRSA one-time passwords are carried in a piece-of-paper as it is in [99], we say that it is not Nothing-to-Carry. SecurePass and Trust-in-the-Middle require users to enter OTP in the time of proxy authentication only and no more password is needed for the other authentication sessions as long as the user's initial session with proxy continues. For this reason, both of these schemas are Quasi-Physically-Effortless and Efficient-to-Use. All the other schemas are non-Physically-Effortless and non-Efficient-to-Use as they require users to enter a credential at each authentication session.

Since all the proxy based systems require prior setup, we grant all the schemas Quasi-Easy-Learn. Because of the fact that Imposter and KLASSP require users to take a cognitive action to determine the password at each authentication, they are not Infrequent-Errors. Wu et al., SecurePass and Trust-in-the-Middle are awarded Infrequent-Errors, as they require less credential input during the operation. Delegate and URRSA are Quasi-Infrequent-Errors as they do not need to spend so much cognitive effort as it is in Imposter and KLASSP but they require users to enter something at each login.

Only Imposter and Wu et al. are Easy-Recovery-from-Loss because they require very simple processes in order to reactivate their credentials. The others are awarded non-Easy-Recovery-from-Loss, as they need more complex processes such as building a new OTP schema.

Figure 11: Usability Scores of Proxy Based Systems

In Figure 11, the blue bubbles illustrate usability scores of proxy based systems and orange bubbles illustrate the ranking of the schema according to the scores of all the schemas. Scores are determined by summing up the score of schemas which offer the property (rated as "1"), quasi-offer the property (rated as "0,5") or does not offer the property (rated as "0"). Although the weight of each property is also important when making a comparison as it is stated in [99], there is not an accepted and objective previous framework which takes into account the weights of each property. Therefore we do not include the weights of the properties in our current figures. All the other figures in following sections are created by using the same technique.

When we compare Trust-in-the-Middle with the other proxy based systems in terms of usability properties in Figure 11, we see that usability of the Trust-in-the-Middle and SecurePass have the best scores.

IV.2.2.2 Deployability Evaluation of Proxy Based Systems

None of the proxy based systems including Trust-in-the-Middle provide Accessibility property, as all the systems do not give possibility to a physically disabled person (i.e. blind person) to use the system easily. Most of them require one time password check and entering it into the proxy.

Since all the systems are proxy based systems and not used for commercial purposes, the properties Negligible-Cost-per-User and Server-Compatible hold for almost every schema. Only URRSA does not hold Server-Compatible property. Because it relies on a link-translating proxy that intermediates traffic between the user and the server, which means some functionality may fail on complex sites and server side modification might be required [99].

Although most of the proxy based systems quasi-offer or offer the Browser-Compatible property, Trust-in-the-Middle does not offer it, as it requires a browser plug-in. The reason why most of the systems are Quasi-Browser-Compliant is that they require proxy settings in users' browsers.

None of the systems are Mature. Because they are not widely deployed in the market. Since only the Imposter is publicly available, the systems other than the Imposter do not hold Non-Proprietary property.

All the proxy based systems do not require change in the available protocols and not insist a specific architecture on client side. So they hold Protocol-Compatible and Client-Architecture Compatible properties.

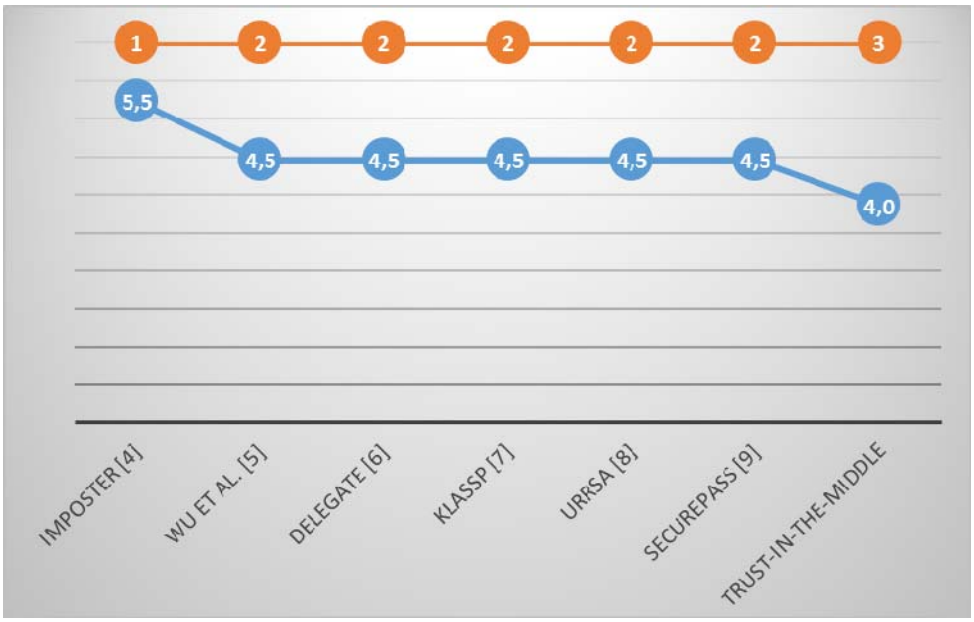


Figure 12: Deployability Scores of Proxy Based Systems

When we compare the deployability scores of the proxy based systems in Figure 12, we see that Imposter has the best deployability score and the Trust-in-the-Middle has the worst deployability score although the scores are very close to each other.

IV.2.2.3 Security Evaluation of Proxy Based Systems

Other than Imposter and Delegate, all the schemas including Trust-in-the-Middle are Resilient-to-Physical-Observation. Because most of them use one-time passwords

which cannot be used second time although they are physically observed. In Imposter, an adversary having physically observed the user's input more than 20 times, may build the original secret phrase. In KLASSP, if the special character, which points out that the original character of the password will be entered next, is detected after having physically observed user's input more than 20 times, there is a possibility to recover the password. So we grant Imposter and KLASSP Quasi-Resilient-to-Physical-Observation.

All the systems except the Imposter, are Resilient-to-Targeted-Impersonation, Resilient-to-Throttled-Impersonation and Resilient-to-Unthrottled-Impersonation. All these three properties are valid for the standard passwords which can be exposed to dictionary attacks, brute force attacks or guessing according to a personal information. Since most of the schemas utilize one time passwords which cannot be used second time even if they are guessed somehow, we say that all three properties are satisfied by those systems. On the other hand, Imposter uses a secret phrase determined by the user and shared with the proxy. Although only a few characters of the secret phrase are being used at each time user authenticates himself to proxy, all three attacks are valid for this secret phrase.

In order to satisfy Resilient-to-Internal-Observation, an adversary should not obtain user's credentials by performing malicious interception attacks from inside the user's device, or eavesdropping the communication line between the proxy and the target server. Although most of the schemas implement different techniques on the user's device or the communication link between user's device and the proxy, most of them send the original passwords through the communication line between themselves and the target servers. This means that a man-in-the-middle attack can easily capture the original passwords of the users. So for these reasons, we grant Quasi-Resilient-to-Internal-Observation to the schemas other than the Trust-in-the-Middle. Since Trust-

in-the-Middle uses SSL connection with the target site and checks the SSL certificate before establishing connection, mounting a man-in-the-middle attack between proxy and the target server is not possible. Therefore, Trust-in-the-Middle is granted Resilient-to-Internal-Observation.

Since, original passwords are submitted by the proxy system to the target server, we can say that there is a possibility that a malicious verifier can reveal user's secret credentials. For this reason, none of the systems are Resilient-to-Leaks-from-other-Verifiers. On the other hand, since original passwords are not entered by the users, all the systems are Resilient-to-phishing attacks.

Since Imposter system is Nothing-to-Carry, it is also Resilient-to-Theft. Trust-in-the-Middle uses mobile phones in one time password generation. However; an adversary, having stolen the mobile phone, is required to pass the PIN protection of both the mobile phone and the OTP application. So we can say that Trust-in-the-Middle is also Resilient-to-Theft. In KLASSP and URRSA, mapping table and the one time passwords which are told to be in a piece of paper can easily be obtained when the paper is stolen. For the other systems, it is not obviously stated whether the one time passwords are stored in a PIN protected device or not. So by assuming that they are under a moderate PIN protection, we grant them Quasi-Resilient-to-Theft.

Since different credentials are being used at the target servers and it is not possible to complete an authentication without explicit consent of the user, we say that all the schemas are Requiring-Explicit-Consent and Unlinkable.

Wu et al. and Delegate implement a second channel where they can verify the connected servers. So they are not vulnerable to SSL Man-in-the-Middle attacks. Since Trust-in-the-Middle establishes an SSH tunnel and encapsulates all the traffic

through this tunnel, we can say that it is also Resilient-to-SSL-Man-in-the-Middle. On the other hand the other schemas are granted Quasi-Resilient-to-SSL-Man-in-the-Middle. Because they are not resilient to active and real time SSL Man-in-the-Middle attacks although they are resilient to passive SSL Man-in-the-Middle attacks.

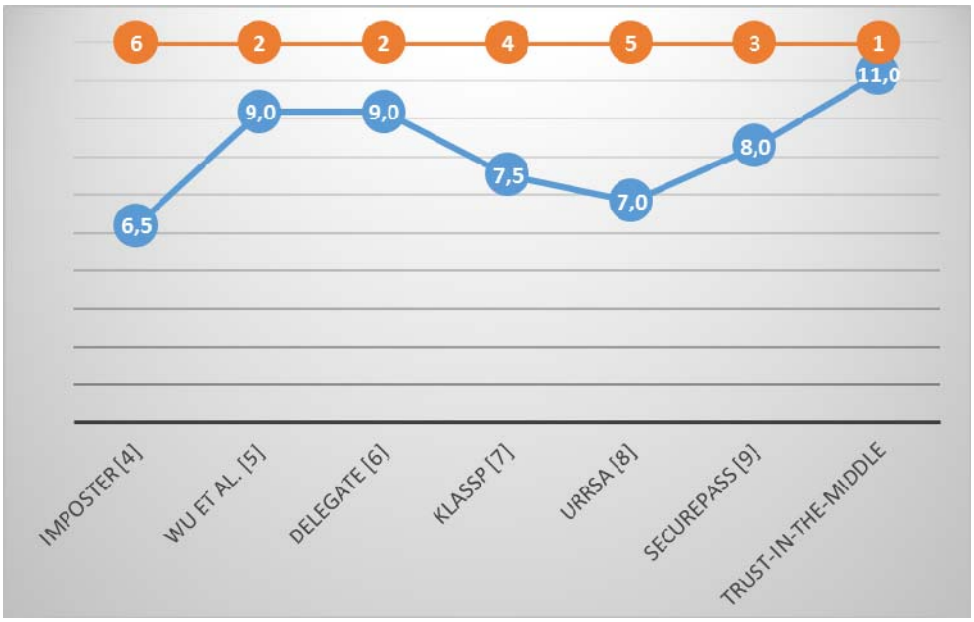


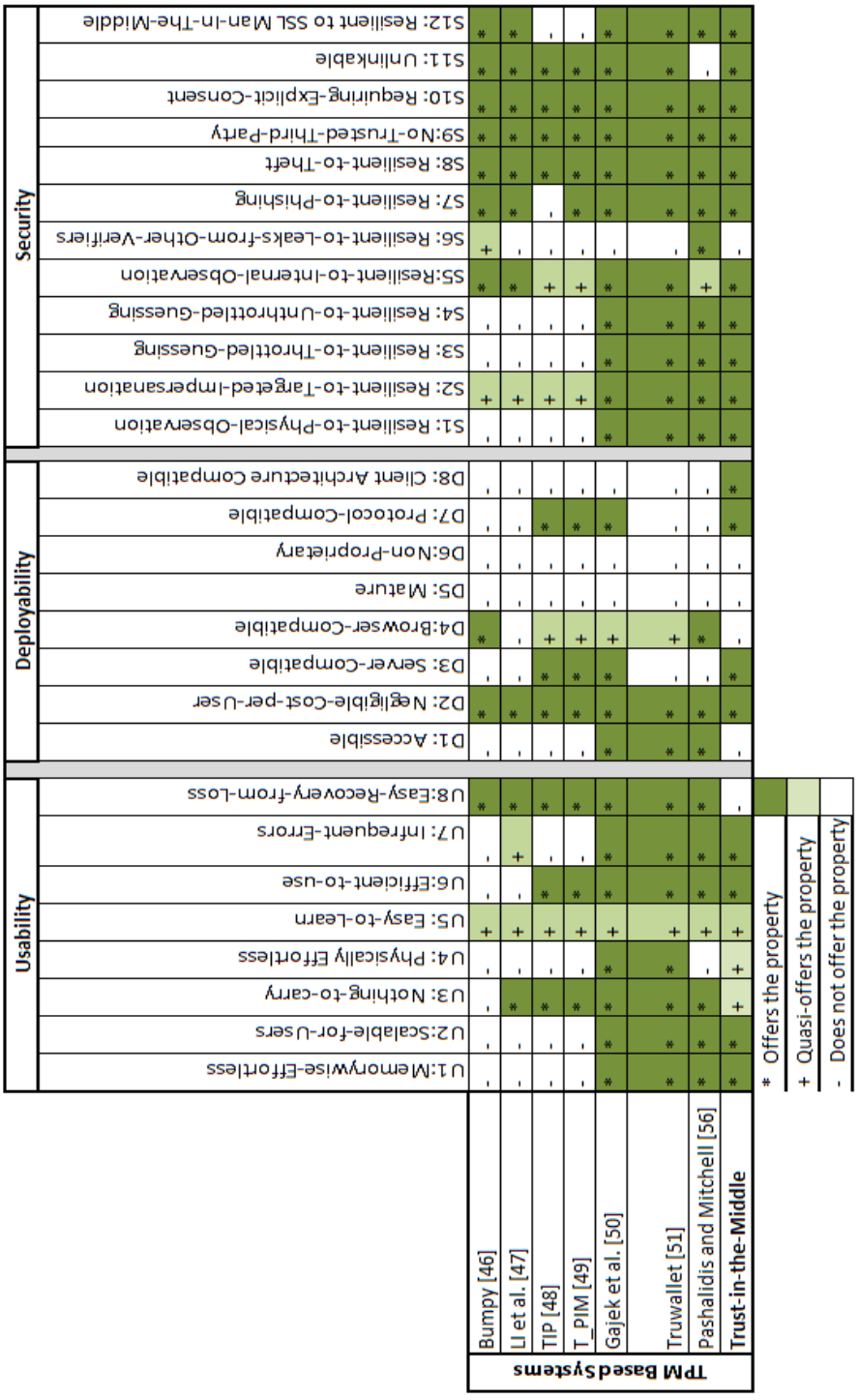
Figure 13: Security Scores of Proxy Based Systems

When we compare all the proxy based systems in terms of security properties in Figure 13, we see that Trust-in-the-Middle system has the best score and Imposter has the worst score.

