Sudan University of Science and Technology

College of Graduate Studies

Transient Authentication Using Mobile Security Token

التحقق العابر بإستخدام مفتاح الحماية المحمول

Thesis submitted in partial fulfillment of the requirements for the degree of M.Sc. in Network and Computer Engineering

Prepared by:

Hassan Abdullahi Salih

Supervised by:

Dr. Rania Abdelhameed

November 2017

الآية

بسم الله الرحمن الرحيم

اقرأ بإسم ربك الذي خلق (1) خلق الإنسان من علق (2) اقرأ و ربك الأكرم (3) الذي علم بالقلم (4) علم الإنسان ما لم يعلم (5).

صدق الله العظيم

سورة العلق

ABSTRACT

Implementing strong, fast and friendly authentication scheme is very important for securing devices (mainly mobile devices) and information systems. There is inversive proportion between usability and security. Designing authentication scheme solving a tension between security and usability will lead to be acceptance by users which are worry in easy and fast access to system resources more than it be strong secure.

Transient authentication rises as idea to get strong authentication mechanism merged with handled token to make authentication mechanism usable.

In this project authentication scheme with two factors is implemented. Password plus hardware token work in idea of transient authentication. Token communicate with device over Bluetooth. Information sent over Bluetooth encrypted by RSA algorithm.

The prototype implemented show how the authentication scheme proposed above is secure and usable with negligible burden on user.

المستخلص

لحماية أجهزة الكمبيوتر (تحديدا الأجهزة المحمولة) و انظمة المعلومات من المهم تطبيق آلية للتحقق قوية وسريعة و صديقة (سهلة الإستخدام). هنالك علاقة عكسية بين سهولة الإستخدام و الأمان. تصميم آلية للتحقق لمعالجة التناقض بين الأمان و سهولة الإستخدام يقود لقبوله من قبل المستخدمين اللذين هم مهمومون بسهولة الاستخدام و السرعة الوصول للنظام بدرجة اعلى من قوة الامان له.

ظهر مفهوم التحقق العابرة للحصول على الية قوية للتحقق. يدمج مفهوم التحقق العابرة مع المفتاح المحمول لجعل الية التحقق سهلة الإستخدام.

في هذا المشروع طبقنا آلية للتحقق من عاملين. كلمة السر بالإضافة للمفتاح المحمول يعملان مع مفهوم التحقق العابرة. يتخاطب المفتاح مع الجهاز عبر البلوتوث مع تشفير المعلومات المرسلة بإستخدام خوارزمية التشفير RSA.

النموذج الذي طبق أوضح كيف أن الآلية المقترحة للتحقق آمنة و سهلة الإستخدام مع عبء طفيف على المستخدم.

ACKNOWLEDGEMENT

I wish to express my profound gratitude to my supervisor Dr. Rania Abdel Hameed, for his valuable guidance, continuous encouragement, worthwhile suggestions and constructive ideas throughout this research. His support, pragmatic analysis and understanding made this study a success and knowledgeable experience for me.

Hassan

November-2017

DEDICATION

To my parents,

Whom they sacrificed their own welfare for me

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACL	**Asynchronous connection-less**
ATM	**Automated Teller Machine**
btspp	**Bluetooth Serial Port Protocol**
DOS	**Denial Of Service**
FNMR	**False Nonmatch Rate**
FMR	**False Match Rate**
GPS	**Global Position System**
GSM	**Global System for Mobile**
HF	**Hash Function**
ID	**Identifier**
IEEE	**Institute of Electrical and Electronics Engineer**
LABM	**Laptop-user Authentication Based Mobile**
L2CAP	**Logical Link Control Adaptation Protocol**
MAC	**Media Access Control**
MD5	**Message Digest 5**
M.I.T	**Massachusetts Institute of Technology**
MS-DOS	**Microsoft Disk Operating System**
NIST	**National institute of Standards and Technology**
NSA	**National Security Agency**
OS	**Operating System**
PC	**Personal Computer**
PIN	**Personal Identification Number**
PKC	**Public Key Cryptography**
PW	**Password**
RDBMS	**relational database management system**
RF	**Radio Frequency**
RFCOMM	**Radio Frequency Communication**
RSA	**Ron Rivest, Adi Shamir And Leonard Adleman**
SAM	**Security Account Manager**
SDP	**Serial Discovery Protocol**
SHA	**Secure Hash Algorithm**
SKC	**Secret Key Cryptography**
SMS	**Short Message Service**
SQL	**Structured Query Language**
UDS	**Usability-Deployability-Security**
USB	**Universal Serial Bus**
ZIA	**Zero Interaction Authentication**

Chapter One

Introduction

1.1 Preface

Authentication is the process of positively verifying the identity of a user or to establish proof of identities. The main result of authentication process is ensuring that system resources will be gain by author user.

For many years' password has been the main method for user authentication whether for web authentication or standalone device [3]. In our lives using of mobile devices like laptops and mobile phones which we use to store sensitive data or do sensitive operations is increase every day according to which it provides high level of convenience and flexibility[8]. Due to the personal characteristics the user always stores huge amount of sensitive data and information in these devices (mobile devices) like passwords, credit cards numbers, personal information and others.

The nature of mobile devices result that they are may be used in hostile environment and they will be susceptible to theft and lost. Using password only not enough to give strong security because users usually chose passwords short and easy to guess as a result of they need memorable passwords. Other problem raises with password when used to secure mobile devices which used in frequent way, recurrent entering password will be disturbing and may be lead user to disable the authentication mechanism, also malicious person can monitor user when he typing the password.

To solve these problems two things was proposed, the first one is Transient Authentication [1,2] which means that the operation of authentication done frequently to insure that user present, the second password vulnerability can solve by adding second factor, here is Token (something you know "password", something you have “token"). Token used to make authentication on behalf of user to increase the usability. The consequential thing token must be small and convenient so user hold it with him always and the cost it will play main role of acceptance of it by user

1.2 Problem Statement

Password based system security become weak on securing any system. The system can easily be attacked. Useable and strong authentication mechanism is very important, cost of mechanism and overhead on system resources is consequential for the system at all to be acceptance. If the authentication becomes transparent to the user, it will lead to avoid disabling the mechanism by the user.

1.3 Propose Solution

A handheld token based system is proposed here to handle the authentication mechanism with the primary devices such as PCs laptops and entry systems [1,2,6,7,8,9,11]. The token will communicate with the primary device(s) with short wireless connection to provide a transient authentication. All communication will be secured based on short range connection security methods. Primary devises should be able to detect the absence of the token is and takes the required steps to forbid the access and secure itself immediately.

1.4 Aim and Objectives

The main aim of this research is to implement transient authentication method using a handheld device as a security token based on near-to-zero interaction authentication scheme.

The Objectives of this research are:

- To implement mutual authentication method between the token and the primary system.

- To minimize the cost the transient authentication system will be implemented using a mobile device as a security token.

- To implement secure communication between the token and the primary device.

- To make authentication mechanism friendly, securely, fast and transparent for user.

1.5 Methodology

Due to that mobile devices become invasively and its computing resources and mostly present with user we use it as security token, when it became in range with PC device authentication mechanism is running between the token (we use laptop) and the PC device to unlock it with near-to-zero user interaction to implement Transient Authentication. RSA algorithm is used to secure communication between token and PC device [14,15]. To get mutual authentication token send random number encrypted to device. The device decrypts it and replay hash value of it after concatenating with PW. The scheme strong depends on two things: firstly, one-way hash function (SHA-1) [12] Secondly the RSA algorithm. Password kept secure in PC device using one-way hash function (SHA-1). The scheme contains two phases:

1. Registration phase which done once time.

2. Login and authentication phase. It done every time user wants to get access resources and then run continuously with no user interaction.

The login and authentication phase triggered by user and then it done frequently between the token and the mobile device to insure that user is present. Bluetooth will be used because the short wireless range is important in the operation so if the user is go far device lock itself also Bluetooth consume less power and it became available in most mobile devices.

The network between token (client) and device (server) using Bluetooth called piconet. Many protocols will be use like SDP (Service Discovery Protocol), L2CAP (Logical Link Control Adaptation Protocol) and RECOMM [13].

In case token is lost the device can be unlock by using traditional method (password) which can be now tall and complex to give strong password difficult to crack [8].

1.6 Research Layout

Chapter One Introduce the general overview, the problem that solved by this research and the objectives that will going to be achieved. Chapter Two contain a Literature review and explains related works. Chapter Three is about methodology and steps and tools used in this research to achieve research objectives. Chapter Four presents the implementation procedure. Chapter Five include the Conclusion and Recommendations.

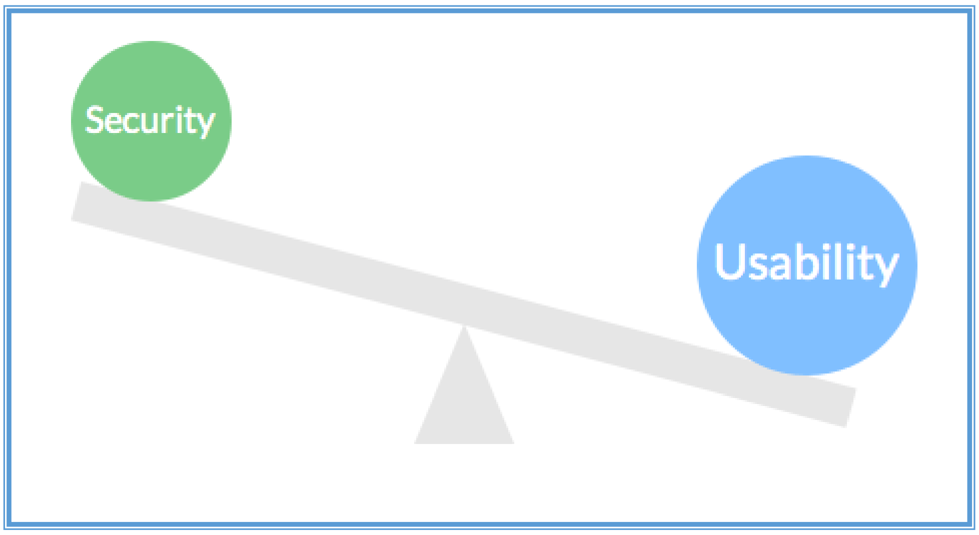
Chapter Tow

Literature review

2.1 Overview

Authentication is a mechanism to establish proof of identities [6]. It ensures that system's resources are not obtained fraudulently by illegal users [5]. When somebody want to get access to the system resources he must go on steps so that system ensure that he has right to get access. These steps and its complexity effect three factors used to evaluate authentication scheme [4] which are: (1) Security. (2) Usability from user point of view. (3) Deploying of authentication scheme. Any one of these factors redounds the other two factors. For example, strong security authentication scheme can lead it to become less in usability or can increase the cost and this effect deploying of the authentication scheme. Substantial research has been conducted in developing sophisticated security methods with authentication mechanism placed in the front line of defense, since these mechanisms are based on user conduct, they may not accomplish the intended objectives with improper use. Despite the influence of usability, little research has been focused on the balance between usability and security in authentication mechanisms when evaluating the effectiveness of these systems [16].

In these days using of mobile devices increase rapidly because it provides users a high level of convenience and flexibility [7]. The mobile devices design to be portable so they are small and light weight. Users always store sensitive and personal data and information in it (password, credit, card number …etc.). According to that securing mobile devices become a problem for users because the mobile devices are vulnerable to theft and loss as results of they were used outdoor and some time in hostile environment. To enhance system security sometimes encryption was used to save data from being got by unauthorized users. Attacker can get chance when user decrypt data and go far away from his device for any reason or as it is rampant user select weak password as it is traditional way for user authentication.



www.smashingmagzin.com/2015/12/passpharses-more-user-frindly-password 19-November-2017

Figure 2.1: Usability & Security Relation

Users displeasure to security system where authentication is part of it and this feeling increase when forcing him to authenticate himself as a result of needing to ensure that is user is still use the mobile device and the nature of mobile device which are used recurrently. Mostly users disable or circumvent security system let device and data susceptible to attack and compromise [6].