IV.2.3 Comparison of TPM Based Systems

The results of our UDS analysis of TPM based systems are given in Table 23.

Table 23: Usability-Deployability-Security Comparison of TPM Based Systems



IV.2.3.1 Usability Evaluation of TPM Based Systems

When we look at the previous work which are based on TPM, Bumpy, Li et al., TIP and T_PIM provide users more protected environments to enter their existing passwords. Since they do not make any change in users' standard password input behavior, all these schemas are not Memorywise-Effortless and not Scalable-for-Users. On the other side, in the other schemas including Trust-in-the-Middle, users either store their passwords or use their TPM based identities as authentication credentials, which do not require users to memorize passwords. Therefore, we say that the other schemas are Memorywise-Effortless and also Scalable-for-Users.

Since Bumpy requires an encrypting keyboard and a mobile phone as Trusted Monitor. It is not Nothing-to-carry. As we mentioned previously Trust-in-the-Middle is Quasi-Nothing-to-Carry because of the mobile phone usage. All the other schemas are Nothing-to-carry.

Bumpy, Li et al., TIP and T_PIM are the schemas which are not Physically Effortless as users still enter their existing passwords. In Pashalidis and Mitchell, users should verify a special ID from the server. So it is also not Physically-Effortless. In wallet based systems, Gajek et al. and Truwallet, users' passwords are automatically entered by the wallet systems, which makes them Physically-Effortless. Since an OTP input is needed for just initial authentication, Trust-in-the-Middle is Quasi-Physically Effortless.

All of the schemas are Quasi-Easy-to-Learn, as they need prior setup in order to make the system work. During password input, Bumpy and Li et al. execute TPM DRTM

environment and make encryption. For this reason we do not grant them Efficient-to-Use. All the other schemas are Efficient-to-Use.

In order to change into secure input environment, users should type special characters in Bumpy, TIP and T_PIM and also enter their standard passwords in these environments. Therefore, they are not Infrequent-Errors. In Li et al. the secure environment is triggered by the browser. However; standard passwords should again be entered. For this reason, we grant Quasi-Infrequent-Errors. The other schemas either do not require password input or use one-time-password just for the initial authentication. So we can say that they are Infrequent-Errors.

All the schemas are Easy-to-Recover except Trust-in-the-Middle which requires re-initialization of one time password schema, if the current one is lost.

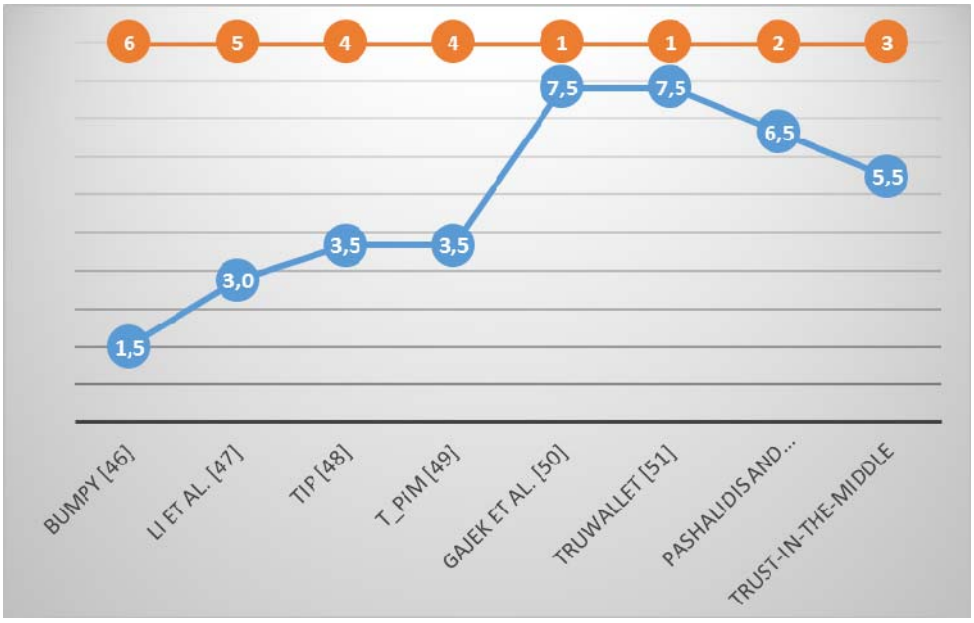


Figure 14: Usability Scores of TPM Based Systems

When we compare all those TPM based systems in terms of usability in Figure 14, we see that Bumpy has the worst, Truwallet and Gajek et al. has the best usability scores. Trust-in-the-Middle is in 4th place in terms of usability among 8 TPM based systems.

IV.2.3.2 Deployability Evaluation of TPM Based Systems

Wallet based systems, Truwallet and Gajek et al. and Single Sign-on system of Pashalidis and Mitchell are Accessible. Because, once registration and pre-settings are done, systems do not require any complex action during authentication. On the other hand, the other schemas including Trust-in-the-Middle do require more complex actions than just password input such as checking one time password, typing special characters to initiate TPM DRTM and so on. For this reason, all the other schemas are not Accessible.

Since all the systems are not used for commercial purposes, the properties Negligible-Cost-per-User hold for every schema. Bumpy, Li et al. and Truwallet use shared secret with the servers and Single Sign-on system of Pashalidis and Mitchell executes an authentication algorithm with the server. For these reasons, all these systems do not hold Server-Compatible property. The other systems including Trust-in-the-Middle do not require any change on server side, thus they are Server-Compatible.

The systems utilizing proxy, TIP, T_PIM, Gajek et al. and Truwallet require to configure proxy settings. Therefore, they are Quasi-Browser-Compatible. Bumpy and Pashalidis and Mitchell are Browser-Compatible as they do not insist any change on browsers. The other systems including Trust-in-the-Middle are not Browser-Compatible.

None-of the systems are Mature and Non-Proprietary. Because they are not widely deployed in the market and the systems are not available in the internet. Bumpy, Li et al. Truwallet and Pashalidis and Mitchell require change on the operation of standard authentication protocols. So they are not Protocol-Compatible. The other systems are Protocol-Compatible. All the systems except Trust-in-the-Middle require TPM on client side. For this reason, all these systems are not Client-Architecture-Compatible.

When we compare all those TPM based systems in terms of deployability in Figure 15, we see that Li et al. has the worst, Gajek et al. has the best scores. When we look at Trust-in-the-Middle, we see that it is in the 2nd best system among 8 TPM based systems in terms of deployability.

Figure 15: Deployability Scores of TPM Based Systems

IV.2.3.3 Security Evaluation of TPM Based Systems

The systems, which still require users to enter their original passwords such as Bumpy, Li et al. TIP and T_PIM, are not Resilient-to-Physical-Observation as an adversary performing a shoulder surfing attack can observe and learn the original password. The other systems are Resilient-to-Physical-Observation because the original passwords are not entered by the user.

Since original passwords are still used at each authentication in Bumpy, Li et al. TIP and T_PIM, standard password vulnerabilities, which are Quasi-Resilient-to-Targeted-Impersonation, Non-Resilient-to-Throttled-Guessing and Non-Resilient-to-Unthrottled-Guessing, are valid for those schemas. The other schemas are not vulnerable to these attacks, so they are Resilient-to-Targeted-Impersonation, Resilient-to-Throttled-Guessing and Resilient-to-Unthrottled-Guessing.

Since password input is carried out in TPM DRTM isolation and the passwords are sent to server in an encrypted fashion, Bumpy and Li et al. are Resilient-to-Internal-Observation. Although TIP and T_PIM provide protection against internal observation in local computer, an adversary mounting SSL Man-in-the-Middle can capture the passwords. So these two schemas are Quasi-Resilient-to-Internal-Observation. In the work of Pashalidis and Mitchell, network based observation is not possible as a special protocol is executed between the client and the server. However; local authentication poses internal observation risks. Since Pashalidis and Mitchell declared that physical authentication schemas can also be implemented besides username and password authentication. We gave Quasi-Resilient-to-Internal-Observation. Wallet based authentication schemas, Truwallet and Gajek et al. and Trust-in-the-Middle are Resilient-to-Internal-Observation as they do not require

password input on client side and establish mechanism to secure the communication line between the system and the server.

Bumpy proposes two authentication schemes; one of them is encrypting the existing password with a secret key shared between Bumpy and the server. The other is hashing the password with the domain name of the server. In the second one, if password update is carried out by the user for all the servers, leakage of the passwords will not make sense as they will be different for each service. For this reason, we grant Quasi-Resilient-to-Leaks-from-Other-Verifiers to Bumpy. The other schemas except Pashalidis and Mitchell are not Resilient-to-Leaks-from-Other-Verifiers as the passwords determined by the user are utilized in these systems and reuse of the same passwords are possible. Since the system of Pashalidis and Mitchell does not use a password based authentication, it is Resilient-to-Leaks-from-Other-Verifiers.

Regarding the fact that the connected server is authenticated, Bumpy, Li et al., Gajek et al., Truwallet, Pashalidis and Mitchell and Trust-in-the-Middle are Resilient-to-Phishing. T-PIM also shows connected server's IP Address and the domain name to the user, it can also be accepted as Resilient-to-Phishing. TIP is not Resilient-to-Phishing.

There is not a third party system utilization in all the TPM based schemas except Trust-in-the-Middle. Since Trust-in-the-Middle establishes the trustworthiness of the third part proxy system and does not make a trust assumption, we can say that all the schemas are granted No-Trusted-Third-Party.

Since client's TPM Identities are used as authentication identities, there is a possibility to establish a link between the authentication sessions in the system of

Pashalidis and Mitchell. So it is not Unlinkable. The other schemas do use standard password based authentication. So they are Unlinkable.

TIP and T_PIM are vulnerable to SSL Man-in-the-Middle attacks. So they are not Resilient-to-SSL-Man-in-the-Middle. Since the other schemas authenticate server before establishing connection by using several methods, they are Resilient-to-SSL-Man-in-the-Middle.

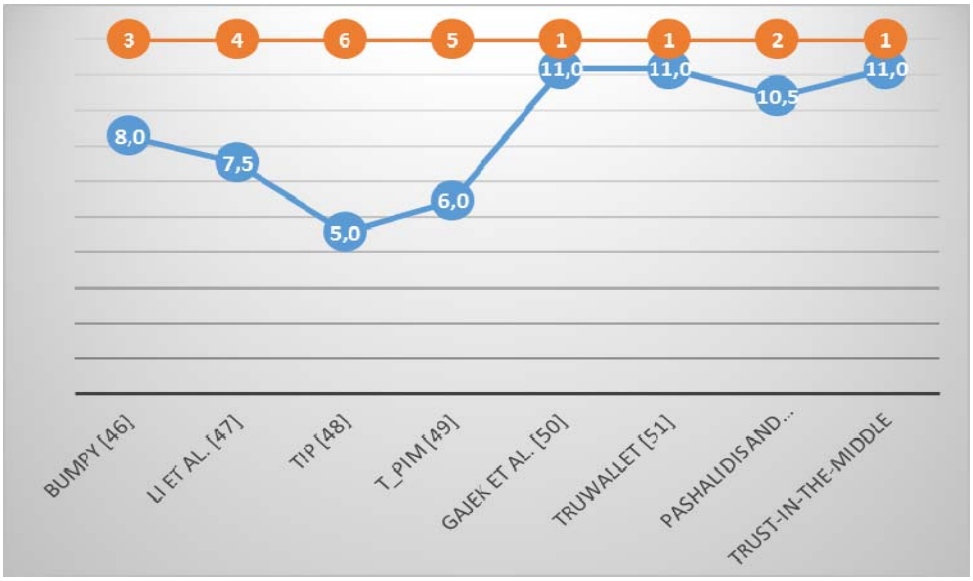


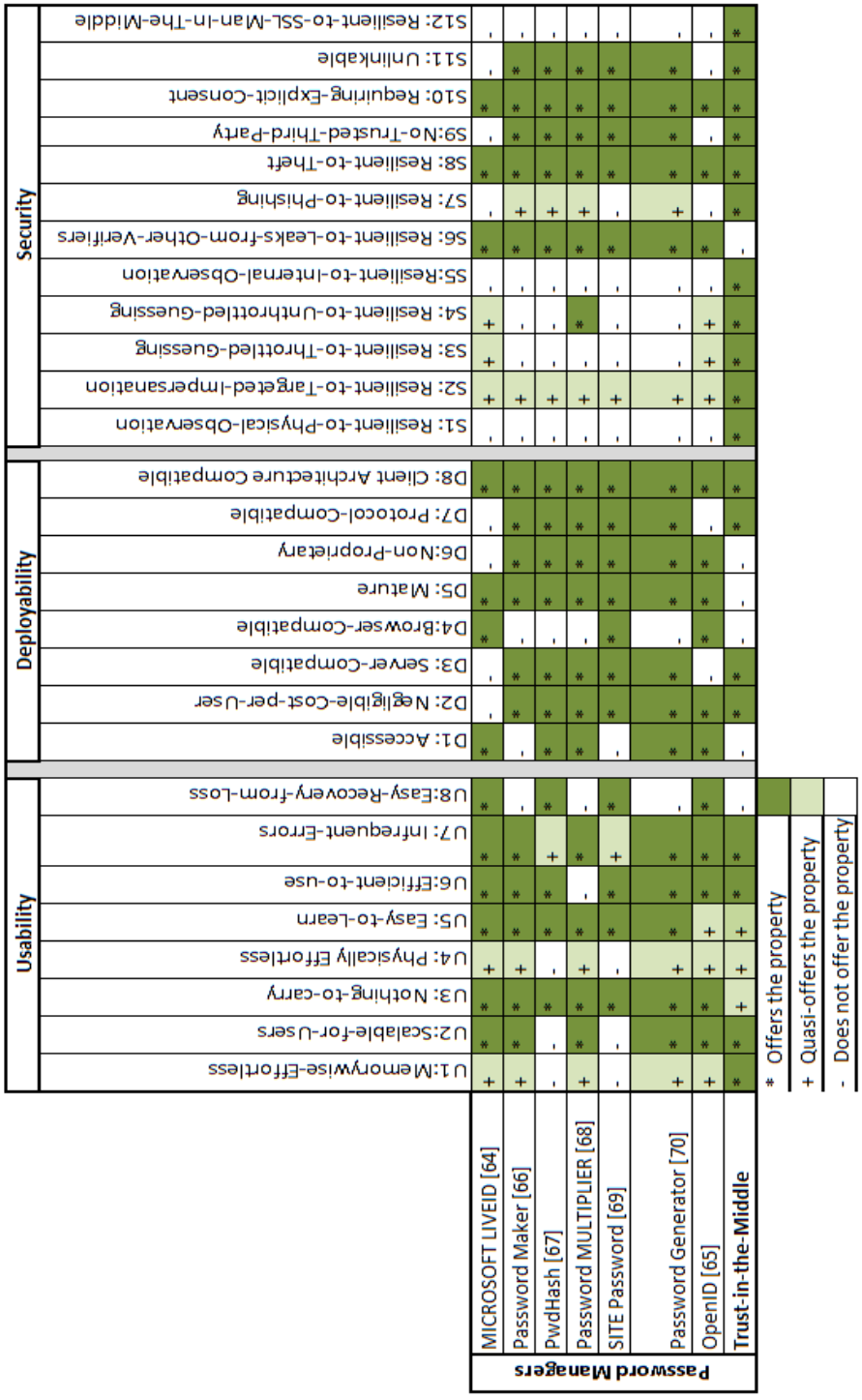
Figure 16: Security Scores of TPM Based Systems

When we compare all TPM based systems in terms of security in Figure 16, we see that TIP has worst scores and Gajek et al., Truwallet and the Trust-in-the-Middle have the best scores.