The most popular authentication scheme is password based authentication. But it became weak on secure digital identities and less usable exactly when force user to select strong password (tall and complex) or force him to re authenticate frequently. This lead researcher to propose other schemes such as wearable token security and biometric authentication schemes or combining of scheme like ZIA (zero interaction authentication) [1,2].

2.2 Persistent and Transient Authentication

Most system use what we call persistent authentication, in this type when user authenticate himself to device he gets access to system resources. Even if user goes far away from device, still everybody can use it. The other serious thing if device stolen when it is unlocking by author user. Some systems like windows [1] and phones systems if no user interaction on keyboard or mouse for period of time device lock itself. This enhance system security but still not enough "security requires frequent authentication" [1]. With persistent authentication if malicious person gets access to device he will get all rights of legitimate user. To solve this problem transient authentication was proposed in ZIA [1,2] and used in [6,7]. With transient authentication user hold or wear small security token shifts the problem of authentication from user so it done frequently. But user authenticates himself infrequently to token, this increase security for device with negligible effect on usability.

Token must be small and light weight and present with user to be less susceptible loss or theft. Token communicate with device using secure short wireless link so encryption was used [1,2,6,7]. Rang of wireless link is important in transient authentication ZIA used IEEE 802.11b when (LABM) used Bluetooth [6]. In [11] mobile phone used as a token and it use GSM network to communicate with device for web authentication.

2.2.1 Authentication types

Users authenticate themselves using one of three kind of authentication options [3,8,9] or combination of them, these option are:

1. Something you know, password is the most popular example of it. The operant of this kind depends on keep password secrecy or obscurity [3].
2. Something you have, like security token. It characterized by physical possession. It removes the burden of remembering password from user but add addition cost and needing of holding by user. Usually it combines with another factor due it vulnerable to theft or lost.
3. Something you are, characterized by uniqueness to one person. Biometric like fingerprint, eye scan, voiceprint, etc. or behavioral characterize [3,4]. It depends on it is difficult to copy. It not adds burden on user to carry anything but it adds cost as a result of need reader to authentication system.

Biometric categorized as physical or behavioral. Physical types like fingerprint, face, iris and hand. The behavioral type examples are handwritten signature and gait. User becomes inconvenient with biometric error. Biometric errors can produce as a result of many reasons (buggier of capture device or not adjust well or change on light and others). False nonmatching rate (FNME) and False match rate (FMR) [3] make authentication system or scheme depend on biometric (something you are) unfeasible. In [4] biometric was marked as more usable than hardware token but less in security for example facial recognition or fingerprint scanning can be circumvented by using a simple photo-graph or fake fingerprint [8]. With biometric nothing to carry and it is memory effort less also it is easy to learn. On the other hand, token get full mark in security in "UDS" (Usability-Deploy Ability-Security) it is more secure than password and biometric due to that token let users to use long passcode [4].

Although of the password in [4] is usable, this is according to the case password used for it. In [1,2,6,7,8] for mobile devices they assume security need frequent authentication and here password will become less in usability**.**

2.2.2 Password authentication

There are several physical means by which you can provide your authentication credentials to the system. The most common (but not the most secure) is password authentication. For many decades, password has been the standard means for user authentication on information systems including single word, phrases and personal identification number (PIN) and other. Password must have kept secret [3]. It is one of the simplest and most convenient authentication mechanisms [5]. Most of systems or internet applications use password authentication, user has an identifier (ID) and password (PW). If he wants to login to system, he prints down his ID and PW. The password is checked against a database that contains all authorized users and their passwords. Password have a higher key space [3], key space is a range of different possible values of a key. Most system kept password secure using one-way hash function (ex: MD5, SHA1). Password as an authentication scheme is simple and has negligible cost, it has no need for special device. In general password is more usable comparing to other authentication schemes it is usually succeeding when user want to login [4]. It is easy to learn because it is common in use for long time also it is easy to recovery if password is forgetting. Password is deploying more than any other authentication schemes.

A password is more usable as in [4] but when transient authentication is implemented to increase security password become unusable. A tension was found between security and usability exactly with mobile devices. Security experts focus on security and care less on usability and visa verse usability experts tend to be optimistic about security (figure 2.1 illustrate this) [4,16].

2.2.3 Password Authentication Drawback

“Over forty years of researches have demonstrated that passwords are plagued by security problem” [4]. It is hated by users, now one person can have many accounts in many application and devices every one of them should have password make user save a group password in his mind. To preserve the security of the networks and devices, passwords must be “strong” that is they should contain a combination of alpha and numeric characters and symbols. They should not be words that are found in a dictionary, and they should be relatively long (eight characters or more). In short, they should not be easily guessed. The operation of authentication using password if it became frequently it will make user fazed and can lead him to disable or eking the authentication mechanism.

Human have difficulty remembering multiple passwords (figure 2.2), as result user decide to choose short and memorable passwords which it is easy to guess by attacker or use one password for many accounts.

www.123rf.com/photo_17765960_young-man-has-confusion-in-his-head 19-November-2017

Figure 2.2: Passwords-Remember

Password guessing is one of the most common security attacks. Some systems forced users to choose long and complex password and this can lead them to write it down make it susceptible to theft. Password studies show that users choose poor passwords (Purdue 1992, Klein 1990), many approaches used to counter this problem which are:

1. Password education, use policies and good user education so user generates strong password.
2. Passwords computer generated, let computer generate strong password in behalf of user and user uses it. The shortcoming is that password usually will be unmemorable for user.
3. Password reactive checking, user select his password freely and use it then the computer run guessing tool as an attack do. If the password is guess password is disabled immediately. Good dictionaries must be used.
4. Password proactive checking, let user select his password freely and before he uses it computer verify it is acceptable

Other attack on password is an attacker can get the password by monitor user when he enters the password and use it later. Replay attack arises when a password sends over network an attacker can capture it and use it later. A more sophisticated attack when attacker personates the user's system and user enter his password to it. Commonly compromise detection for password is week [3], compromise detection is the ability of user to know if someone accesses his account or system using his identity.

If verification table is stored inside computer or system, there is likelihood to be stolen and modify by attacker such as adding new account or modify existing account and prevent legitimated user from access his account. Researchers develop authentication schemes trying adopts benefit of password authentication scheme and avoids the shortcomings of it.

2.2.4 Authentication Schemes

In [4] many schemes used for web authentication were evaluate using Usability-Deploy Ability-Security framework. The schemes are proposed to replace password. Most schemes do better than password on security. Pairs of schemes that complement each other well in a two-factor arrangement might be those where both achieve good scores in usability and deploy ability and at least one does so in security. Combined scheme might be viewed as having the AND of the usability-deploy ability scores and the OR of the security scores. In [17] they look at fifteens password schemes and its usability and security level. It seems the relation is linear between usability and security.

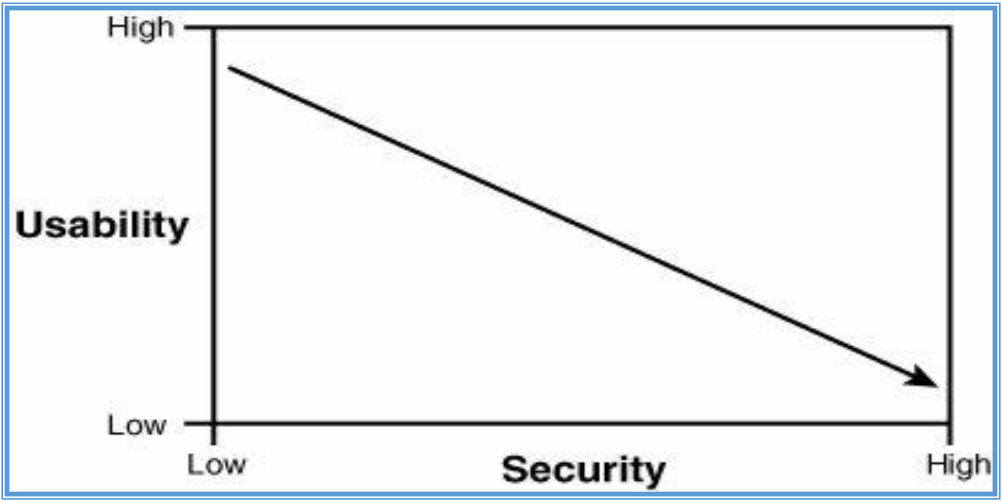


Figure 2.3: Usability vs Security

To determine the best scheme for specific case a trade-off must be done. Some benefits become less important than other according to where scheme will be implement.

In [5] they put ten requirements to evaluate the authentication scheme use password and smart-card over insecure networks is it strong or not, which are:

1. The passwords or verification tables are not stored inside the computer.
2. The password can be chosen and changed freely by the owner.
3. The password can't be revealed by the administrator of the server.
4. The passwords are not transmitted in plain text on network.
5. No one can impersonate a legal user to login the server.
6. The scheme must resist the replay attack, guessing attack, modification attack, and stolen-verifier attack.
7. The length of a password must be appropriate for memorization.
8. The scheme must be efficient and practical.
9. The scheme can achieve mutual authentication. Not only can server verify the legal users, but users can verify the legal server.
10. The password cannot be broken by guessing attack even if the smart card is lost.

2.2.5 Combining Schemes

Most of two factor authentication solutions combine "something you know" usually password and "something you have" security token and less combine token and "something you are" biometric. Password and biometric do not combine because always biometric used to avoid using password, according to fact that password is heated by users. When system designed where it is critical that the person gaining access is authorized person biometric is a reasonable choice [3], it should combine with a token. Token is a good choice when user needs to remember multiple passwords. It shifts the burden from user to the token make him more convenient. Best authentication scheme must be more usable and secure in the same time, this trouble for authentication scheme designers. Cost of scheme per user is more important and cost of administration must be toke in account. Password is still dominant on other authentication methods.

2.3 Security Token Device

Physical key like car's key can call as security token. If it is lost or theft it enables its holder to drive the car, this why digital tokens combine with other factor (password or biometric or both). Token must be less vulnerable to theft or lost. It must be temporo with user and be negligible to user while hold it. In [1,2] user ware small IBM Linux watch. To minimize the cost of adding token to authentication system, in [6,9] mobile phone was proposed to use as a security token. Most of people get mobile phone in these days and they hold their mobile phones most of time. Smart mobile phones have wireless link like Wi-Fi and Bluetooth and others, it become powerful in processing.

Smartcard is other kind of token; it is wide in use. In [5] as in others smartcard is used plus password to get strong authentication scheme.

Recovery solution when token lost is more important. Token give a good compromised detection when it lost user immediately take steps to disable it. Token authenticate to system by insert it to reader or over network (wire or wireless). In windows as an example bit locker can use USB flash as security token to hold encryption key. Authentication over network result risk of violates security of system. Securing link between token and device must be taken seriously. Token authenticate itself to system but if the system also authenticate itself to the token mutual authentication is got.

2.3.1 Secure Authentication with Token

Authentication system using token has weaknesses can be used to infiltrate it, in other words attacker to violate authentication system could attack it in three places: (1) Token. (2) Device. (3) Network link when token authenticates over network link "side channel attacks" [7].

An attacker can eavesdrop to the link to get the user identity prover. Authentication schemes use tow ways to secure link between token and device, encryption is popular method using symmetric or asymmetric algorithms. Other way is one-way hash function. Hash function is differed from encryption because hash function does not decrypt and it produce always value has a same size. Hash algorithms are deemed the most energy efficient among cryptographic algorithms [8]. Even password does not store in clear text, it stored after it hashed also discrete algorithm problem add complexity to authentication system when it used to protect authentication system [4]. Minimize wireless link network rang used for authentication as possible as prevent attacker from get identity prover or run denial of service (DOS) attack. A powerful attacker can use what we called relay attacks [10]. Here two connivances attacker's relays messages between token and device when token away from device make device deduces that token is closely to device. one of the sophisticated solutions proposed to solve relay attack problem is to use sensors of device and token like GPS, microphone, wireless network interface and so on which are equipped with most modern electronic devices. By comparing information acquisite by sensors in token and in device authentication mechanism can conclude if the token is close proximity or not even if there is relay attacks. Simple solution for relay attacks is user must switch off token when he is not present nearby device.