IV.2.4 Comparison of Password Managers

The results of our UDS analysis of Password Managers are given in Table 24.

Table 24: Usability-Deployability-Security Comparison of Password Managers



IV.2.4.1 Usability Evaluation of Password Managers

Since users should memorize only master password, Microsoft LiveID, Password Maker, Password Multiplier, Password Generator and OpenID are Memorywise-Effortless. Because of the fact that users should memorize a password or keyword for each authentication, PwdHash and Site Password schemas are not Memorywise-Effortless. On the other hand, since Trust-in-the-Middle implements one time password authentication, users are not required to memorize passwords. That is why we grant Trust-in-the-Middle Memorywise-Effortless.

The schemas, PwdHash and Site Password, which do not hold Memorywise-Effortless, do not also hold Scalable-for-Users property. The others are Scalable-for-Users. Among all password managers, the only schema that users should carry something with them is Trust-in-the-Middle. Since mobile phone, which users always carry with them, is used, Trust-in-the-Middle is Quasi-Nothing-to-Carry. The others are Nothing-to-carry.

The schemas which use master password during authentication or which require one-time password input only once at the initial authentication, are Quasi-Physically-Effortless and Infrequent-Errors. The other schemas, PwdHash and Site Password are not Physically Effortless and are Quasi-Infrequent-Errors because they require to input a password or a keyword at each authentication.

Ordinary users may have difficulty in using OpenID and Trust-in-the-Middle system for the first time. However; the other schemas can easily be learned and used. So we grant OpenID and Trust-in-the-Middle systems Quasi-Easy-to-Learn and we grant the other systems Easy-to-Learn. Furthermore; all the schemas except Password Multiplier are Efficient-to-Use. Since Password Multiplier implements some

mechanisms to slow down the hashing procedure, it does not hold Efficient-to-Use property.

The systems which run on client side and utilize the same master password to generate the original passwords for every target sites may create a huge burden on user to update his passwords in all the target sites when the master password is lost or forgotten. So we say that Password Maker, Password Multiplier and Password Generator are not Easy-Recovery-from-Loss. Trust-in-the-Middle is also not Easy-Recovery-from-Loss as it requires to build a new one-time password schema when the current one is lost. The other schemas can apply easy recovery procedures when the passwords are forgotten or lost. So they are Easy-Recovery-from-Loss.

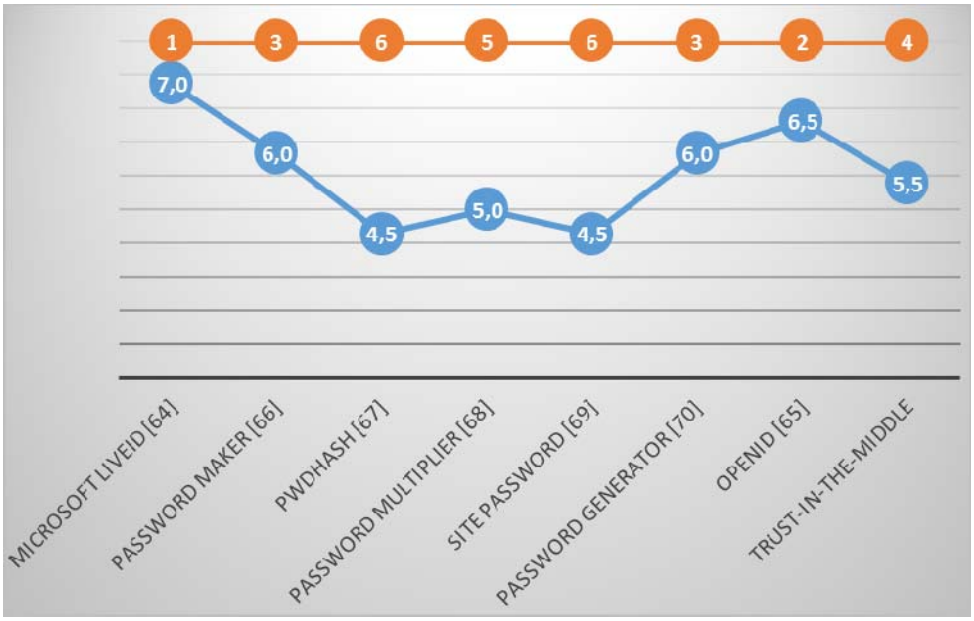


Figure 17: Usability Scores of Password Managers

When we compare all password managers in terms of usability in Figure 17, we see that Microsoft LiveID has the best scores. On the other hand PwdHash and

SitePassword schemas have the worst scores. When we look the performance of Trust-in-the-Middle, we can say that the usability scores of Trust-in-the-Middle is not promising when compared to the other systems. Trust-in-the-Middle is in the 5th order in terms of usability among 8 systems.

IV.2.4.2 Deployability Evaluation of Password Managers

In order to input password, Password Maker requires to execute browser add-on, then enter a master password twice, create the hash value, copy and paste it in the password field. The similar difficulties are also valid for Site Password schema. On the other hand, Trust-in-the-Middle requires to check one time password and enter it on the system during authentication. For these reasons, we say that all these three systems are not Accessible. The other schemas are very similar to standard password input. For this reason, they are awarded as Accessible.

Only the Microsoft LiveID is used commercial. For this reason, it is not Negligible-Cost-per-User and the others are Negligible-Cost-per-User. Since Microsoft LiveID and OpenID require change on server side and on the existing authentication protocols, they are not Server-Compatible and not Protocol-Compatible. The other schemas are Server-Compatible and Protocol-Compatible. Only Trust-in-the-Middle system is not accessible in the internet and widely deployed. So except Trust-in-the-Middle, we can say that all the other schemas are Mature.

Microsoft LiveID and Trust-in-the-Middle are not Non-Proprietary as they are not open systems. The other systems hold the Non-Proprietary property. Finally, we can say that all the password managers are Client-Architecture-Compatible as they do not require a special architecture on client side.

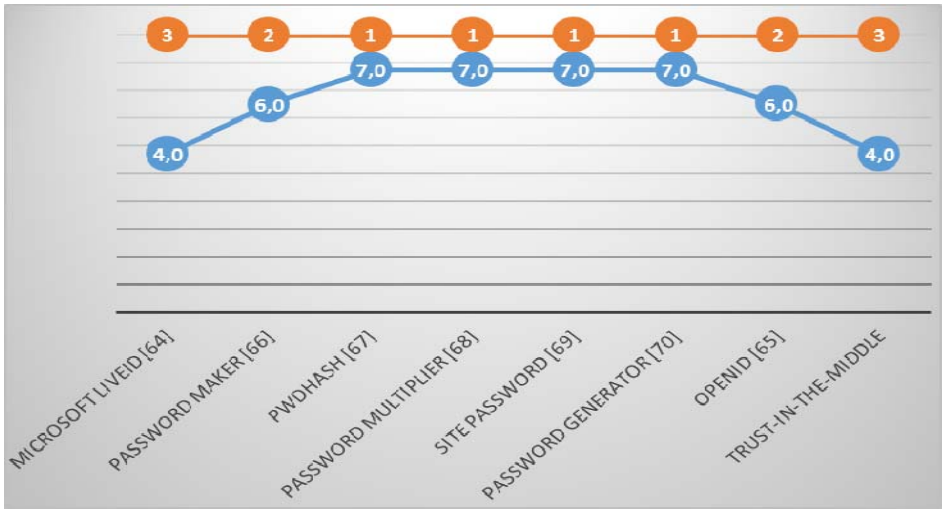


Figure 18: Deployability Scores of Password Managers

When we compare all the password managers in terms of deployability in Figure 18, we see that Trust-in-the-Middle and Microsoft LiveID have the worst scores. PwdHash, Password Multiplier, Site Password and Password Generator share the best score.

IV.2.4.3 Security Evaluation of Password Managers

Since most of the schemas use standard passwords, they are vulnerable to physical observation and targeted impersonation. Only Trust-in-the-Middle provides protection against physical observation and targeted impersonation as it implements one time passwords. So Trust-in-the-Middle is Resilient-to-Physical-Observation and Resilient-to-Targeted-Impersonation. The others are non-Resilient-to-Physical-Observation and Quasi-Resilient-to-Targeted-Impersonation. We rate Quasi-Resilient-to-Targeted-Impersonation because there is a risk that user may use

personal information in his password. But he may also chose a high-entropy password.

Microsoft LiveID and OpenID are Quasi-Resilient-to-Throttled-Guessing and Quasi-Resilient-to-Unthrottled-Guessing. Because, in these schemas, most of the attacks are limited against password authentication between client and his identity provider. Regarding the one-time-password usage, Trust-in-the-Middle is both Resilient-to-Throttled-Guessing and Resilient-to-Unthrottled-Guessing. Because almost all of the other schemas implement standard passwords, they are not Resilient-to-Throttled-Guessing and not Resilient-to-Unthrottled-Guessing. Only Password Multiplier is Resilient-to-Unthrottled-Guessing because of its mechanisms to slow down hash operations, which means that it reduces the risk of brute force attacks.

Since there is a password input on client side, all the systems except Trust-in-the-Middle are non-Resilient-to-Internal-Observation. Since standard passwords determined by the user are still conveyed to target server in Trust-in-the-Middle, it is non-Resilient-to-Leaks-from-Other-Verifiers. The other schemas either change existing passwords with hash values or implement a special authentication protocol using federated single sign-on systems such as Microsoft LiveID and OpenID, they are all Resilient-to-Leaks-from-Other-Verifiers.

Considering the fact that federated single sign on systems (OpenID, Microsoft LiveID) involve re-direction to an identity provider from a relying party, they are non-Resilient-Phishing. Site Password is also vulnerable to phishing attacks. Trust-in-the-Middle is Resilient-to-Phishing as it verifies the connected server's certificates. The other systems do not provide protection for pharming attacks which manipulate DNS configuration although they are resistant to standard phishing attacks. So we

rate Password Maker, PwdHash, Password Multiplier and Password Generator Quasi-Resilient-to-Phishing.

All of the systems are Resilient-to-Theft and Requiring-Explicit-Consent. Because of the fact that third party systems do exist in the framework of Microsoft LiveID and OpenID, they are not No-Trusted-Third-Party. The other systems except Trust-in-the-Middle do not use third parties. Therefore, they are awarded as No-Trusted-Third-Party. Trust-in-the-Middle establishes the trustworthiness of the third party proxy system and guarantees that users' sensitive credentials cannot be accessed even if the proxy system is compromised. So we also rate Trust-in-the-Middle as No-Trusted-Third-Party.

Federated single sign-on systems maintain users' identities and make users authenticate to various services based on their identities. So it might be possible to establish links between authentication sessions. For this reason, Microsoft LiveID and OpenID are rated as non-Unlinkable. The other systems are Unlinkable. And finally all the schemas except Trust-in-the-Middle are non-Resilient-to-SSL-Man-in-the-Middle as it is possible to intercept the connection. Trust-in-the-Middle is Resilient-to-SSL-Man-in-the-Middle because it tunnels SSL connection in SSH connection between client and proxy system and does not permit any man-in-the-middle attacks between proxy and target server by verifying the SSL certificate of the target server before establishing connection.

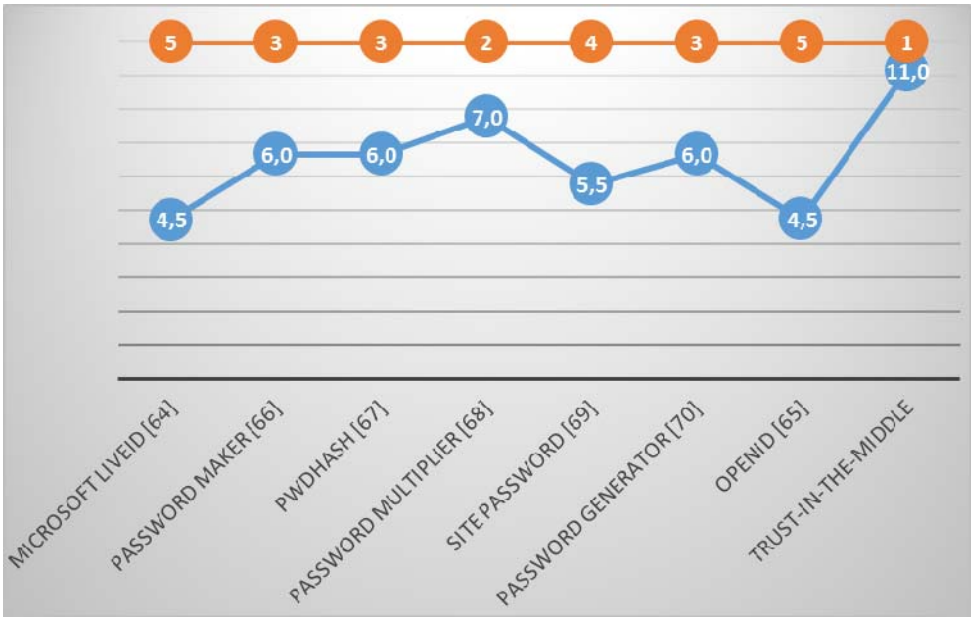


Figure 19: Security Scores of Password Managers

When we compare all password managers in terms of security in Figure 19, we see that Microsoft LiveID and OpenID have the worst scores and Trust-in-the-Middle has the best scores.