2.4 Cryptography

Cryptography is playing a major role in data protection in applications running in a network environment. It allows people to do business electronically without worries of deceit and deception in addition to ensuring the integrity of the message and authenticity of the sender. It has become more critical to our day-to-day life because thousands of people interact electronically every day through e-mail, e-commerce, ATM machines, cellular phones, etc. This geometric increase of information transmitted electronically has made increased reliance on cryptography and authentication by users [14].

A Fundamental rule of cryptography is that one must assume that the cryptanalyst informs the methods used for encryption and decryption. Strong of cryptography depends on key. The key can be changed as often as required. This called Kerckhoof's principle: "All algorithms must be public; only the keys are secret"

2.4.1 Cryptography Algorithms

Cryptographic algorithms can be classified in many ways, one of it is classified based on the number of keys that are employed for encryption and decryption. The three common types of algorithms are:

1. The Secret Key Cryptography (SKC) method uses only a single key for both encryption and decryption. The schemes are generally

categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing while block cipher scheme encrypts one block of data at a time using the same key on each block.

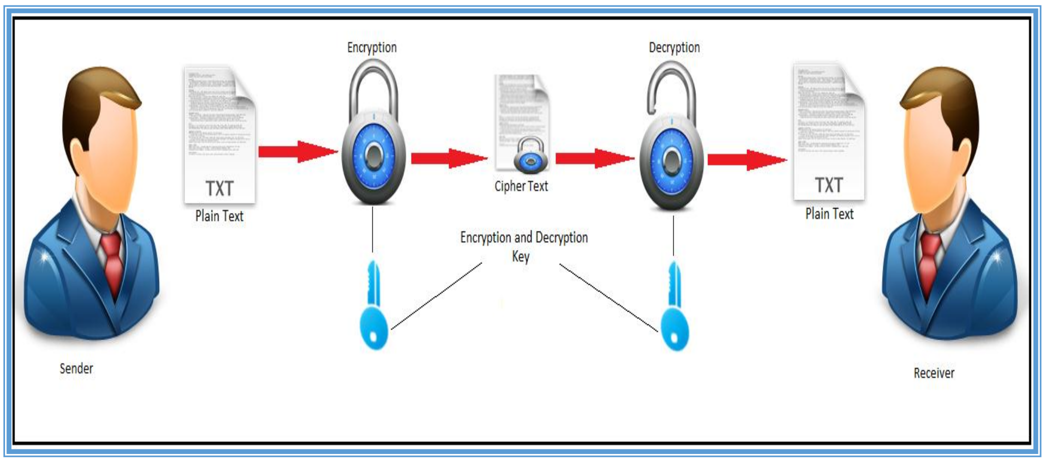


Figure 2.4: Secret Key Cryptography

2. Public Key Cryptography (PKC) scheme uses one key for encryption and a different key for decryption. Modern PKC was first described using a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key. In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. RSA is one of the first and still most common PKC implementation that is in use today for key exchange or digital signatures**.**

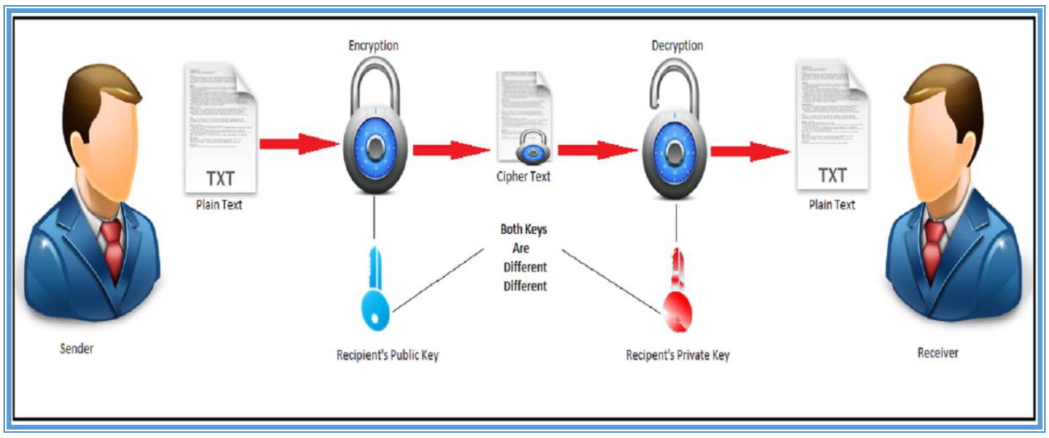


Figure 2.5: Public Key Cryptography

3. The Hash Functions (HF) uses a mathematical transformation to irreversibly "encrypt" information. This algorithm does not use keys for encryption and decryption of data. It rather uses a fixed-length hash value which computed based on a plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. These algorithms are typically used to provide a digital fingerprint of a file's content, often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords to provide some measure of the integrity of a file.

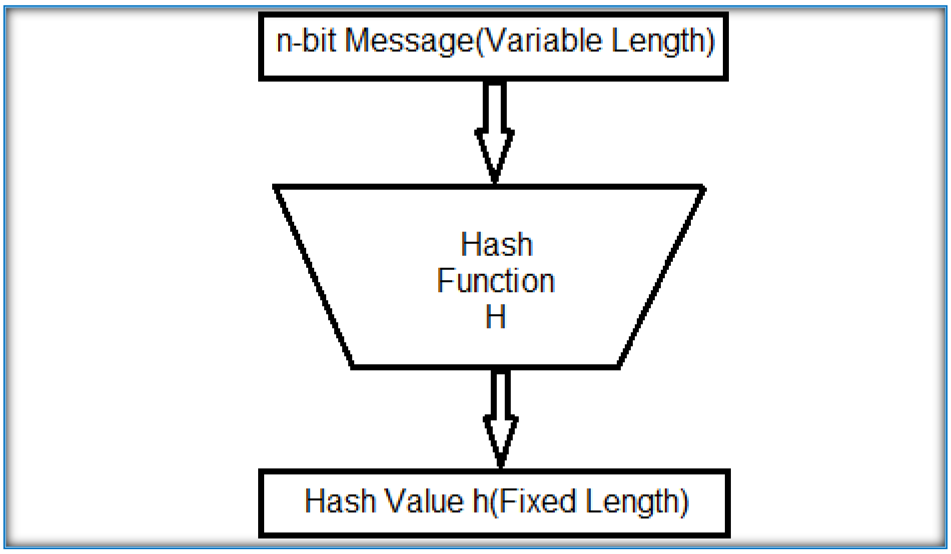
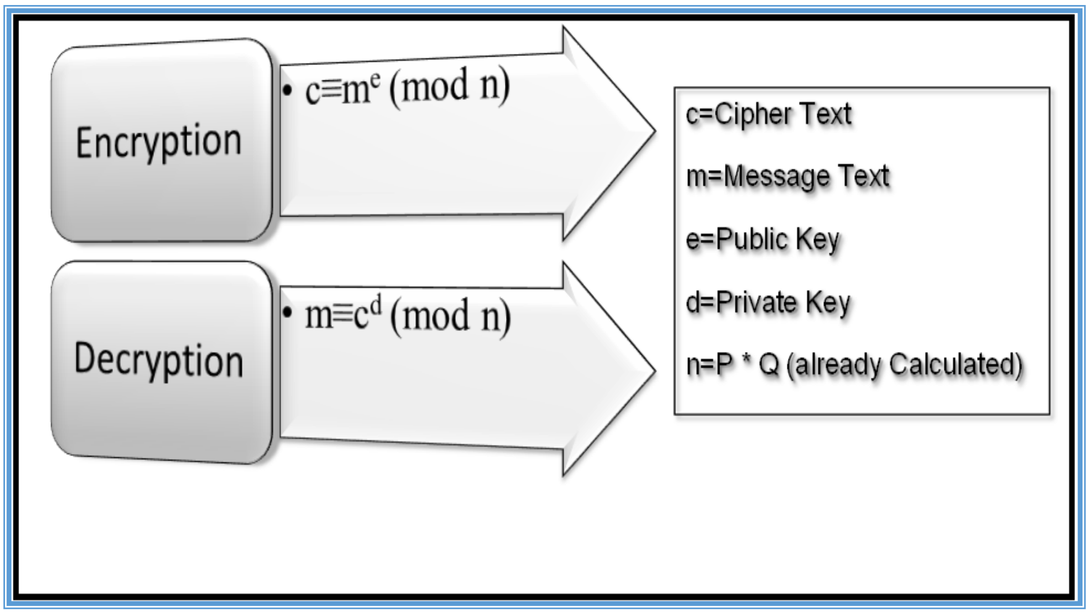


Figure: 2.6 Hash Function

2.4.2 RSA Algorithm

One good method was discovered by a group at M.I.T in 1978. It is known by the initials of the three discovers (Rivest, Shamir, Adleman) RSA. Much practical security is based on it. Its major disadvantage is that it requires keys of at least 1024 bits for good security which makes it quite slow. RSA is the first public key cryptography, and its safety is based on big integer factorization. It can be used in encryption, decryption and digital signature at the same time. Since it has been put forward in 1978, there is no serious drawback in it after it has experienced much cryptography analysis [15].



www.c-sharpcorner.com/UploadFile/75a48f/rsa-algorithm-with-C-Sharp2/ 19-November-2017

Figure 2.7: RSA Encryption and Decryption Operations

To generate RSA key firstly two numbers were selected p and q. they must be big prime numbers. As in figure 2.7 "n" is calculated.

$$\mathbf{n} = \mathbf{p} * \mathbf{q} \qquad (2.1)$$

e was chosen using the equation below:

$$\mathbf{gcd\big(e,(p-1)(q-1)\big)=1} \qquad (\mathbf{2.2})$$

“gcd” is abbreviation for grater common division. {e,n} represents a public key. Then d was chosen using this equation:

$$\mathbf{(d*e)mod\ n=1} \qquad (\mathbf{2.3})$$

{d,n} represents a private key.

2.4.3 SHA1

SAH1 is a cryptographic hash function designed by the national security agency (NSA) and published by the NIST as a U.S.Federal information processing standard. “SHA” stand for secure hash algorithm. Input to SHA1 must be less than 2^{64} bit and it will produce constant 160 bit as an output.

2.5 Related Works

The needing of enhancing authentication system lead researchers to propose varies authentication schemes. ZIA [1,2] proposed to solve tension between security and usability. ZIA not only authentication system, it is also encrypt data in hard disk [1]. Encryption key stored encrypted in device when token used to decrypt encryption key. In [2] data encrypted not only in hard disk but also in all the memory locations. In [6] Laptop-user Authentication Based Mobile phone (LABM) mobile phone used as security token. User uses his Bluetooth –enable mobile phone without combining it with encryption file system. ZIA and LABM work in concept of transient authentication. In [7] Sun et al combine authentication with encryption like in ZIA system with same idea of transient authentication. They consider their system is more simple and efficient than ZIA system with careful key management prudent communication mechanism they reduce the cost of transient authentication.

Another example of authentication with token in [8] is Token-based authentication for smartphones. Here symmetric key was used by using the Diffie-Hellmann protocol to generate key and then keep secret on smartphone and token. Key generation step must be done in secure environment. In authentication and rekeying phase SHA-256 [12] was selected as hash function to secure communication between phone and token with no mutual authentication. Every period time key is changed to prevent brute force attacks.

In [9] (Strong authentication with mobile phone as security token) concerned in avoiding using additional device in authentication mechanism in addition of basic problems of designing strong authentication mechanism. They begin from fact that strong authentication must be at least use two factors. They use password plus token, token here proposed to be mobile phone to avoid use additional device which could be inconvenience and costly for the user. The scheme used in authentication to remote server over network. To ensure that same user control two device (computer & mobile phone) closed loop begin from computer to authentication server and come back throw mobile phone using GSM network to user and end in computer. Authentication server must be connecting to GSM network. The authentication can be implementing in many ways and no encryption used here. as an example “whatsapp” do something like this by send verification code from server as SMS message to mobile phone.

Chapter Three

Methodology

3.1 System Description

Due to that mobile devices become invasively and its computing resources and mostly present with user it used as a security token, when it became in range with PC device authentication mechanism is running between the token (we use laptop) and the PC device to unlock it with near-to-zero user interaction to implement Transient Authentication. RSA algorithm is used to secure communication between token and PC device [14,15]. To get mutual authentication token send random number encrypted to device. The device decrypts it and replay hash value of it after concatenating with PW. The scheme's strong depends on two things: firstly, one-way hash function (SHA-1) [12] Secondly the RSA algorithm. Password kept secure in PC device using one-way hash function (SHA-1) [12]. The scheme contains two phases:

1. Registration phase which done once time

2. Login and authentication phase. It done every time user wants to get access resources and then run continuously with no user interaction.

The login and authentication phase triggered by user and then it done frequently between the token and the mobile device to insure that user is present. Bluetooth will be used because the short wireless range is important in the operation so if the user is go far device lock itself also Bluetooth consume less power and it became available in most mobile devices.

The network between token (client) and device (server) using Bluetooth called piconet. Many protocols will be use like SDP (Service Discovery Protocol), L2CAP (Logical Link Control Adaptation Protocol) and RECOMM [13].

In case the token is lost the device can be unlock by using traditional method (password) which can be now tall and complex to give strong password difficult to crack [8].

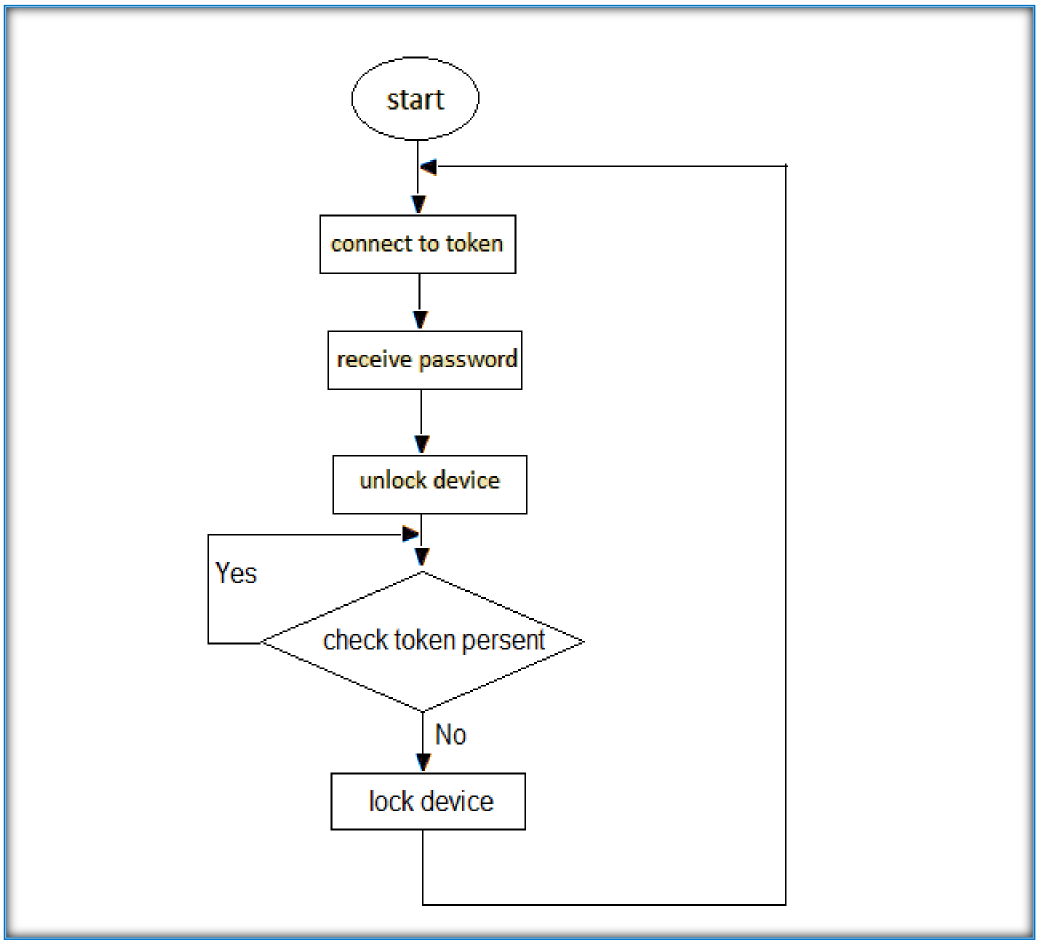


Figure 3.1: Transient Authentication Run on Mobile Device

3.2 Authentication scheme phases

Our scheme contains two main phases:

Registration phase. Which done only one time for user, it done in the device user want to authenticate himself to it (server device). Here laptop is used and we prefer it be done in secure environment avoiding malicious observing. User freely chose his password (6 character at least). This gives user an option to choose memorable password. Then Bluetooth inquiry run a here token device Bluetooth must be on and discoverable. Device show to user all devices friendly name and their Bluetooth address. User chose his token and the software generate hash value for password. "SHA1" is used. The hash value and the token MAC address store in the database using SQL DATABASE. If device does not generate its RSA keys it generates them. Keys do not generate every registration operation; it must run only one time. Private Key must be kept secure and public key store in token. Even the public key is not important to be kept secure but if it is not gain by third party he miss an ability to run offline attack and compare the encrypted value with encrypted value send by token to know the password. So we prefer send public key through a secure channel or given by hand.

When these steps complete the user have ability to authenticate himself with token. Now we gain two factors authentication (password & token). Not just one token can be used but more than one token can be register to allow multi users access to device.

Login and authentication phase. When user wants to access to device resources he must have a token and know the password. The phase runs in token and device in the same time. In the device side run "btspp" server (Bluetooth serial port protocol) and wait for token. In token "btspp" client runs after user insert the password which selects it in registration phase. Token runs Bluetooth inquiry and show all Bluetooth device in rang. If token becomes closely to device, it will detect it and show it in the list of devices nearby token. When user chooses the device the password sends as encrypted string text to device. The encrypted string text is generated as 256 characters' even if the password be one character due to padding used by RSA algorithm to make encryption strong and make intruder with no idea of the tall of password if he captures the encrypted string password and try offline attack.

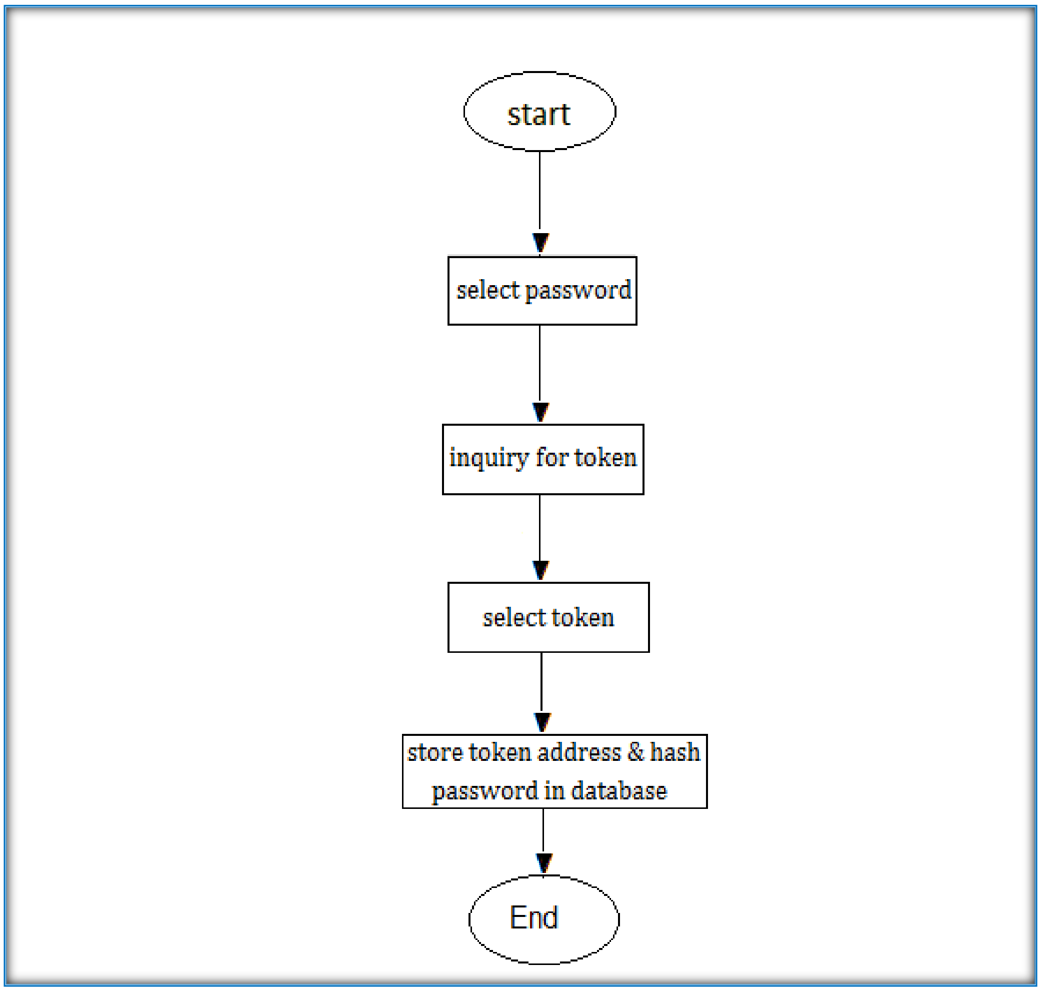


Figure 3.2: Registration Phase

In other side (device side) when request to Bluetooth connection come from token, device checks if token's Bluetooth physical address are in the database. If token was registered before the request accept and if not request is denying. If the request accepted, encrypted string password received from token and decrypts using RSA with private key. After that generate SHA1 on it to get hash value of decrypted password and compare it with hash value store with physical address with the same ID in database, if they are same, device unlock and user get access to device resources. MS-DOS commands used to unlock device.