CHAPTER V

INCREASING TRUSTWORTHINESS OF POLL-SITE E-VOTING SYSTEM

In this chapter, we present our second proposal to increase trustworthiness of poll-site e-voting systems, a Trusted Computing based Three Ballot E-Voting System (Trusted3Ballot).

Three Ballot Voting has first been proposed by Rivest [100, 101] as a paper-based voting system which improves standard one ballot voting with several important functionalities (i.e. providing individual and universal verifiability by preventing vote trade) without using cryptography. In three-ballot voting scheme, not only can each voter verify that her vote is recorded as she intended, but she gets a "receipt" that she can take home to be used later to verify that her vote is actually included in the final tally. The voter is also able to check all the ballots in a bulletin board and verify the tally results. Voter's receipt, however, does not allow her to prove to anyone else how she voted. By this way vote trade may not be of concern.

As well as providing some important benefits, three ballot scheme suffers from the following issues:

- Three Ballot scheme puts an extra burden on user to understand and use the system in the right way when compared to the conventional voting systems. According to the [102], Three Ballot Scheme bothers significant usability issues.
- Since the number of paper ballots cast need to be handled is three times as large as with conventional ("One Ballot") voting, Three Ballot causes extra work for poll workers.
- Three Ballot suffers from Three-Pattern-Attack, Chain Voting, Malicious Checker Machine, Paying for Receipt, Voter's memorizing ballot IDs, ballot modification before casting and reconstruction attack problems (see Section V.2).

In our proposed system, we implement a trusted electronic three ballot system which both minimizes several usability issues and also provides a secure and trusted environment for each step in the election process to solve important security problems of Three Ballot utilizing trusted computing technologies. All used software in our framework are open-source and publicly available. So that anyone can inspect the codes before elections and carry out remote attestation during e-voting process to understand whether the software used is the one inspected before. The steps of the election process is transparently designed enabling different entities to take part and observe each process except the ones violating the privacy of the voter.

V.1 Three Ballot Scheme

We, in this section, introduce Rivest's three ballot scheme [100, 101] and give an overview of each step in the voting process.

V.1.1 Structure of the Ballot

Three-Ballot scheme, depicted in figure 20, has a multi-ballot structure having three columns each of which is a complete ballot on its own. Each ballot is identical except the ballot ID number, printed at the bottom and uniquely identifies the ballot among the ballots both on its own multi-ballot and the others. The perforations between each ballot is used to easily separate the ballots from each other.

There are two regions on the ballot: the upper part is voting region including candidate names and the corresponding bubbles to be filled by the voter. The lower part is ballot ID region.

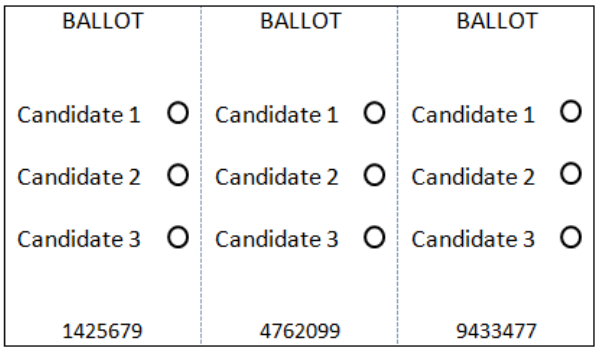


Figure 20: Empty Three-Ballot

V.1.2 Voting and Casting

In order to vote, voter randomly checks off two bubbles on the same row for the candidate he wants to vote, just one random bubble for the other candidates. Figure 21 illustrates an example filled three-ballot on which candidate 2 is selected.

After marking is completed, voter puts its multi-ballot into a checker machine which ensures the validity of the vote by checking the row and race constraints on the bullet.

Row constraint requires one or two marks in each row. Race constraint requires only one candidate has two marks in each race.

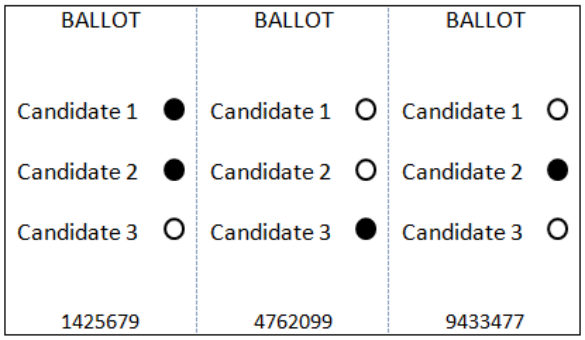


Figure 21: Voted Three-Ballot

If the row and race constraints hold, checker machine then puts a horizontal red stripe across the bottom of the multi ballot and cuts the multi ballot into three separate ballots along with the perforations. Voter can then cast each ballot separately.

V.1.3 Getting Receipt

Voter can also take copy of one of the ballots as receipt before casting operation. There may be several options to carry out receipt operation. However; one of convenient ways for the receipt operation is to do it in checker machine. Checker machine can ask voter's choice as which ballot will be used as receipt and prints a copy of the ballot as receipt. Receipt can be printed on a different colored paper in order to look different than the original ballots.

V.1.4 Publishing the Ballots on Bulletin Board

At the end of the Election Day, all ballots are scanned and published on a bulletin board. Scan operation is indeed a representation of the ballot with the selected bubbles and the ballot ID. It is not a pixel based scan for security reasons. The names of the voters who participated the election are also listed on bulletin board.

V.1.5 Individual Verification

In order to verify that his cast vote is included in the election (individual verification), each voter can basically look for the ballot that is identical with the receipt on bulletin board. If the voter cannot find a match, he can file a protest to the election office to declare that his vote has not been displayed on bulletin board. After checking the validity of the receipt, election office can decide to make a rescan of the cast ballots.

V.1.6 Tallying and Universal Verification

Since each ballot is published as clear text on bulletin board, tallying can be done by anyone by basically summing up the marks on each ballot. The only difference from the conventional one-ballot system is that the total value of each candidate has been inflated by the numbers of voters. So that the real number of votes can easily be found by subtracting the number of voters from the total number of marks for that candidate.

So universal verification is satisfied if the voter verifies that the total number of marks is three times the number of voters and the result of the individual tally operation is as same as the announced one on bulletin board.

V.2 Threat Model

In this section, we give a threat model and analyze each possible threats to the three ballot scheme in detail.

V.2.1 Three-Pattern Attack

In three pattern attack [100, 101], the voter is asked for marking a pre-specified pattern in each of her three ballots. So that attacker can check this pattern on bulletin board. If he finds a match, voter is awarded or otherwise voter might be punished.

V.2.2 Malicious Checker Machine

The security of checker machine is crucial for the security of the whole scheme. If the checker machine is compromised, various attacks can be mounted [100, 101]; For example if an adversary finds a way to eliminate the checker machines' control on row and race constraints, he can triple the number of votes he has given to his favorite candidate as each voter can cast three ballots. It is very difficult to detect this attack later on as it is impossible to check the constraints once the multi ballot has been split into separate ballots. This is an obvious violation of the core democratic principle – each voter should have an equal effect on the result.

A malicious checker can also note the ballot IDs of the receipts and gives the attacker ability to make modification on the other non-receipt ballots. Since the ballots selected as receipts are not changed, it is impossible for the voters to recognize the fraud.

V.2.3 Paying for Receipt

In this attack, adversary pays the voter to take his receipt when he leaves the poll site. After that, since the voter loses his ability to carry out individual verification,

adversary can hack the bulletin board and modify the corresponding ballot. In this attack, the more receipt the adversary obtains, the more he can affect the election results.

This is a complex attack as the adversary both needs to obtain a high amount of receipts and hack the bulletin board. However; it is not impossible.

V.2.4 Chain Voting

In order to start chain voting attack [103], the adversary needs to obtain an initial ballot somehow i.e. stealing a ballot before election, counterfeiting a ballot, getting one of the ballots out of the polling place and etc. After obtaining the initial ballot, the adversary marks the ballots for his candidate and hand in to a subverted voter who will then go to the polling booth, exchange the prefilled ballot with the blank ballot and return it back to the adversary. The same cycle is followed until the adversary cannot take the process further. The voters are paid if they agree to take place in the chain and follow the process or be punished if they do not return the ballot.

V.2.5 Voter's memorizing the Ballot IDs

By using advanced memory techniques, voter can keep all the ballot IDs in his mind and prove to a third party how he voted. This brings about the problem of vote trading.

V.2.6 Ballot Modification before Casting

After the approval of the checker machine, voter can make modification on the ballots such as marking extra bubbles on the ballots before he casts them into the ballot box. Since row or race constraints cannot be checked once the checker machine approves the ballot, it will not be possible to detect the modification.

V.2.7 Reconstruction Attack

In Three Ballot, Rivest makes Short Ballot Assumption which means that there are many voters in an election than ways to fill out an individual ballot [100, 101]. However; if this assumption does not hold, then reconstruction attack can be of concern.

Strauss showed through simulations in [104] that the three ballots of the voter can be reconstructed by using his receipt and all the ballots published in bulletin board. This is basically done by comparing the receipt with every possible pair ballots on the bulletin board. At the end, the attacker expects to find two other ballots that can form a unique three ballot with the receipt. If the ballot is lengthy including many races and candidates, then the number of possible patterns may be more than the number of voters, which increases the probability of checking a unique three ballot pattern.

V.3 Related Work

Three Ballot Voting has first been proposed by Rivest in [100] in 2006 as a paper-based voting offering individual and universal verifiability but preventing vote trade without using cryptography. The system was extended with some other paper based systems (VAV, Twin) and discussions on possible problems and potential solutions in [101] in 2007.

Cryptographic techniques can also provide all of the security properties of Three Ballot i.e. Chaum [105], Chaum et al. [106], Ryan et al. [107,108], Karloff et al. [109], Smith [110-112], and Adida [113].

The problems of Three Ballot voting scheme has been examined in several previous work [104, 114-117].

Appel presents a combined attack in [114] where attacker bribes or intimidates the voters to bring out a specific receipt. The attack is carried out changing some votes on the ballot box of a precinct. The attacker decides which changes he can do without being detected according to the obtained receipts.

Henry et al. provide a detailed analysis of known receipt-based attacks against Three Ballot voting system, focusing on two-candidate races in [115].

Storer, examines three pattern attack in his paper [116] and proposes a randomization device to mitigate.

Strauss in [117], has pointed out usability problems and potential receipt buying attacks against Three Ballot. Strauss [104] and Jones et al. [102,118] examine reconstruction attack in their work and provide some empirical results that prove the effectiveness of the attack.

In order to mitigate the reconstruction attack, Rivest, in [100], proposes to replace the receipts in a way similar to Farnel idea in [119, 120]. In this work, each voter replaces their receipt with some other's receipt using a Farnel like box. Rivest also presents some other mitigation techniques against reconstruction attack in his paper [101].

Araujo et al. in [121], draw attention to the point that the receipt may expose some statistical information about the vote (the leakage of information problem). By this way early information about the election results can be obtained. Araujo et al. in

[122] proposes some enhancements to the original Farnel scheme to mitigate reconstruction attack and leakage of information problem.

In [123], Clark et al. examine and compare the security of ballot receipts in three end-to-end auditable voting systems Pret a Voter [106, 107], Punchscan [124, 125] and Three Ballot [100, 101]. They find that Pret a Voter and Punchscan have similar security properties with respect to ballot receipts and provide no non-negligible information on the receipt itself that could compromise privacy and security. However, Three Ballot receipts leak partial information useful for compromising voter privacy and the integrity of the tally.

A Three Ballot based secure electronic voting system has been proposed by Costa et al. in [10]. The proposed system is based on classic cryptography techniques including the standard public key cryptosystem and addresses vote receipts, voter privacy and anonymity. The software utilizes web services and Election Markup Language [126]. One important drawback of the system is the trustworthiness and the security of the software and the keys have not been taken into account.

Smart et al. present a remote, coercion-resistant electronic voting protocol using trusted computing in [127]. With the proposed protocol, system verifies the state of the voter's (remote) machine and permits revocable anonymity.

Fink et al. propose using TPMs in direct recording electronic voting machines in [128]. They try to ensure election data integrity by binding voter's choices with the presented ballot.

Paul et al. propose a trustworthy voting: from machine to system, which is the main inspiration point of our proposed system. In [129], Paul et al. take into consideration

of each step in the election process and try to strengthen the security and trustworthiness of the scheme by utilizing trusted computing technologies as we do in Trusted3Ballot. There are some important problems of this work; the system is open to vote trade problems as it gives a receipt to the voter which clearly shows the selected candidate. For attestation an RS232 connection is being established with the voting machine which may cause an infection to the system. Paul et al. work does not support universal verifiability.

In our proposed scheme, we try to overcome all those problems explained in previous work by implementing a trusted computing based electronic three ballot voting system in a well-defined election process.

V.4 Proposed System

We, in this section, introduce our proposed system.

V.4.1 Voting Machine

Voting machine used in our proposed system is depicted in Figure 22. It has a diskless embedded computer system with a touch screen panel. Our trusted three ballot voting software is bundled in a bootable and secured operating system written in a CD. The kernel of the operating system is specifically designed to execute only the e-voting software, required modules and the relevant drivers to run DVD-ROM, optical storage (DVD RW) and printer. Other software are eliminated in order to keep the system as minimal as possible for security reasons. During the election period, the bootable CD is used to load the operating system and the voting software. DVD-ROM is used to load the bootable CD including the operating system and the trusted three ballot voting software. Token Reader is used to read chips or barcodes on cards.

An optical storage (DVD RW) is used to store the vote database. Vote token is a ticket including a barcode given by poll worker to the voter in order to make him identify himself to the system. The barcode on the token is read by the token reader.

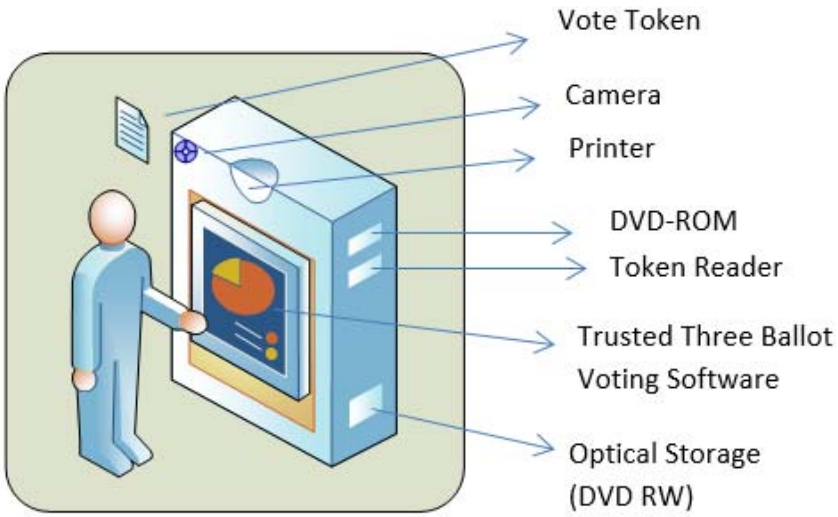


Figure 22: Voting Machine

V.4.2 Design Principles

Open Source Software. In our proposed system, we use open source operating system and software. So that anyone can inspect the code to understand the functioning of each module on the system and check whether there is any security breaches or not.

Simplicity. We believe that one of the main reasons why electronic voting system cannot be widely deployed is people's lack of trust to the system as they do not understand what is going on behind the screen. Although there are lots of different proposed solutions for e-voting, most of them employs heavy and complex cryptographic operations which are very difficult to understand by most of the

ordinary people using the system. Thus, while designing the system, we try to keep it as simple as possible. Although we also utilize some cryptographic operations of trusted computing in order to establish the security of the system and make it attested by users, we support the proposed scheme with human verifiable paper ballots and receipts as well, which we believe that it will increase the trust of nontechnical people to the system.

Transparency. All procedures except the ones violating the privacy of the voter are carried out in front of different actors and stakeholders taking role in the election. Each procedure is well documented and explained in detail.

Usability. Security-Usability tradeoff is one of the well-known conflicting issues in security domain as they negatively affect to each other. In this thesis work, while we try to increase the security of the system, we also try not to give up some important usability advantages of e-voting.

V.4.3 Preparation Phase

The security of the system highly depends on the security of the voting software and the operating system burned into the bootable CD. Therefore; we should ensure that the operating system and the voting software are trusted and establish the integrity of them until the end of the election. With this goal in mind, we follow the below steps in preparation phase:

a) Preparation Meeting: For each precinct, we organize an open meeting for CD preparation and invite different actors that have critical role in the elections i.e. members from different political parties, voting registrar, police department and etc. We assume that each actor sends at least one technical

person in order to be able to understand and follow the cryptographic operations.

b) Training: Each participant are informed in detail about the procedures to be followed, how operating system kernel is designed and the voting software functions. One copy of the source codes are handed in to the participants in a CD including the informative manual as well.

c) TPM Verification: During the meeting, technical members of each participant actor verify the certificates of TPM endorsement keys produced by the vendors. Hence, it is ensured that each machine has an original and enabled TPM.

d) AIK Certificate Generation: During the meeting, AIK key for each TPM is generated and the AIK certificates are created by a trusted third party. By this way, the signed values can securely be verified during attestation.

e) Attestation of Key Generation Software: Key generation software which is embedded in a bootable CD with the operation system is executed on a diskless PC. An attestation operation is executed via a portable device and the results are checked by all technical experts. We assume that the key generation CD and all the open source codes and the required checksums have already been publicly available for a while before this meeting and the experts inspected the codes before. This attestation can also be run by any technical expert who wishes to use his own portable device and the results are verified using the AIK certificate.

f) Election Key Generation: A public and private key generation for each precinct is carried out in front of the participants and private part has been split into different parts which are then delivered to different actors in CD or any other storage device. The main idea is that only when all those actors come together, the private key can be recovered. Let's call the generated public key as election public key in order not to confuse it with the other keys.

g) Preparation of Voting Software: The same procedures as key generation software are executed for our Trusted Three Ballot Voting software and if attestation is successfully performed, the bootable CD is encrypted with the election public key and kept secure until the Election Day.

h) Preparation of Barcode Box: Before the Election Day, election registrar prepares unique barcode ID pairs for each voter. However; these pairs are not linked to a specific voter. These barcode pairs are glued on a paper in a detachable format and enveloped. All the envelops including barcode ID pairs are grouped according to the number of voters of precincts and located into a barcode box which is then sealed to be opened in the election day. These barcode IDs are delivered randomly to the voter during the voting process as a proof indicating that the user has cast his vote.

V.4.4 Election Day

Booting Voting Machines and Initial Attestation

In Election Day, before election starts, each part holder of the election private key comes to the polling site. We assume that heavy legal sanctions are in place for those who do not bring the part of the private key on time and try to disrupt the election.

Poll workers start a decryption software via a bootable CD. First of all each part holder executes attestation on this software and then plug their CD's into the PC one by one. The system then forms the original private key by assembling each part and then requests the encrypted precinct election software. After receiving the encrypted CD, it performs decryption operation and writes the decrypted operating system and

the voting software in a new CD. Then poll workers gets this CD and boots each voting machine in the poll.

After each machine has been started, attestation operation has been performed one by one for each voting machine in front of the stakeholders and systems are verified whether they have been tampered or not since the preparation meeting.

Individual Attestation

Our voting environment allows voters to initiate individual attestation by using two methods; in the first one, voter declares to poll workers that he wants to perform attestation by using his portable device (mobile phone, tablet etc.). In the second one voter requests one of the mobile tablets of the precinct which have an internet connection.

In normal conditions, voters are not allowed to go into polling booth with a device in order to prevent any recording facility which may cause vote trading later on. However; if voter informs the poll worker that he is going to use the device for the attestation purposes, a special poll worker who is in charge of attestation operations, accompanies the voter during the attestation operation.

In order to perform attestation operation, attestation button is touched on the voting software. System asks the voter to input a challenge into the given input text field. Voter enters the challenge in the system, performs attestation and receives a signed result. Then he enters the received result into the mobile application on the portable device. This can for example be basically done by taking photo of the screen and automatically input it into the mobile application having OCR capabilities. The mobile application can then verify that the attestation with the given challenge is

correct and the AIK certificate is verified. We assume that voter has already loaded the correct AIK certificate published for his precinct by connecting the election web site before he comes to the polling site.

If voter does not have a mobile device capable of attestation, he can use the mobile tablet of precinct. By using the mobile tablet, voter connects one of the trusted web sites having capability of verification of the attestation result for the current election. We assume that there are several such web sites especially belonging to trusted certificate authorities serving in the internet.

By using one of this methodologies, users are able to carry out individual attestation before starting the voting process. After the attestation is completed the accompanying poll worker takes the devices from user and gets out of the polling booth. Please note that there is a camera on top of the voting machine recording the user activities (see figure 22) but is not able to display the screen of the voting software. After the attestation button is clicked the software can switch on a warning lamp located on the polling booth where camera can see and after the operation is finished this lamp can be switched off. By this way, the camera can follow whether the poll worker gets out of polling booth after the attestation operation. So any other process except the attestation operation cannot be done with a corrupted poll worker.

Voting Process

Voting process is depicted in figure 23. After voters pass a security check in the entrance of the poll site, they come to identification desk. Here, there are poll workers who have the list of all the voters assigned for this precinct. Poll worker requests an identity card from the voter, checks whether his identity number is in the list or not. If a biometric identity card is in use in the country, a biometric verification can also be

performed at this step. If it is ok, poll worker wants user to select one of the closed envelops from the barcode box.

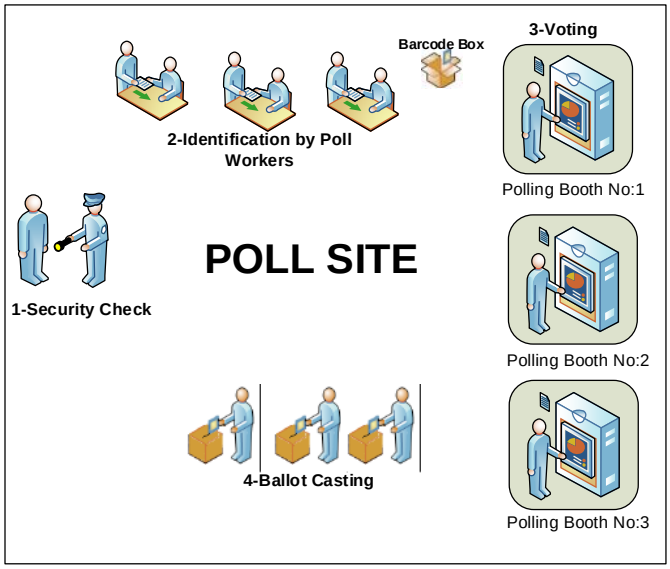


Figure 23: Voting Process in Poll Site

Voter opens the envelop, takes out the paper including two identical barcodes. He, then, removes the first barcode which is glued to the paper and sticks it on the voting form (see figure 24). After that, he writes his identity number and signs it. Voter makes the same operation for a second voting form and gives them to the poll worker. Poll worker also puts his signature on the forms and gives back one of them to the voter and puts the other one into a file. This form is kept as a record on both sides indicating that the voter has casted his vote and prevents anyone to cast a vote without having this uniquely prepared barcode number.

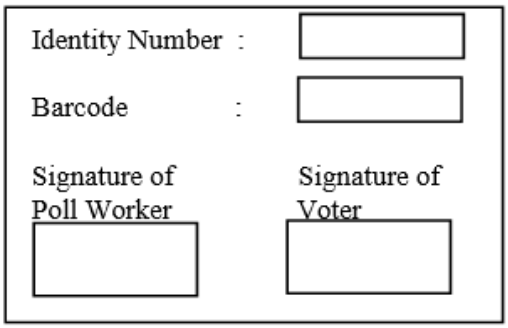


Figure 24: Voting Form

User enters the polling booth with this form and starts voting process by touching the vote button on the voting software. Onscreen directions tell the voter to put his barcode in the token reader. User completes the voting by following the instructions on the trusted three ballot voting software explained in Section V.5. During the voting, software holds two separate tables; one of them includes only the identity numbers and the associated barcodes. The other includes the electronic ballots. All the records are also located in random order in the table. By this way, associating the ballots with the voters are prevented.

After finishing the voting process, voter selects one of the three ballots as receipt and touches the print button. System prints the selected ballot as receipt in a different paper and the other three ballots separately as the original ballots. Voter completes the voting process by casting all three ballots in the ballot box, and taking the receipt one with him.

Tabulating the Votes and Publishing Results

At the end of the voting period, poll workers enter a CD into the optical storage of voting machines and enter a special code in the software to indicate that the election

is finished. The software encrypts the vote database with the election public key and writes them into the CD and also prints the total scores of the candidates in plaintext.

Then, the poll workers inform the election registrar about the results by phone and take all CDs and the ballot box to the headquarters with an escort. All these facilities are carried out in front of the observers of different stakeholders.

The first preliminary results obtained by phone calls are announced by the election registrar and the encrypted vote database including the ballots and the list of identity numbers and barcodes indicating who has participated the election are decrypted with the precinct's private keys created again by assembling each part from the different part holders. The decrypted ballots are stored in a common election database and each ballot and the identity numbers are published on the election bulletin board with the results. So that each voter having a receipt can check whether his own vote has been taken into account in the election and also perform universal verifiability. At the headquarters all the software running go through attestation process as it has happened in the poll site.

We note that paper based voting forms and the electronic list including the identity numbers and the associated barcodes are the proof that the voter has cast his vote. In terms of any objection, these electronic and paper based records including a hardcopy signature can be taken under inspection. Since there is not any association between the voting form and the user's vote, it is impossible to link the identity to a vote, which may violate the voter's privacy.

V.5 Prototype Trusted3Ballot Software

Throughout this thesis work, we have developed a prototype Trusted3Ballot software seen in figure 25. System is designed for touch screen usage. Our prototype

Trusted3Ballot software presents voters randomly prefilled ballots. Selected bubbles (red ones) are configured as disabled and cannot be changed by the voter. When a voter wants to vote, he has to mark an extra bubble belonging to the candidate he would like to vote (see figure 26). Three-ballot voting row and race constraints have been automatically implemented in the system. Voter is able to mark only one extra bubble. The system does not let the voter to mark more than one bubble.

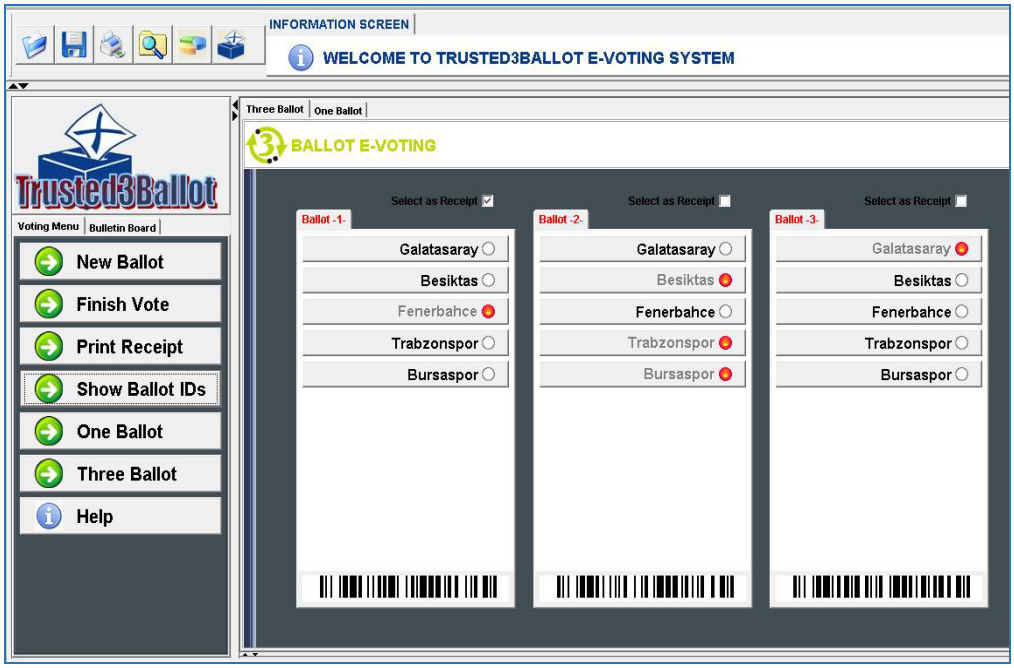


Figure 25: Three Ballot Screen

Ballot ID numbers which are unique for each ballot is displayed as barcodes at the bottom of each ballot. Since our system is a test system, we are able to view and hide ballot IDs. As it seen in Figure 26 when “Show Ballot ID” button is clicked, system displays each ballot ID at the bottom of the ballots. However; in the original implementation Ballot IDs should only be seen in the printed receipt in order to make

it possible for the candidates willing to verify his receipt but does not have a barcode reader. By this way voter's memorizing ballot IDs problem is prevented.