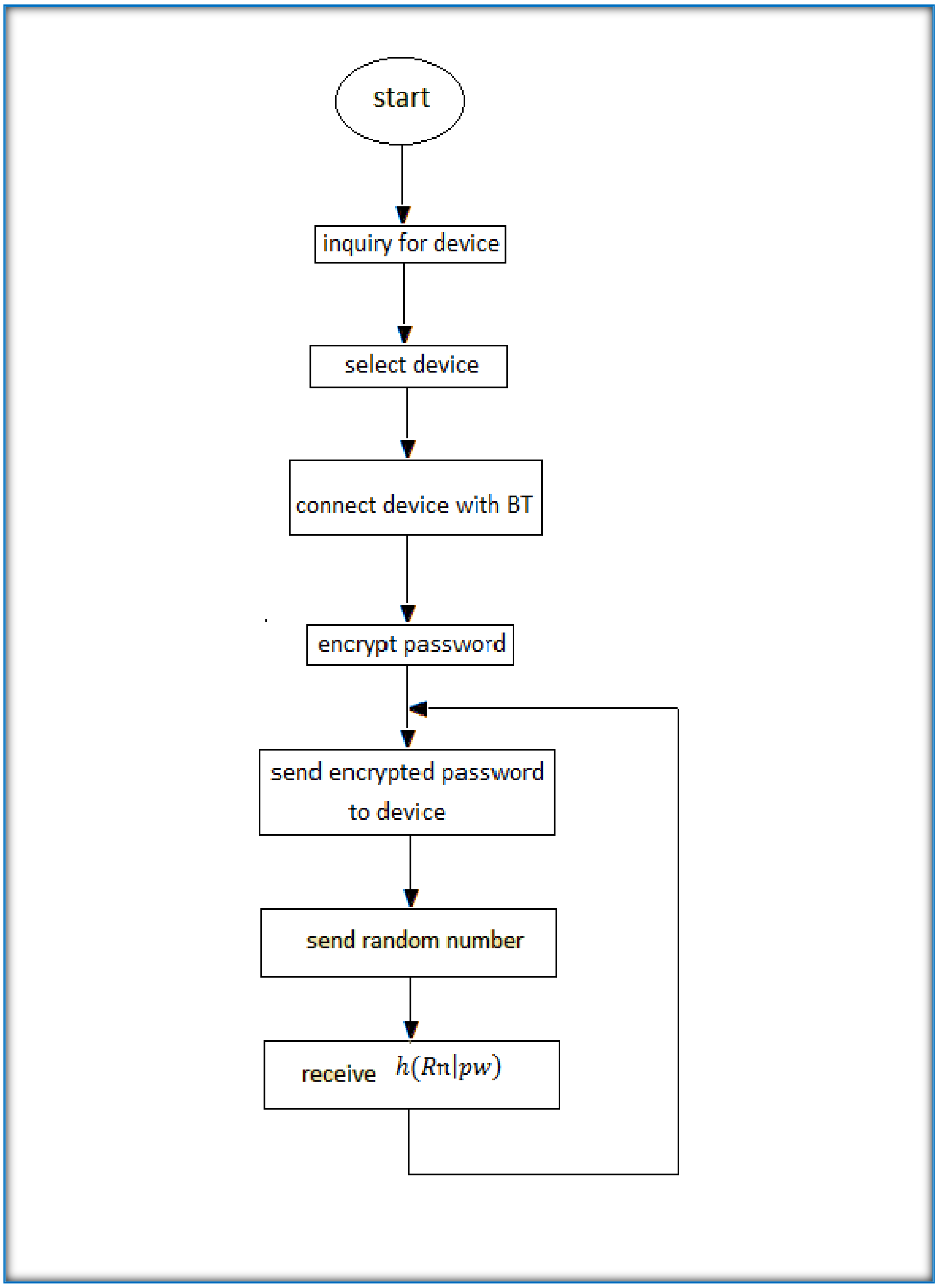


Figure 3.3: Login and Authentication Phase (token side)

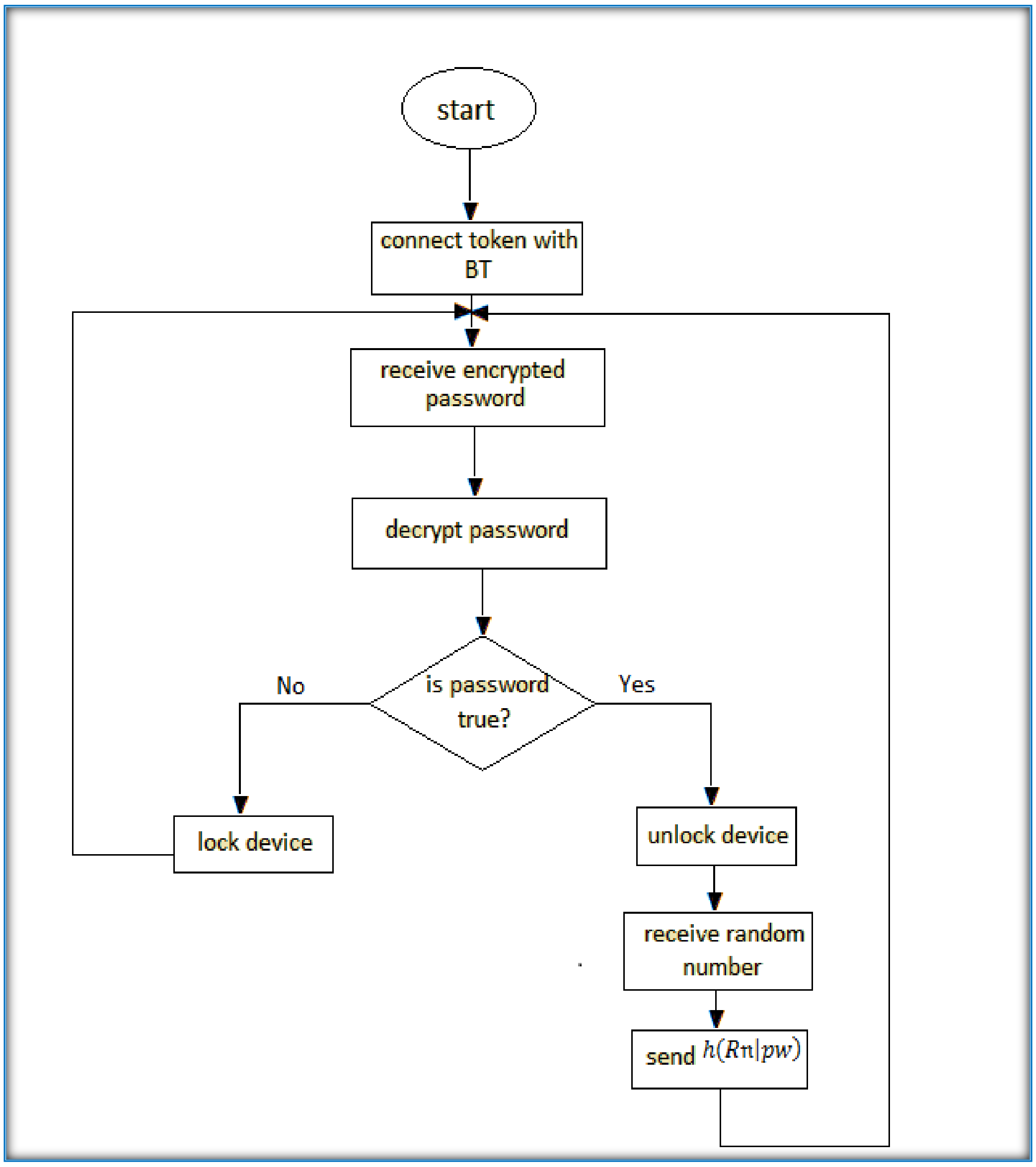


Figure 3.4: Login and Authentication Phase (device side)

To gain transient and mutual authentication token send the encrypted string password periodically to device and device do the same steps of decryption and hashing and compare with hash value of password store in database. If they are same no action to do but if they are not immediately device secures itself. Also after the token send encrypted string password to device it sends encrypted random number to it. device receive encrypted number and after decrypt it device concatenate it with password then replay the hash value of the result to token which compare it with hash value of number send before after concatenate with password given by user, this done to get mutual authentication. if other party communicate with token and impersonate the device identity token will know that and disconnect it. Now we have one complete cycle it will repeat from sending password from token again.

$$\mathbf{Token} \rightarrow \mathbf{Device}: \mathbf{E}(\mathbf{pw}) \qquad \text{———}\mathbf{1}$$

$$\mathbf{Token} \rightarrow \mathbf{Device}: \mathbf{E}(\mathbf{R}\boldsymbol{n}) \qquad \text{———}\mathbf{2}$$

$$\mathbf{Device} \rightarrow \mathbf{Token}: \mathbf{h}(\mathbf{R}\boldsymbol{n}|\mathbf{pw}) \qquad \text{———}\mathbf{3}$$

E ≡ Encryption **pw ≡ password** **R$\boldsymbol{n}$ ≡ Random number**

| ≡ concatenation

Other scenario if device miss connectivity with token as a result of user switch off token or stop run authentication mechanism or token become out of range of device's Bluetooth also device secures itself.

Once device miss connectivity with token it returns to situation before token request connection with it and wait for reconnection with token again. If user loses token he will not miss his ability to gain access to device resources, operating system build in authentication mechanism can be used to authenticate.

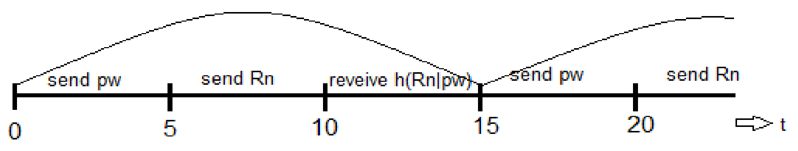


Figure 3.5: Authentication Cycle (token) t ≡ time in second

3.3 Software

To implement our authentication, scheme some software was selected to build it. Software availability and ease of use offer great chance for authentication scheme to be acceptance by user.

3.3.1 Operating System

Our authentication mechanism prototype runs on two devices. Both of them use windows as operating system. Windows dominated the world's personal computer market. It is easy to use for most computers users [11].

3.3.2 Java

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. Java language can run on all platforms. Although the execution efficiency of language is slower than that of C language and C++ language and it is said that java speed is only one tenth of C, its speed is about one third of C and C++ really as java language provides many class libraries which are implemented with native languages [15].

We use netbean ide 8.2 (integrated development environment) which is free software provides by sun Microsoft. 'bluecov' is special library give a programmer an ability to use Bluetooth adapter and send and receive data over it. Also 'mysql' special library for MySQL database. RSA algorithm and SAH1 have classes in java reduce effort of programmer to implement cryptosystem. RSA used with 2048 bit and use 'ECB/PKCS1PADDING' padding to make encryption stronger.

In the environment of electronic commerce, electronic commerce systems must run on different operating system platforms such as windows, Unix, and Linux. For example, if the system is developed with C language, it will bring up a problem that the system cannot run on all platforms although the system developed with C language is easy to transplant. As the same data type has different length on different platforms, this makes it difficult to transplant

3.3.3 MySQL

A database is a separate application that stores a collection of data. RDBMS (relational database management system) is software used to store and manage data. Data stores in different tables and relation between them using primary keys. MySQL is a fast, easy to use RDBMS released under an open-source license so no need to pay to use it. We use MySQL to create database to store Bluetooth physical address and hash password value. It seems as Security Account Manager (SAM) database in windows OS. Database used to verify user identity so keeping it secure is very important for our authentication scheme success. The database is protecting with password to make it invisible. if intruder get access to database he can modify it to access system using his own token or prevent legitimate user from access system resources or get hashed password and generate offline attack trying to get password to use it if he gets the token.

3.4 Hardware

The system prototype consist two laptops. One run as token and the other represent the device user want to access it. Each device has its Bluetooth adapter. Laptop run as token is not an ideal implementation but we use it as an example to show how we can implement transient authentication and its enhance on security and usability user get it in same time.

3.4.1 Bluetooth

Bluetooth is the name given to technology standard using short rang radio links intended to replace the cables connecting portable or fixed electronic devices. It is standard a uniform structure for wide range of devices to communicate with each other with minimal user effort. It has low complexity, low power and low cost. Bluetooth radio operate in unlicensed ism band at 2.4GHz (industrial, scientific and medical (ism) radio bands are radio bands (portions of the radio spectrum) reserved internationally for the use of radio frequency (RF) energy for industrial, scientific and medical purposes other than telecommunications). We use serial port protocol (spp) the figure 3.5 below show its model.

Bluetooth client device prefers inquiry using SDP (serial discovery protocol) to find the RFCOMM server channel. After that client request a new L2CAP channel to the remote RFCOMM and initiate an RFCOMM session on the L2CAP channel. Bluetooth server device in the other side accepts a new channel establishment indication from L2CAP and accepts an RFCOMM session establishment on that channel [13].