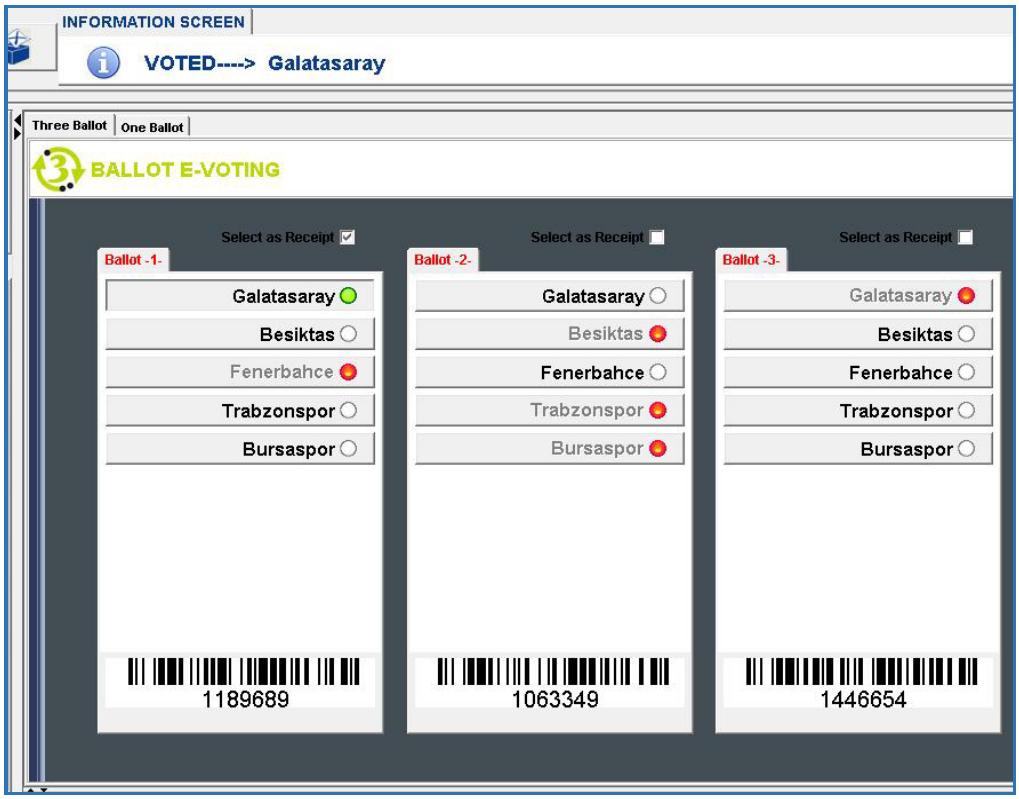


Figure 26: Voted Three Ballot Screen

In order to increase the usability of three ballot system and prevent misvoting, we placed an information screen at top of the window showing which candidate is voted. System is also designed to support one ballot voting. So that voters can change between one ballot and three ballot voting screens (see figure 27). However; the printed ballot which will be cast in ballot box is in three ballot form.

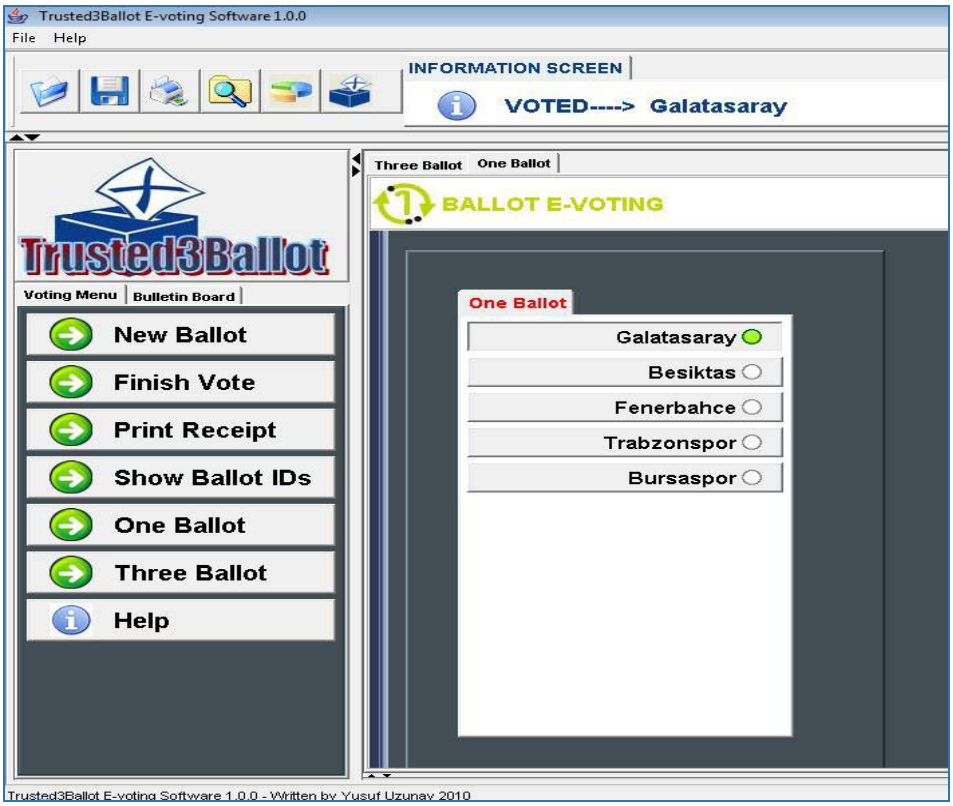


Figure 27: One Ballot Screen

After the voting has finished, Trusted3Ballot system gives the user the ability to verify his vote by checking whether his receipt is included in the final tally. This can be done basically connecting to the Election Bulletin Board and searching his receipt according to the Ballot ID (see figure 28).

Voters can also view all ballots used in the election and can calculate the result of election himself, which we refer as universal verification.

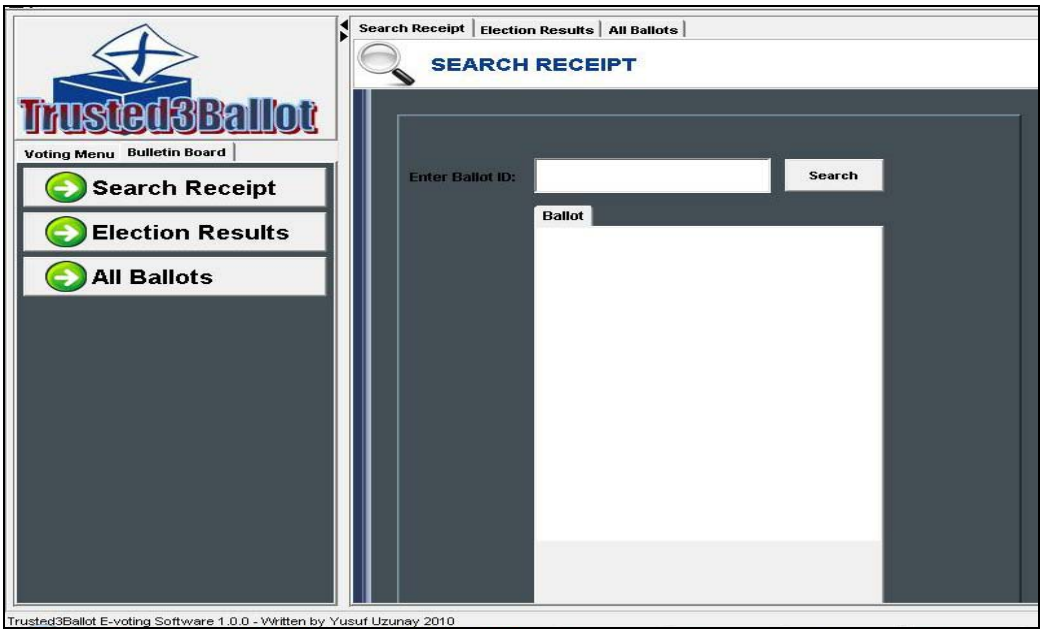


Figure 28: Searching Receipt

V.6 Security and Usability Analysis

In this section, we analyze both the security of the proposed system by taking into account the threat model given in section IV.1 and how usability issues are taken into account.

Security Analysis

Three Pattern Attack: In Trusteted3Ballot system, the ballots are prefilled randomly and cannot be changed by the voter. So that it is impossible for the voter to select a pre-specified pattern in each ballot.

Malicious Checker Machine: In Trusted3Ballot system, voting machine does all the checker machine functionalities. The software of the voting machine is open source and by using TPM based remote attestation the code can be verified by anyone (voters, observers, poll workers etc.). All the security critical software in the election process go through attestation process and the secret data are stored as encrypted. Private key used in decryption is separated into different parts and each of them are delivered to different holders. So that unless all the part holders come together and assemble the key, private key cannot be recovered and the decryption becomes impossible.

Paying for Receipt: In Trusted3Ballot system, the trustworthiness of bulletin board is also provided by TPM remote attestation, hence the modification of bulletin board will not be possible, no matter the adversary captures the receipts of the voters.

Chain Voting: In Trusted3Ballot, paper based ballots are only used at the time of casting after electronic voting. Therefore; system does not permit to acquire an empty paper ballot to establish the chain before the voting. System also gives a specific ID for each ballot during the voting which makes it impossible to insert a paper ballot with the correct ballot ID without hacking the voting machine.

Voter's Memorizing the Ballot IDs: Ballot IDs are displayed in the form of barcodes and only seen after printing the receipt. Since the voters are not allowed to bring any device to read or capture votes or barcodes, it is impossible for the user to memorize the ballot IDs.

Ballot Modification before Casting: Since both electronic and paper based ballots are used. Any modification on the paper ballot can easily be detected. If any dispute is of

concern, we believe that criminal investigation can easily reveal whether the ballot is changed by hand or not.

Reconstruction Attack: In order to prevent reconstruction attack, our scheme can be extended with floating receipts method explained in [101], which means having voters take home copies of receipts other than their own. This method fully breaks the connection between the voter's receipt and his vote by adding a new anonymity layer.

Usability Analysis:

In paper based three ballot mechanism, users have to put four marks instead of one when compared to standard one ballot mechanism. Furthermore; sometimes it might be difficult to tell the voting logic behind three ballot to some voter profiles such as elderly people.

While we are designing our proposed system, not only a secure architecture is put into place, but usability problems of paper based three ballot are minimized as well. Usability improvements of the proposed system can be listed as follows:

- Trusted3Ballot displays randomly pre-filled ballots to the users. Therefore; users do not need to mark 4 bubbles for a 3 candidate election. He only marks one bubble as it is in standard voting.
- In order to prevent confusion, pre-filled bubbles are marked as red and the bubble voted by voter is marked green.
- Three ballot constraints have been embedded in the system. User cannot change prefilled bubbles and cannot mark more than one bubble. When user tries to mark another bubbles, previously marked bubble returns to unmarked.
- There is an information screen at top of the window in order to show the voted candidate

- There is also one ballot interface for the voters who have difficulty in voting with three ballots. One ballot and three ballot interfaces work synchronous. When one of the candidates is voted in one scheme, it is also voted on the other scheme.
- System can easily be extended with audio and video technologies to aid people with disabilities.

CHAPTER VI

CONCLUSION and FUTURE WORK

In this dissertation, we have focuses on how we can increase the trustworthiness of security critical applications using trusted computing technologies. We have selected two case applications, authentication proxy systems and e-voting systems. After analyzing all those systems in detail throughout this thesis work, we have come up with two proposals, Trust-in-the-Middle – a trusted computing based authentication proxy system and Trusted3Ballot – a trusted computing based three ballot e-voting system.

In our first proposed system, Trust-in-the-Middle, our goal was to increase the security of the proxy system in order to make it a trustworthy central intermediary system which takes over credential storage and submission operations from its users.

Using TPM DRTM functionality, Trust-in-the-Middle securely authenticates users and stores their credentials on proxy encrypted with TPM protected keys. Whenever these credentials need to be used, they are securely decrypted and submitted to the target servers.

Security critical operations on the proxy are performed in an isolated environment protected by TPM DRTM and credentials are never disclosed to outside without

being encrypted. Attestation is used to verify the integrity of security sensitive code running in TPM DRTM protection and the modules running on proxy. Only if the verification succeeds, then the sensitive data is sent to these modules.

Main contribution of Trust-in-the-Middle is its being the first system that takes into account the security of third party authentication proxy systems and establishing the trustworthiness of them utilizing trusted computing technologies. Trust-in-the-Middle also presents a novel proxy security architecture with several brand new protocols. From performance aspects, although Trust-in-the-Middle does not have promising scores on proxy system due to some slow TPM operations, it offers significant gain on user side (i.e. more than 90% in credential submission operation).

According to UDS (Usability-Deployability-Security) analysis performed in Section IV, Trust-in-the-Middle has the top security scores among twenty previous work including proxy based systems, TPM based systems and password managers. Regarding usability, although it has average scores among TPM based systems and password managers, it has again best usability scores among other proxy based systems. Trust-in-the-Middle has not promising deployability scores when compared to other password managers and proxy-based systems. Nevertheless; it has second best deployability score among other TPM based systems.

In the current version of Trust-in-the-Middle, run-time integrity problems are not taken into consideration. This is a serious limitation considering the security of user credentials on proxy. Therefore, a possible future work can be to adopt run-time security measures in the proposed system. Furthermore; in order to prevent real time malicious code attacks and transaction generators, we plan to integrate a second channel authentication in proxy authentication phase and at each transactions.

Our proof-of-concept implementation requires a special software installation on user side which may be considered as not being a usable solution. Although we have performed a preliminary usability study with the aim of evaluating the performance of the proposed system, more advanced and carefully-crafted usability studies can be carried out as a future work.

Proxy systems are used not only for user authentication but also for many other security and privacy purposes (e.g., [130]). Since proxy trust problem is common in all of these applications, we think it is a promising future work to investigate on positioning our solution as a more general framework.

In our second proposal, we present a trusted computing based three ballot e-voting system. We take into account each step in the election process in detail and propose a secure framework built on top of trusted computing technologies.

By utilizing three ballot voting mechanism, our proposed system satisfies important and contradictory requirements of voting such as providing individual and universal verifiability without causing vote trade problems. The main contribution of our work is integrating three ballot scheme into an electronic voting system secured by trusted computing technologies and giving users ability to attest the software during e-voting. The second contribution is solving various security and usability problems of three ballot scheme in the given architecture without giving up security.

Each security critical software taking a role in the election process has gone under inspection by different parties and the codes of the software are attested before being used.

We have also developed a prototype electronic three ballot software and showed how we can minimize most of the usability and security problems of paper based three

ballot mechanism. Since the system also incorporates human verifiable paper ballots, we believe that user trust problem into the electronic voting systems is minimized and the recovery of the election becomes possible by counting the paper ballots in terms of any dispute.

Trusted3Ballot system is designed as a poll-site e-voting system. One future work may be to utilize trusted computing technologies to increase the trustworthiness of remote voting as well. Another future work may be to evaluate the usability aspects of each election step performing different usability studies.

In closing, we see that trustworthiness of security critical software is one of the challenging issues. With this regard, trusted computing technologies can offer many functionalities. In this thesis, we have applied this technology in authentication proxy systems and e-voting systems and showed how this technology can improve the trustworthiness of those systems.

REFERENCES

[1] *Symantec Internet Security Threat Report: Trends for 2010*. Volume 16. April 2011.

[2] *2011 CWE/SANS Top 25 Most Dangerous Software Error*. SANS Institute, Retrieved December 30, 2013, from http://cwe.mitre.org/top25/#Categories

[3] P.G. Neumann. Risks of untrustworthiness. In *Proceedings of 22nd Annual Computer Security Applications Conference (ACSAC)*, 2006.

[4] A. Pashalidis and C. J. Mitchell. Impostor: A single sign-on system for use from untrusted devices. In *Proceedings of the IEEE Globecom*, 2004.

[5] M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. In *Proceedings of the DIMACS Workshop on Usable Privacy and Security Software*, 2004.

[6] R. C. Jammalamadaka, T. W. v. d. Horst, S. Mehrotra, K. E. Seamons, and N. Venkasubramanian. Delegate: A proxy based architecture for secure website access from an untrusted machine. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, IEEE Computer Society, pages 57–66, 2006.

[7] D. Florencio and C. Herley. KLASSP: Entering passwords on a spyware infected machine using a shared-secret proxy. In *Proceedings of the 22nd Annual Computer Security Applications Conference*, ser. ACSAC '06. IEEE Computer Society, pages 67–76, 2006.

[8] D. Florencio and C. Herley. One-time password access to any server without changing the server. In *Proceedings of the 11th international*

conference on Information Security, ser. ISC ’08. Springer-Verlag, pages 401–420, 2008.

[9] J. Martineau and P. A. Kodeswaran. *Securepass: Guarding sensitive information from untrusted machines*. Retrieved November 17, 2013, from http://www.csee.umbc.edu/~palanik1/SecurePassPaper.pdf, 2008.

[10] A.O. Santin, R.G. Costa and C.A. Maziero. A Three-ballot-based secure electronic voting system. *IEEE Security & Privacy*, 6(3), 14-21, 2008.

[11] M. Byrne, K. Greene and S. Everett. Usability of voting systems: baseline data for paper, punch cards and lever machines. In *Proceedings of Politics & Activism*, ACM Press, 1,171-180, 1997.

[12] R. Aditya, B. Lee, C. Boyd and E. Dawson. Implementation issues in secure e-voting schemes. In *Proceedings in the 5th Asia-Pacific Industrial Engineering and Management Systems Conference*, Goldcoast, Australia, 2004.

[13] *Countries with e-voting projects*. RNIB Scientific Research Unit. Retrieved November 17, 2013, from http://www.tiresias.org/research/ guidelines/evoting_projects.htm

[14] B. Schneier. *The Problem with Electronic Voting Machines*. Retrieved November 17, 2013, from http://www.schneier.com/blog/archives/2004/11/ the_problem_wit.html

[15] D. Wallach. *Vendor misinformation in the e-voting world*. Retrieved November 17, 2013, from http://freedom-to-tinker.com/blog/dwallach/vendor-misinformation-e-voting-world

[16] Y.Uzunay, K. Bicakci. Trust-in-the-Middle: Towards establishing trustworthiness of authentication proxies using trusted computing, in submission to *IEEE Transactions on Computers*, 2013.

[17] Y.Uzunay, K. Bicakci. “Trusted3Ballot: Improving security and usability of threeballot voting system using trusted computing”, in *Proceedings of*

ISMS2014 5th International Conference on Intelligent Systems, Modelling and Simulation, IEEE Computer Society, Langkawi, Malaysia, 2014.

[18] *Trusted computing group web site*. Retrieved November 17, 2013, from https://www.trustedcomputinggroup.org

[19] Common Criteria for Information Technology Security Evaluation. Part 1: Introduction and General Model, August 1999.

[20] Common Criteria for Information Technology Security Evaluation. Part 2: Security Functional Requirements, August 1999.

[21] Common Criteria for Information Technology Security Evaluation. Part 3: Security Assurance Requirements, August 1999.

[22] *Trusted Computing Group (TCG). TPM Main Specification — Part 1: Design Principles*. Retrieved November 17, 2013, from http://www.trustedcomputinggroup.org/resources/tpm_main_specification

[23] *Federal Information Processing Standards (FIPS). FIPS PUB 140-2: Security Requirements for Cryptographic Modules*. Retrieved November 17, 2013, from http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[24] *LPC Bus Definition*. Retrieved November 17, 2013, from http://en.wikipedia.org/wiki/Low_Pin_Count

[25] *Southbridge Definition*. Retrieved November 17, 2013, from http://en.wikipedia.org/wiki/Southbridge_(computing)

[26] *Random Number Generation Definition*. Retrieved November 17, 2013, from http://en.wikipedia.org/wiki/Random_number_generation

[27] *Trusted Platform Module (TPM) Specification Overview*. Retrieved November 17, 2013, from http://www.fidis.net/resources/deliverables/hightechid/int-d37002/doc/9/

[28] D. Challener, K. Yoder, R. Catherman, D. Safford and L.V. Doorn. *A Practical Guide to Trusted Computing*, IBM Press, 2008.

[29] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In Proceedings *of the 11th ACM conference on Computer and communications security*, ser. CCS '04. ACM, pages 132–145, 2004.

[30] *Trusted Computing: TCG Proposals*. Retrieved November 17, 2013, from http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/TrustedComputingTCG.html

[31] B. Kauer. OSLO: Improving the security of trusted computing. In *16th USENIX Security Symposium*, August, 2007.

[32] R. MacDonald, S. W. Smith, J. Marchesini, and O. Wild. *Bear: An open-source virtual secure coprocessor based on TCPA*. Technical Report TR2003-471, Dartmouth College, Hanover, NH, August 2003.

[33] J. Marchesini, S. W. Smith, O. Wild and R. MacDonald. *Experimenting with tcpa/tcg hardware, or: How I learned to stop worrying and love the bear*. Technical Report TR2003-476, Dartmouth College, Hanover, NH, December 2003.

[34] *Enforcer Project*. Retrieved November 17, 2013, from http://enforcer.sourceforge.net

[35] H. Maruyama, F. Seliger, N. Nagaratnam, T. Ebringer, S. Munetoh, S. Yoshihama, and T. Nakamura. *Trusted platform on demand*. Technical Report RT0564, IBM Corporation, February 2004.

[36] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the USENIX Security Symposium*, August 2004.

[37] B. Kauer. *Authenticated booting for L4*. TU Dresden, November 2004.

[38] *TCG PC Client Implementation Specification for Conventional BIOS*. Retrieved February 20, 2010, from https://www.trustedcomputinggroup.org/specs/PCClient

[39] *Advanced Micro Devices, AMD64 virtualization: Secure virtual machine architecture reference manual*. AMD, Publication No. 33047, Rev.3.01. May 2005.

[40] *Intel Corporation, LaGrande technology preliminary architecture specification*. Intel, Publication No. D52212. May 2006.

[41] *Multiboot Specification*. Retrieved November 17, 2013, from http://www.gnu.org/software/grub/manual/multiboot/multiboot.txt

[42] *SYSLINUX Project*. Retrieved November 17, 2013, from http://syslinux.zytor.com

[43] *Kexec Article*. Retrieved November 17, 2013, from http://lwn.net/Articles/15468

[44] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: An execution infrastructure for TCB minimization. In *Proceedings of the ACM European Conference in Computer Systems (EuroSys)*, April 2008.

[45] A. Karole, N. Saxena, and N. Christin. A comparative usability evaluation of traditional password managers. In *Proceedings of the 13th international conference on Information security and cryptology*, ser. ICISC'10. Springer-Verlag, pages 233–251, 2011.

[46] J. M. McCune, A. Perrig, and M. K. Reiter. Safe passage for passwords and other sensitive data. In *Proceedings of the Symposium on Network and Distributed Systems Security (NDSS),* 2009.

[47] C. Li, A. Raghunathan, and N. K. Jha. A secure user interface for web applications running under an untrusted operating system. In *Proceedings*

of the 10th IEEE International Conference on Computer and Information Technology. IEEE Computer Society, pages 865–870, 2010.

[48] K. Borders and A. Prakash. Securing network input via a trusted input proxy. In *Proceedings of the 2nd USENIX workshop on hot topics in security*, ser. HOTSEC'07. USENIX Association, pages 7:1–7:5, 2007.

[49] M. Hirano, T. Umeda, T. Okuda, E. Kawai and S. Yamaguchi. T-PIM: trusted password input method against data stealing malware. In *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, ser. ITNG'09. IEEE Computer Society, pages 429–434, 2009.

[50] S. Gajek, A. R. Sadeghi, C. Stuble, and M. Winandy. Compartmented security for browsers - or how to thwart a phisher with trusted computing. In *Proceedings of the Second International Conference on Availability, Reliability and Security*, ser. ARES'07, pages 120–127, 2007.

[51] S. Gajek, H. Lohr, A. R. Sadeghi, and M. Winandy. Truwallet: trustworthy and migratable wallet-based web authentication. In *Proceedings of the 2009 ACM workshop on Scalable trusted computing*, ser. STC '09, 2009.

[52] T. Garfinkel and M. Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th Workshop on Hot Topics in Operating Systems,* ser. HOTOS '05, 2005.

[53] K. Kostiainen, J. E. Ekberg, N. Asokan, and A. Rantala. On-board credentials with open provisioning. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security,* ser. ASIACCS '09, 2009.

[54] S. Bugiel and J.-E. Ekberg. Implementing an application-specific credential platform using late-launched mobile trusted module. In *Proceedings of the fifth ACM workshop on Scalable trusted computing*, ser. STC '10, 2010.

[55] S. Bugiel, A. Dmitrienko, K. Kostiainen, A. R. Sadeghi, and M. Winandy. TruWalletM: secure web authentication on mobile platforms. In *Proceedings of the Second international conference on Trusted Systems*, ser. INTRUST'10, 2011.

[56] A. Pashalidis and C. J. Mitchell, Single sign-on using trusted platforms, in *Proceedings of the 6th International Conference on Information Security*, ser. ISC 2003. Springer-Verlag, pages 54–68, 2003.

[57] *Cloud Computing and Security - A Natural Match*. Retrieved November 17, 2013, from http://www.trustedcomputinggroup.org/resources/cloud_computing_and_security__a_natural_match

[58] K. Patidar, R. Gupta, G. Singh, M. Jain, and P. Shrivastava. Integrating the trusted computing platform into the security of cloud computing system. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(2), 2012.

[59] Z. Liu, F. Wu, and K. S. W. Chai. C-MAS: The cloud mutual authentication scheme. In *2nd International Conference on Computer and Information Applications (ICCIA 2012)*, 2012.

[60] P. Senthil, N. Boopal, and R. Vanathi. Improving the security of cloud computing using trusted computing technology. *International Journal of Modern Engineering Research (IJMER)*, 2(1): 320–325, 2012.

[61] J. Naruchitparames and M. H. Gunes. Enhancing data privacy and integrity in the cloud. In *HPCS*, pages 427–434, 2011.

[62] Z. Shen, L. Li, F. Yan, and X. Wu. Cloud computing system based on trusted computing platform. In *Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation*, ser. ICICTA'10, 2010.

[63] N. Santos, R. Rodrigues, K. P. Gummadi and S. Saroiu. Policy-sealed data: a new abstraction for building trusted cloud services. In *Proceedings of the 21st USENIX conference on Security symposium,* ser. Security'12, 2012.

[64] *Windows Live ID*. Retrieved June15, 2013, from http://www.passport.net

[65] *Open ID*. Retrieved November 17, 2013, from http://openid.net

[66] *Password Maker*. Retrieved November 17, 2013, from http://passwordmaker.org

[67] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th conference on USENIX Security Symposium*, ser. SSYM'05, 2005.

[68] J. A. Halderman, B. Waters, and E. W. Felten. A convenient method for securely managing passwords. In *Proceedings of the 14th international conference on World Wide Web*, ser. WWW '05, 2005.

[69] A.Karp. *Site-specific passwords*. Tech. Rep., 2002, Retrieved November 17, 2013, from http://www.hpl.hp.com/personal/Alan_Karp/site_password/site_password_files/site_password.pdf.

[70] N. Wolf. *Password Generator*. Retrieved November 17, 2013, from http://angel.net/~nic/passwd.html

[71] *Security Assertion Markup Language (SAML) OASIS Standard*. Retrieved November 17, 2013, from http://saml.xml.org

[72] *Yadis Discovery Protocol*. Retrieved November 17, 2013, from http://infogrid.org/trac/wiki/Yadis

[73] *OAuth Web Site*. Retrieved November 17, 2013, from http://oauth.net

[74] Liberty Alliance Web Site, Retrieved November 17, 2013, from http://www.projectliberty.org

[75] A. Leicher, A. U. Schmidt, Y. Shah, and I. Cha. Trusted computing enhanced opened. In *2010 International Conference for Internet Technology and Secured Transactions (ICITST)*, 2010.

[76] A. Leicher, A. U. Schmidt, Y. Shah, and I. Cha. Trusted computing enhanced user authentication with openid and trustworthy user interface. *International Journal of Internet Technology and Secured Transactions*, 3(4) 331–353, 2011.

[77] Intel Corporation. *Intel Trusted Execution Technology Software Development Guide*. Intel, March 2011.

[78] Intel Corporation. *Trusted boot (tboot)*. Retrieved November 17, 2013, from http://sourceforge.net/projects/tboot

[79] D. Boneh, S. Inguva, and I. Baker. *SSL MITM proxy*. Retrieved November 17, 2013, from http://crypto.stanford.edu/ssl-mitm

[80] *Putty Download Page*. Retrieved November 17, 2013, from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[81] *How to create an SSH tunnel using Putty and then use that tunnel as a Firefox SOCKS proxy*. Retrieved November 17, 2013, from http://www.devdaily.com/unix/edu/putty-ssh-tunnel-firefox-socks-proxy

[82] *OPIE Linux Wiki Page*. Retrieved November 17, 2013, from http://wiki.linuxquestions.org/wiki/Opie

[83] *1Key application*. Retrieved November 17, 2013, from http://itunes.apple.com/tr/app/1key/id295500470?mt=8

[84] *Proxy Switch Add-on*. Retrieved November 17, 2013, from http://www.proxy-offline-browser.com/ProxySwitch/

[85] *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, 4-294 Vol. 2B*. Retrieved November 17, 2013, from http://download.intel.com/products/processor/manual/325383.pdf

[86] *Keystroke Logging Definition*. Retrieved November 17, 2013, from http://en.wikipedia.org/wiki/Keylogger

[87] *Acoustic Keylogger definition*. Retrieved February 12, 2011, from http://www.keyloggerreview.com/acoustic-keyloggers

[88] M. Vuagnoux and S. Pasini. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th USENIX Security Symposium*, Canada, 2009.