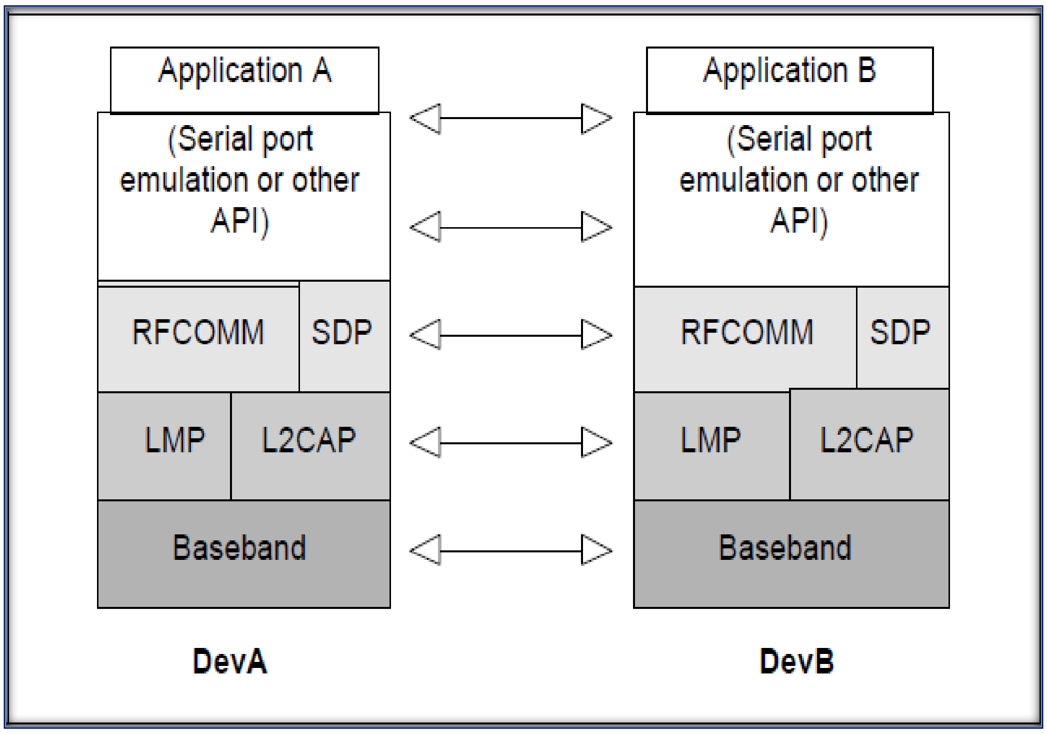


Figure 3.6: Bluetooth Protocol Model[13]

More specification about important Bluetooth protocol mentioned above will explain below:

- Logical link control adaptation protocol (L2CAP) __ it takes data from higher layer of the Bluetooth stack and from application and sends them over the lower layers. The major function of the l2cap are:
  1. Multiplexing between different higher layer protocols to allow several higher layer links to share a single ACL (asynchronous connection-less) connection. L2CAP uses channel numbers to label packets.
  2. Segmentation and reassembly to allow transfer of large packets than lower layer support.
  3. Quality of service management for higher layer protocol.

- RFCOMM __ is a simple reliable transport protocol that provides emulation of the serial cable line setting. It provides connection to multiple devices by relying on L2CAP to handle multiplexing over single connection.
- Serial discovery protocol__it provides a means for an SDP client to access information about services offered by SDP server

When Bluetooth is used continuously for send encrypted string text we notice audio run over Bluetooth affect negatively. Also decryption continuously adds delay on program execution. So encrypted password every 15 second (15 second is a period of one transmission cycle between token and device), we assume it is more sufficient to sense user existent.

3.5 prototype explication

The figure 3.6 below show how user interaction with other prototype components:

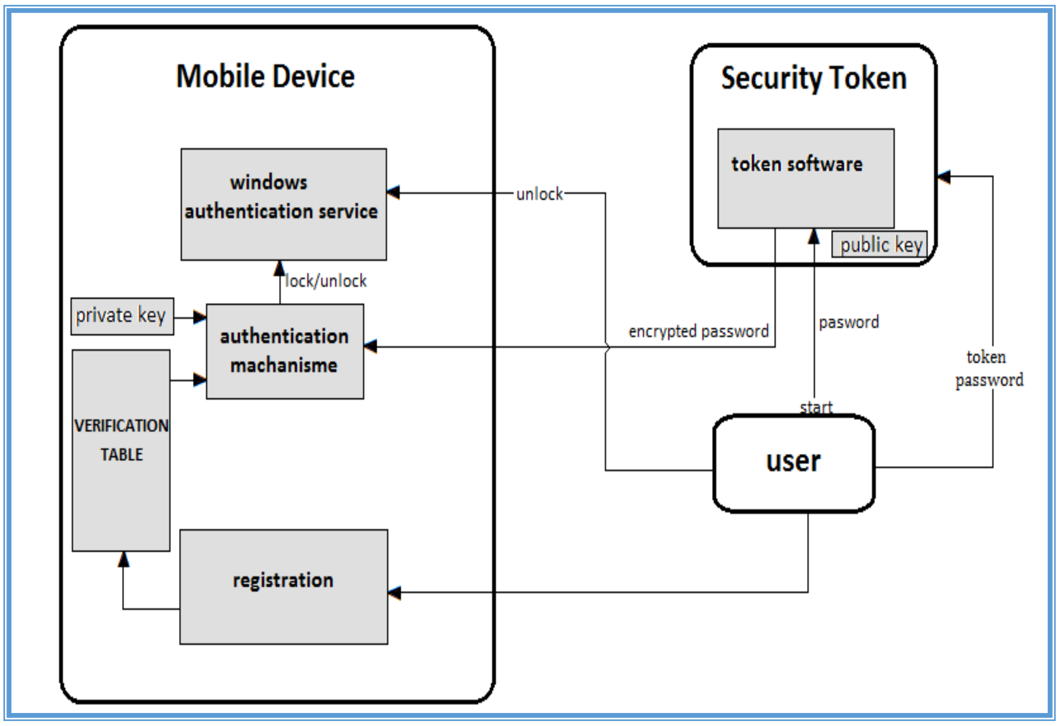


Figure 3.7: User Interaction with Other System's Components

Chapter Four

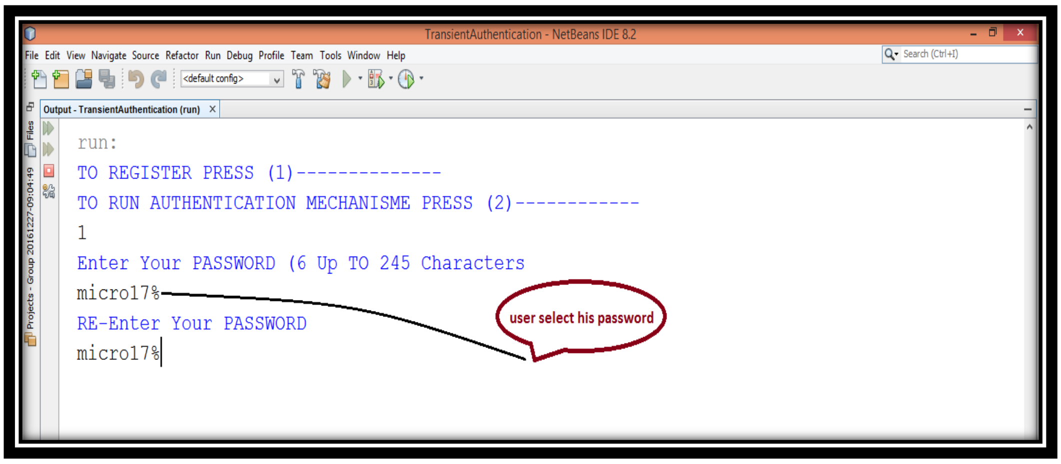
Results and Discussion

4.1 Results

After compiling the java code in both token and device the following outputs were got. In device mainly user selects one of two options. One option let user register his token. Second option runs the authentication mechanism.

4.1.1 Registration Phase

User login to device using windows password and then run registration phase which let user freely chose his password, at least six and up 245 characters (due to data type of java language that used to represent password for encryption and decryption operations).

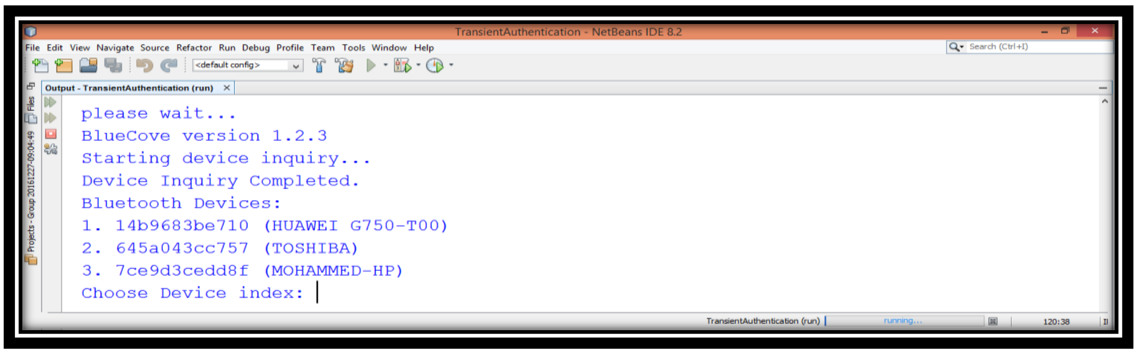


```
run:
TO REGISTER PRESS (1)--------------
TO RUN AUTHENTICATION MECHANISME PRESS (2)------------
1
Enter Your PASSWORD (6 Up TO 245 Characters
micro17%
RE-Enter Your PASSWORD
micro17%
```

TransientAuthentication - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - TransientAuthentication (run)
user select his password

Figure 4.1: User Selects his Password

Next step, device starts inquiry searching for near Bluetooth devices. Token Bluetooth adapter must be on, discoverable and in Bluetooth radio range.



```
please wait...
BlueCove version 1.2.3
Starting device inquiry...
Device Inquiry Completed.
Bluetooth Devices:
1. 14b9683be710 (HUAWEI G750-T00)
2. 645a043cc757 (TOSHIBA)
3. 7ce9d3cedd8f (MOHAMMED-HP)
Choose Device index: 
```

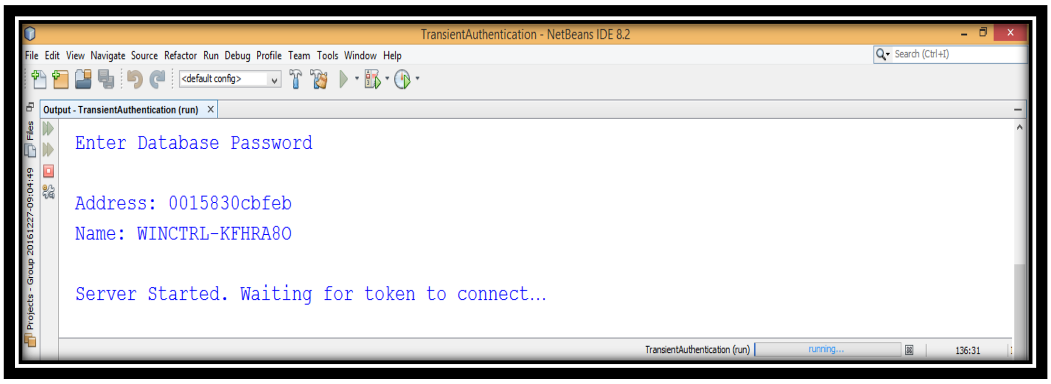
TransientAuthentication - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - TransientAuthentication (run)
TransientAuthentication (run)
running...
120:38

Figure 4.2: User Selects his Token

After user chooses the token according to its physical address and friendly Bluetooth name he must enter database password to store token physical address and password.

4.1.2 Login and Authentication Phase

Login and authentication phase run immediately after registration phase finish or from beginning by select second option in figure 4.1 wait for token connection in device side.