[89] *Screen-scraper definition*. Retrieved November 17, 2013, from http://en.wikipedia.org/wiki/Data_scraping

[90] N.Lawson. *TPM hardware attacks*. Retrieved December 30, 2013 from http://rdist.root.org/2007/07/16/tpm-hardware-attacks

[91] D. Schellekens, B. Wyseur, and B. Preneel. Remote attestation on legacy operating systems with trusted platform modules. *Sci. Comput. Program*. 74(1-2): 3–22, 2008.

[92] A. M. Azab, P. Ning, E. C. Sezer, and X. Zhang. Hima: A hypervisor-based integrity measurement agent. In *Proceedings of the 2009 Annual Computer Security Applications Conference, ser. ACSAC '09*. IEEE Computer Society, pages 461–470, 2009.

[93] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, ser. SOSP '05. ACM, pages 1–16, 2005.

[94] M. Ceccato, Y. Ofek, and P. Tonella. Remote entrusting by run-time software authentication. In *Proceedings of the 34th conference on Current trends in theory and practice of computer science*, ser. SOFSEM'08. Springer-Verlag, pages 83–97, 2008.

[95] L. Gu, X. Ding, R. H. Deng, B. Xie, and H. Mei. Remote attestation on program execution. In *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, ser. STC '08. ACM, pages 11–20, 2008.

[96] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. TrustVisor: efficient TCB reduction and attestation. In *Proceedings of the IEEE Symposium on Security and Privacy*, ser. SP '10. IEEE Computer Society, pages 143–158, 2010.

[97] N.Lawson. *TPM Hardware Attacks (part 2)*. Retrieved December 30, 2013 from http://rdist.root.org/2007/07/17/tpm-hardware-attacks-part-2

[98] C. F. Cid. *Cryptanalysis of RSA: A Survey*. SANS Institute. Retrieved December 30, 2013 from http://www.sans.org/reading-room/whitepapers/vpns/cryptanalysis-rsa-survey-1006

[99] J. Bonneau, C. Herley, P. C. van Oorschot, F. Stajano. The quest to replace passwords: a framework for comparative evaluation of web authentication schemes, In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, p.553-567, 2012.

[100] R. Rivest. *The ThreeBallot Voting System*. Retrieved November 17, 2013, from http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf

[101] R. Rivest and W. Smith. Three voting protocols: threeballot, vav, and twin. *Usenix/Accurate Electronic Voting Technology Workshop in 16th Usenix Security Symp*osium, 2007.

[102] H. Jones, J. Juang and G. Belote. *Threeballot in the field*. Fall 2006. Term paper for MIT course. Retrieved November 17, 2013, from http://courses.csail.mit.edu/6.857/2006/projects/threeBallotPaper.pdf

[103] D. W. Jones, *Chain voting*. Retrieved November 17, 2013 from http://www.bbvdocs.org/reports/NIST-Threats/ChainVoting.pdf

[104] C.Strauss. *A critical review of the triple ballot voting system. part 2: cracking the triple ballot encryption draft version 1.5*, Verified Voting New Mexico. Retrieved November 17, 2013, from http://www.cs.princeton.edu/~appel/voting/Strauss-ThreeBallotCritique2v1.5.pdf October 2006.

[105] D. Chaum. Secret ballot receipts: True voter-verifiable elections. *IEEE Journal of Security and Privacy,* pages 38 – 47, 2004.

[106] D. Chaum, P. Y. A. Ryan, and S. A. A. Schneider. *Practical, voter-verifiable election scheme*. Tech. Rep. CS-TR- 880, University of Newcastle upon Tyne School of Computing Science, 2004.

[107] P. Y. A. Ryan and T. Peacock. *Pret a Voter: A system perspective*. Tech. Rep. CS-TR-929, University of Newcastle upon Tyne School of Computing Science, 2005.

[108] P. Y. A. Ryan and S. A. Schneider. *Pret a Voter with re-encryption mixes*. Tech. Rep. CS-TR-956, University of Newcastle upon Tyne School of Computing Science, 2006.

[109] C. Karlof, N. Sastry and D. Wagner. Cryptographic voting protocols: A system perspective. In *Proceedings 14th USENIX Security Symposium*, August, 2005.

[110] W. D. Smith. *Cryptographic election protocols for reweighted range voting & reweighted transferable vote voting*. Retrieved November 25, 2013, from http://scorevoting.net/WarrenSmithPages/homepage/crirv.pdf

[111] W. D. Smith. New cryptographic voting scheme with best-known theoretical properties. In *Proceedings of Workshop on Frontiers in Electronic Elections,* 2005.

[112] W. D. Smith. *Cryptography meets voting*, Retrieved November 25, 2013, from http://www.hit.bme.hu/~buttyan/courses/BMEVIHI5316/Smith.Crypto_meets_voting.pdf

[113] B. Adida. *Advances in cryptographic voting systems*. Ph.D Thesis at MIT Department of EECS, August 2006.

[114] A. W. Appel. *How to defeat Rivest's threeballot voting system*. Retrieved November 17, 2013, from http://www.cs.princeton.edu/~appel/papers/DefeatingThreeBallot.pdf , 2007.

[115] K. Henry, D. R. Stinson, and J. Sui. The effectiveness of receipt-based attacks on threeballot. *IEEE Transactions on Information Forensics and Security*, 4(4):699-707, 2009.

[116] T. Storer. *Identification and mitigation of a vulnerability in the threeballot voting scheme*. Retrieved November, 25, 2013, from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1975&rep=rep1&type=pdf 2006.

[117] C. Strauss. *The trouble with triples: A critical review of the triple ballot (3ballot) scheme. Part 1*, Verified Voting New Mexico, Retrieved November 17, 2013, from http://www.cs.princeton.edu/~appel/voting/Strauss-TroubleWithTriples.pdf , October, 2006

[118] G. Belote, H. Jones, and J. Juang. *Threeballot analysis*. Term paper presentation for MIT class 6.857 Fall2006. Retrieved November 17, 2013, from http://courses.csail.mit.edu/6.857/2006/projects/threeBallotPresentation.pdf

[119] A. J. Devegili. Farnel: Uma proposta de protocol criptogr´afico para vota¸c˜ao digital (in portuguese). Master's thesis, Curso de P´os-Gradua¸c˜ao em Ci^encia da Computa¸c˜ao da Universidade Federal de Santa Catarina, Florian´opolis, Santa Catarina, Brasil, 2001.

[120] R. Araujo, R. Custodio, A. Wiesmaier, and T. Takagi. An electronic scheme for the Farnel paperbased voting protocol. In *ACNS'06*, 2006.

[121] R. Araujo, R.F. Custodio, and J. van de Graaf. A verifiable voting protocol based on farnel. *IAVoSS Workshop on Trustworthy Elections (WOTE'07)*, June 2007.

[122] R. Araujo. *Improving Farnel, Threeballot, and Randell-Ryan Voting Schemes*. IACR Cryptology ePrint Archive (2008): 82, 2008.

[123] J. Clark, A. Essex and C. Adams. On the security of ballot receipts in E2E voting systems. In *Proceedings of Workshop on Trustworthy Elections*, 2007.

[124] K. Fisher, R. Carback and A.T. Sherman. Punchscan: introduction and system denition of a high-integrity election system. In *Proceedings of Workshop on Trustworthy Elections 2006*.

[125] S. Popoveniuc and B. Hosp. An introduction to Punchscan. In *Proceedings of Workshop on Trustworthy Elections 2006*.

[126] Oasis, *The Case for Using Election Markup Language (EML)*. Oasis Election and Voter Services TC, 2007. Retrieved November 17, 2013, from http://www.oasis-open.org/committees/election

[127] M. Smart and E. Ritter. True trustworthy elections: remote electronic voting using trusted computing. *Autonomic and Trusted Computing*. Springer Berlin Heidelberg, 187-202, 2011.

[128] R.A. Fink, A.T. Sherman, R. Carback. TPM meets DRE: reducing the trust base for electronic voting using trusted platform modules. *IEEE Transactions on Information Forensics and Security*, 4(4):628-637, 2009.

[129] N. Paul, A. S. Tanenbaum. Trustworthy voting: from machine to system. *Computer* 42(5):23-29, 2009.

[130] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, and R. Rivest. Proxy-based security protocols in networked mobile devices. In *Proceedings of the ACM symposium on Applied computing*, ser. SAC'02. ACM, pages 265–272 2002.

APPENDICES

APPENDIX A. CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Uzunay, Yusuf
Nationality: Turkish (TC)
Marital Status: Married with a son
email: yusuf.uzunay@afad.gov.tr

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Middle East Technical University Informatics Intsitute	2006
B.Sc.	Gazi University Technical Education Faculty Electronics and Computer Department	2002

AWARDS AND HONOURS

Year	Subject
2006-2007	Academic Year METU Graduate Courses Performance Award (The Highest PhD CGPA in the Department of Information Systems)
2005-2006	Middle East Technical University Best Thesis of the Year Award
2002	Honour List (Gazi University BSc Graduation)

WORK EXPERIENCE

Year	Place	Enrollment
2012-Now	T.R. Prime Ministry Disaster and Emergency Management Presidency (AFAD)	Head of Information Systems and Communication Department
2011-2012	Ministry for EU Affairs Centre for EU Education and Youth Programmes	Head of IT Department

	(Turkish National Agency)	
2007-2011	T.R. Prime Ministry State Planning Organization Centre for EU Education and Youth Programmes (Turkish National Agency)	Head of IT Department
2006-2007	T.R. Prime Ministry State Planning Organization Centre for EU Education and Youth Programmes (Turkish National Agency)	IT Expert
2002- 2006	Ankara Police Department Computer Unit	Chief of Internet and Programming Departments

FOREIGN LANGUAGES

English, German

PUBLICATIONS

1. Yusuf Uzunay, Kemal Bicakci, Trust-in-the-Middle: Towards Establishing Trustworthiness of Authentication Proxies using Trusted Computing, in submission to IEEE Transactions on Computers, 2013.
2. Yusuf Uzunay, Kemal Bicakci, "Trusted3Ballot: Improving Security and Usability of ThreeBallot Voting System using Trusted Computing", in Proceedings of ISMS2014: 5th International Conference on Intelligent Systems, Modelling and Simulation, IEEE, Computer Society, Langkawi, Malaysia, 2014.
3. Ahmet Efe, Yusuf Uzunay, Guverhan Tascioglu, "An E-government Information System Framework: Turkish National Agency Software Project", Proceedings of EGOVSHARE2009: International Conference on Egovernment Sharing Experiences, Vol:1, pp 105-138, December 2009
4. Yusuf Uzunay, Kemal Bicakci, "SHA: A Secure Voice Activated Smart Home for Quadriplegia Patients", International Workshop on Knowledge Discovery and Management in Health Informatics in conjunction with the IEEE International Conference on Bioinformatics and Biomedicine, Silicon Valley, USA, November 2007
5. Didem Gökçay, Şeref Arıkan, Yusuf Uzunay, 'Investigating behavioural problems in text-based communication by deriving an analogy between brain damage and media richness', ICANN 2007, 'Workshop on What it Means to Communicate', 2007, Porto
6. Didem Gokcay, Seref Arikan, Yusuf Uzunay, "Does computer mediated communication create people without amygdala", COGNITIVE IV:

International Cognitive Neuroscience Symposium Marmaris Turkey, May 2007

7. Yusuf Uzunay, Davut Incebacak, Kemal Bicakci, "Towards Trustable Digital Evidence with PKIDEV: PKI based Digital Evidence Verification Model", Proceedings of the 2nd European Conference on Computer Network Defence (EC2ND)", in conjuction with the First Workshop on Digital Forensics and Incident Analysis, Springer London, United Kingdom, December 2006
8. Yusuf Uzunay, "Design and Implementation of an Unauthorized Internet Access Blocking System Validating the Source Information In Internet Access Logs", M.Sc. Thesis submitted to Graduate School of Informatics Middle East Technical University (METU), Ankara Turkey, September 2006
9. Yusuf Uzunay, Kemal Bıçakcı, "UNIDES: An Efficient Real-Time System to Detect and Block Unauthorized Internet Access", 1st International Workshop on Security in Networks and Distributed Systems (SNDS 2005) in conjunction with 11th International Conference on Parallel and Distributed Systems (ICPADS 2005). IEEE, Computer Society, July 2005, Fukuoka, Japan
10. Yusuf Uzunay, Kemal Bıçakcı, "A3D3M: A PKI Based Digital Evidence Verification Model", ABG2005: National Network and Information Security Symposium, June, 2005, İstanbul, Turkey
11. Yusuf Uzunay, Mustafa Koçak, "Child Pornography on Internet and the Difficulties in Combating", Turkish Journal of Police Studies, Vol:7 (2), 2005, pp. 97-116, Ankara, Turkey
12. Yusuf Uzunay, "Network Forensics", Computer Forensics Workshop'05, May 2005, İzmir Turkey
13. Yusuf Uzunay, Mustafa Koçak, "Digital Evidences in the Domain of Cyber Crime", AB'05 Academic Informatics Conference, February 2005, Gaziantep, Turkey
14. Yusuf Uzunay, "Digital Evidence Investigation Process", The 2.nd Police Informatics Symposium, April, 2005, Ankara, Turkey
15. Yusuf Uzunay, "Digital Attacks, It's Importance to Security Forces and Ways of Protection", Turkish Journal of Police Studies, Vol:5 (2) 2003 Page:131 Ankara, Turkey

HOBBIES

Table Tennis

TEZ FOTOKOPİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü ☐

Sosyal Bilimler Enstitüsü ☐

Uygulamalı Matematik Enstitüsü ☐

Enformatik Enstitüsü ☐

Deniz Bilimleri Enstitüsü ☐

YAZARIN

Soyadı : UZUNAY
Adı : YUSUF
Bölümü : Bilişim Sistemleri

TEZİN ADI (İngilizce) : INCREASING TRUSTWORTHINESS OF SECURITY CRITICAL APPLICATIONS USING TRUSTED COMPUTING

TEZİN TÜRÜ: Yüksek Lisans ☐ Doktora ☐

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın. ☐
2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullanıcılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐
3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

Yazarın imzası ..... Tarih .....