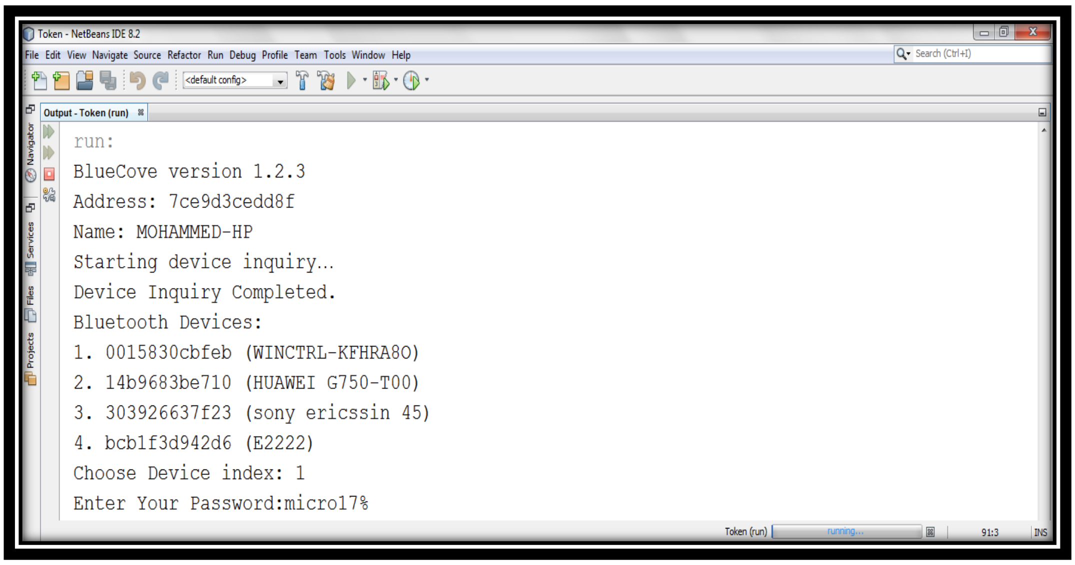


```
Enter Database Password

Address: 0015830cbfeb
Name: WINCTRL-KFHRA80

Server Started. Waiting for token to connect...
```

TransientAuthentication - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Output - TransientAuthentication (run)
TransientAuthentication (run)
running...
136:31

Figure 4.3: Device Wait for Token Connection

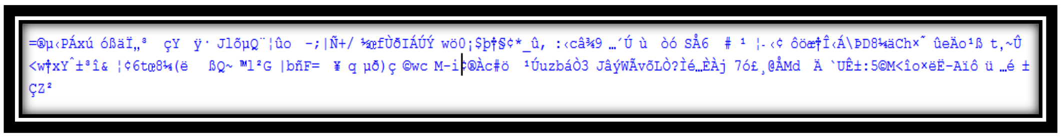
Once device waits for token to connect, user can run token program. Also token has its password must be enter before user can run token program. Token program begin with device inquiry let user select the device. Then ask user to insert his password.



```
run:
BlueCove version 1.2.3
Address: 7ce9d3cedd8f
Name: MOHAMMED-HP
Starting device inquiry...
Device Inquiry Completed.
Bluetooth Devices:
1. 0015830cbfeb (WINCTRL-KFHRA8O)
2. 14b9683be710 (HUAWEI G750-T00)
3. 303926637f23 (sony ericssin 45)
4. bcb1f3d942d6 (E2222)
Choose Device index: 1
Enter Your Password:micro17%
```

Figure 4.4: Token Run Authentication Mechanism

The figure 4.5 shows as an example the encrypted text for password (micro17%) send from token to device.



```
=®µ‹PÁxú óßäÏ„³  çY  ÿ· JlõµQ¨¦ûo  -;|Ñ+/ ¼œfÙðIÁÚÝ wö0¡$þ‡§¢*_û, :‹câ¾9 …´Ú ù  òó SÅ6  # ¹ ¦-‹¢ ôöæ‡Î‹Á\ÞD8¼äCh×˜ ûeÄo¹ß t,~Û
<w‡xYˆ±ªî& ¦¢6tœ8¼(ë   ßQ~ ™l²G |bñF=  ¥ q µð)ç ©wc M-i¢®Àc#ö  ¹ÚuzbáÒ3 JâýWÃvõLÒ?Ìé…ÈÀj 7ó£¸@ÅMd  Ä `UÊ±:5©M<îo×ëË-Aïô ü …é ±
ÇZ²
```

Figure 4.5: Encrypted Password

When connection establish between token and device and the authentication mechanism run, the device continuously sense token present. If connection failed or the password send by token not match the password in device database store in registration phase device locked itself immediately.

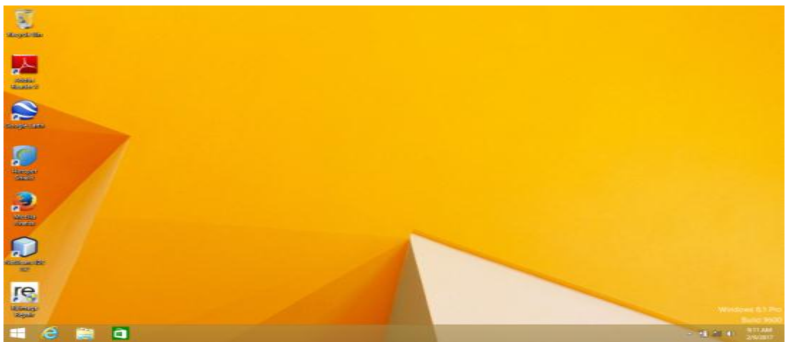


Figure 4.6: Device When Token Exist

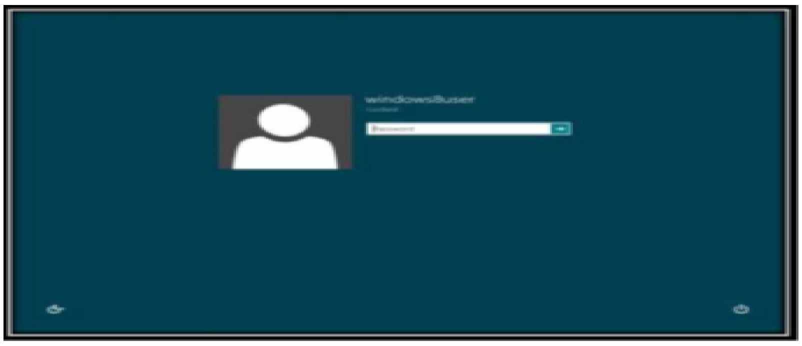


Figure 4.7: Device When Token Absent

4.2 Security Analysis

The reauthentication operation when done by token frequently enhances the security of device but some security issues raise here. “man-in-the-middle attack” is an attack where the attacker secretly relays and possibly alters the communication between two parties. For our authentication scheme attacker can works as a relay station to increase authentication range forsakes device unsecure when user is away. in our model once device lose connection with token it secures itself and even if token come back device still in secure situation unless the user re-start the token program again. So man-in-the-middle will miss the ability of increase the range of authentication.

Reply attack is the most problem of our model because if attacker gets the encrypted string send by token he can resend it if he can impersonate the token with no need to decrypt the password.

Compromise detection shows if an authenticator has been stolen before it used illegally. When token lost it give good compromise detection especially when we use device used repeatedly by user. Token has its own password must be entered before someone go ahead start authentication mechanism.

A token is a good tool to generate high-entropy passcodes from lower entropy passwords. This is evident in figure 4.5 token generate long password from short one.

When device check the password send from token according to the database to let user get access the windows password is revoked for seven second and it return again, this done by batch file contain windows password in clear text and this can get by malicious person let him access device resource with no need to token.

In our model the security was enhanced by getting mutual authentication. Malicious person and man in the middle will lose an ability to make token transmit him. Communication between token and device has burden on token power consumption. This is opposite of our goal of making authentication scheme in minimum cost.

Chapter Five

Conclusion and Recommendations

5.1 Conclusion

In this research near-to-zero interaction authentication model was designed with handled token plus password to implement transient authentication. Laptop is used as token. It used to authenticate to user's device in behalf of him via Bluetooth. Communication between token and device is secured using RSA algorithm and SHA1 hash function. We see that improving the security with negligible impact on usability. The device can be unlocked with more than one token to multiple user accounts. Also token can be used for more than one device. Most attacks can violate our scheme are shown; replay attack rises as the most problem in our scheme. Also modify verification table will lead to violate scheme security.

5.2 Recommendations

In future work points below are suggested to enhance our project:

- Use mobile phone or watch so token becomes linked with user with negligible cost.
- Enhance security by get one-time password to prevent reply attack.
- Use other tool than batch file to avoid put password in clear text.

References

[1] M.Corner and B.Noble "Zero-Interaction Authentication" MOBICOM September 2002.

[2] Anthony,Corner,Noble "Mobile Device Security Using Transient Authentication" September 2006.

[3] LAWRENCE O'GORMAN,FELLOW, IEEE "Comparing Passwords, Tokens, Biometrics for User Authentication" 2003.

[4] Bonneau, Herley, Oorschot and Stajano "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes" 2012 IEEE.

[5] Liao, Lee and Hwang "A password authentication scheme over insecure networks" Journal of Computer and System Science 2006.

[6] Rania Abdelhameed, Sabira Khatun, Borhanuddin Mohd Ali and Abdul Rahman Ramli “Authentication model based Bluetooth enabled mobile phone” 2005.

[7] D.Z.Sun, J.P.Huai, J.Z.Sun, J.W.Zhang and Z.Y.Feng "A New Design of Wearable Token System for Mobile Device Security"2008.

[8] Manuel, Matthias, Hubert and Zsolt "Token-based Authentication for Smartphone".

[9] Thanh, Jorstad, Jonvik and Thuan "Strong authentication with mobile phone as security token" 2009.

[10] Hien, Xiang, Babins, Nitesh, Asokan and Petteri "Comparing and Fusing Different Sensor Modalities for Relay Attack Resistance in Zero-Interaction Authentication" 2014.

[11] Hishameldeen, Mohammed, Mohammed Osman, Rabiee "Implementation of Distributive Authentication System Using a Security Token.

[12] NIST,Secure hash standard, Technical report FIPS 180-4 ,NIST, Department of commerce, March 2012.

[13] BLUETOOTH SPECIFICATION Version 1.1 / *Serial Port Profile.*

[14] Nentawe Y. Goshwe. "Data Encryption and Decryption Using RSA Algorithm in a Network Environment" july 2013.

[15] Guicheng Shen, Bingwu Liu, Xuefeng Zheng "Research on Fast Implementation of RSA with Java" 2009.

[16] Martin Mihajlov, Martin Mihajlov, Saso Josimovski "Quantifying Usability and Security in Authentication" 2011.

[17] Anne Wildenhain, Jeremiah Blocki, Anupam Datta, Manuel Blum "Comparison of Usability and Security of Password Creation Schemes" August 11, 2012

Appendix A

Java code for transient authentication (device side)

```
// This is progarm to implement transient Authentication (device side)
package TransientAuthentication;
import java.io.*;
import java.security.PrivateKey;
import java.util.Scanner;
import static TransientAuthentication.Encryption.PRIVATE_KEY_FILE;
import static TransientAuthentication.Encryption.decrypt;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.concurrent.TimeUnit;
import javax.bluetooth.*;
import javax.microedition.io.*;
// Class that implements an SPP Server which recive encrypted PW
public class SampleSPPServer  {
private static boolean systemLock=false;
//physical address of token
static String tokenAddress;
//hash value of password
static String hashPw = null;
// user name of database
static String userName="root";
```

```
//password of database
static String password;
//database URL
static String URL="jdbc:mysql://localhost:3306/mac";
//start server
private void startServer() throws IOException, ClassNotFoundException, InterruptedException{
//Create a UUID for SPP
UUID uuid = new UUID("1101", true);
//Create the servicve url
String connectionString = "btspp://localhost:" + uuid +";name=Sample SPP Server";
//open server url
StreamConnectionNotifier streamConnNotifier = (StreamConnectionNotifier)Connector.open( connectionString );
int v=0;
do{
//Wait for client connection
System.out.println("\nServer Started. Waiting for token to connect…");
StreamConnection connection=streamConnNotifier.acceptAndOpen();
RemoteDevice dev = RemoteDevice.getRemoteDevice(connection);
tokenAddress=dev.getBluetoothAddress();
System.out.println("Token address: "+tokenAddress);
System.out.println("Token name: "+dev.getFriendlyName(true));
 try {
```

```
            Connection           conn=DriverManager.getConnection(URL,
userName, password);
            ResultSet rs = conn.createStatement().executeQuery("SELECT *
FROM macHashpw");
            while (rs.next()){
                        if(rs.getString(2).contentEquals(tokenAddress))
                        hashPw=rs.getString(3);}
            if(hashPw==null){
            System.out.println("Sorry____Your Token Is Not Registered...
Register It befor You Can Use It.");System.exit(0);}
            } catch (SQLException ex) {
                 System.err.println("Error"+ex);}
        System.out.println("\nAuthentication begin");
        //read string from spp client
        InputStream inStream=connection.openInputStream();
        BufferedReader        bReader=new        BufferedReader(new
InputStreamReader(inStream));
        ObjectInputStream  inputStream  =  new  ObjectInputStream(new
FileInputStream(PRIVATE_KEY_FILE));
        final      PrivateKey      privateKey      =      (PrivateKey)
inputStream.readObject();
        String lineRead,plainText;
        long w=0;
        for(int y=0;y<20;){
        try{
        lineRead=bReader.readLine();
        //do decryption every 5 seconds
        if (System.currentTimeMillis()-w>=5000){
         byte[] zz=lineRead.getBytes("ISO-8859-1");
         plainText = decrypt(zz, privateKey);
        if(systemLock==true){
         if(sha1.SHA1(plainText).contentEquals(hashPw)){
              String path="cmd /c start
C:\\Users\\elham\\Desktop\\revokPW.bat - Shortcut";
              Runtime rn=Runtime.getRuntime(); rn.exec(path);
              System.out.println("System is Unlock");
              TimeUnit.SECONDS.sleep(7);
              path="cmd /c start C:\\Users\\elham\\Desktop\\setPW.bat";
              rn.exec(path);
              systemLock=false;}
                    }
        else {
           if(sha1.SHA1(plainText).contentEquals(hashPw)==false){
             String path="cmd /c start C:\\Users\\elham\\Desktop\\lock.bat -
Shortcut";
             Runtime rn=Runtime.getRuntime(); rn.exec(path);
             System.out.println("\nWrong Password, System is Locked...");
             systemLock=true;}
           }
        w=System.currentTimeMillis();}
        }catch(IOException e){
           String path="cmd /c C:\\Users\\elham\\Desktop\\lock.bat -
Shortcut";
           Runtime rn=Runtime.getRuntime();
           rn.exec(path);
```

```
          path="cmd /c start C:\\Users\\elham\\Desktop\\setPW.bat";
          rn.exec(path);
          System.out.println("\nToken Connection Failed");
          systemLock=true;
          break;}
        catch(NullPointerException ex){
          System.err.println("Token was not registerd Or Database connection faild");
          System.exit(0);}
        }//end for loop
        }while(v==0);
        streamConnNotifier.close();}
        public static void main(String[] args) throws IOException, ClassNotFoundException, SQLException, InterruptedException {
        System.out.println("TO REGISTER PRESS (1)--------------");
        System.out.println("TO RUN AUTHENTICATION MECHANISME PRESS (2)------------");
        Scanner sc=new Scanner(System.in);
         int choose=sc.nextInt();
        //registration
        if (choose==1){
          int check;
          String pw1,pw2;
          do{
            System.out.println("Enter Your PASSWORD (6 Up TO 245 Characters");
            check=0;
            Scanner sc2=new Scanner(System.in);
```

```
                pw1=sc2.nextLine();
              if (pw1.length()>245){
                check=1;
                System.err.println("Your PASSWORD More Than 245 Character..Please Choose Another PASSWORD");}
              else if(pw1.length()<6){
                 check=1;
                 System.err.println("Your PASSWORD Less Than 6 Charactors..Please Choose Another PASSWORD");}
              if (check==0){
               System.out.println("RE-Enter Your PASSWORD");
              Scanner sc3=new Scanner(System.in);
              pw2=sc3.nextLine();
              if(pw1.contentEquals(pw2)){
                 hashPw=sha1.SHA1(pw1);}
              else if(!(pw1.contentEquals(pw2))){
                 System.err.println("You Insert Tow Different PASSWORD---Repeat PASSWORD Insertion");
                 check=1;
              }
           }
                }while(check==1);
             //MAC address of token
             System.out.println("please wait...");
              tokenAddress=DeviceInquery.inquiry();
              if(tokenAddress!=null){
                 boolean insert;
```

```
        do{
          insert=false;
          System.out.println("Enter Database Password");
          Scanner sc4=new Scanner(System.in);
          password=sc4.nextLine();
      try{
        Connection          conn=DriverManager.getConnection(URL,
userName, password);
                    String sql=("INSERT INTO `macHashpw`"
                      +"values(null,"
                      +"'"+tokenAddress+"',"
                      +"'"+hashPw+"')");//save token physical address and
hashed password to database
         conn.createStatement().executeUpdate(sql);
         }catch(SQLException e){
           System.err.println(e);
           insert=true;}
        }while(insert); }
    // Check if the pair of keys are present else generate those.
      if(!Encryption.areKeysPresent()){
         Encryption.generateKey();
      }
      choose=2;
    }
  //run authentication mechanism
  if(choose==2){
  //display local device address and name
```

```
        LocalDevice localDevice = LocalDevice.getLocalDevice();
        System.out.println("Address:
"+localDevice.getBluetoothAddress());
        System.out.println("Name: "+localDevice.getFriendlyName());
        SampleSPPServer sampleSPPServer=new SampleSPPServer();
        sampleSPPServer.startServer();}}}
```

Appendix B

Java code contains methods for encryption, decryption and key generation

```
package TransientAuthentication;
import java.io.File;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;
public class Encryption {
// String to hold name of the encryption algorithm.
public static final String ALGORITHM = "RSA";
//String to hold the name of the private key file.
public static final String PRIVATE_KEY_FILE =
"C:/keys/private.key";
//String to hold name of the public key file.
public static final String PUBLIC_KEY_FILE =
"C:/keys/public.key";
/* Generate key which contains a pair of private and public key
using 2048
 * bytes. Store the set of keys in Private.key and Public.key files.*/
public static void generateKey() {
   try {
```

```
            final KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ALGORITHM);
            keyGen.initialize(2048);
            final KeyPair key = keyGen.generateKeyPair();
            File privateKeyFile = new File(PRIVATE_KEY_FILE);
            File publicKeyFile = new File(PUBLIC_KEY_FILE);
            // Create files to store public and private key
            if (privateKeyFile.getParentFile() != null){
              privateKeyFile.getParentFile().mkdirs();}
            privateKeyFile.createNewFile();
            if (publicKeyFile.getParentFile() != null){
              publicKeyFile.getParentFile().mkdirs();}
            publicKeyFile.createNewFile();
            // Saving the Public key in a file
            ObjectOutputStream publicKeyOS = new ObjectOutputStream(
                new FileOutputStream(publicKeyFile));
            publicKeyOS.writeObject(key.getPublic());
            publicKeyOS.close();
            // Saving the Private key in a file
            ObjectOutputStream privateKeyOS = new ObjectOutputStream(
                new FileOutputStream(privateKeyFile));
            privateKeyOS.writeObject(key.getPrivate());
            privateKeyOS.close();
          } catch (Exception e) {
            e.printStackTrace();}
        }
        /**
```

```
         * The method checks if the pair of public and private key has been
generated.
         * @return flag indicating if the pair of keys were generated.*/
        public static boolean areKeysPresent() {

          File privateKey = new File(PRIVATE_KEY_FILE);
          File publicKey = new File(PUBLIC_KEY_FILE);
          if (privateKey.exists() && publicKey.exists()) {
            return true;    }
          return false;  }
        /**
         * Encrypt the plain text using public key. */
        public static byte[] encrypt(String text, PublicKey key) {
          byte[] cipherText = null;
          try {
            // get an RSA cipher object and print the provider
            final Cipher cipher =
Cipher.getInstance("RSA/ECB/PKCS1PADDING");
            // encrypt the plain text using the public key
            cipher.init(Cipher.ENCRYPT_MODE, key);
            cipherText = cipher.doFinal(text.getBytes());
          } catch (Exception e) {
            e.printStackTrace();}
          return cipherText;  }
        /**
         * Decrypt the cipher text using private key.*/
        public static String decrypt(byte[] text, PrivateKey key) {
```

```
    byte[] dectyptedText = null;
    try {
      // get an RSA cipher object and print the provider
      final Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1PADDING");
      // decrypt the text using the private key
      cipher.init(Cipher.DECRYPT_MODE, key);
      dectyptedText = cipher.doFinal(text);
    } catch (Exception ex) {
      ex.printStackTrace();}
    return new String(dectyptedText);  }
  }
```

Appendix C

Java code contains method to generate hash code

```
package TransientAuthentication;
import java.security.MessageDigest;
/**
 *This class generate hash value */
public class sha1 {
    public static String SHA1(String input){
   try{
      MessageDigest mDigest=MessageDigest.getInstance("SHA1");
      byte[] result=mDigest.digest(input.getBytes());
      StringBuffer sb= new StringBuffer();
      for (int i=0;i<result.length;i++){

sb.append(Integer.toString((result[i]&0xff)+0x100,16).substring(1));
      }
      return sb.toString();
      }catch (Exception e){
      throw new RuntimeException(e);}
}
}
```

Appendix D

Java code do Bluetooth inquery

```
package TransientAuthentication;
import java.io.*;
import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
// A simple class that return MAC address of token
public class DeviceInquery implements DiscoveryListener{
    //object used for waiting
    private static final Object lock=new Object();
    //vector containing the devices discovered
    private static  Vector vecDevices=new Vector();
    @SuppressWarnings("CallToPrintStackTrace")
      public static String inquiry() throws IOException{
        DeviceInquery client=new DeviceInquery();
        LocalDevice localDevice = LocalDevice.getLocalDevice();
        //find devices
        DiscoveryAgent agent = localDevice.getDiscoveryAgent();
        System.out.println("Starting device inquiry...");
        agent.startInquiry(DiscoveryAgent.GIAC, client);
        try {
```

```
    synchronized(lock){
       lock.wait();}
     }
catch (InterruptedException e) {
    e.printStackTrace();}
System.out.println("Device Inquiry Completed. ");
//print all devices in vecDevices
int deviceCount=vecDevices.size();
if(deviceCount <= 0){
    System.out.println("No Devices Found .");
    System.out.println("Registeration not complate");
    System.exit(0);}
else{
    //print bluetooth device addresses and names in the format [
No. address (name) ]
    System.out.println("Bluetooth Devices: ");
    for (int i = 0; i <deviceCount; i++) {
       RemoteDevice
remoteDevice=(RemoteDevice)vecDevices.elementAt(i);
       System.out.println((i+1)+".
"+remoteDevice.getBluetoothAddress()+"
("+remoteDevice.getFriendlyName(true)+")");
    }
}
//select MAC address of token
int index=0;
try{
System.out.print("Choose Device index: ");
```

```
            BufferedReader bReader=new BufferedReader(new
InputStreamReader(System.in));
            String chosenIndex=bReader.readLine();
            index=Integer.parseInt(chosenIndex.trim());
            }catch(NumberFormatException e){
               System.err.println(e+"__________Start Again");
               System.exit(0);}
            String mac = null;
            try{
            RemoteDevice
remoteDevice=(RemoteDevice)vecDevices.elementAt(index-1);
            String address=remoteDevice.getBluetoothAddress();
            mac=address;
            }catch(ArrayIndexOutOfBoundsException e){
               System.err.println(e);
               System.out.println("Registeration not
complate");System.exit(0);
            }
            return mac;
         }
         @Override
         public void deviceDiscovered(RemoteDevice btDevice,
DeviceClass cod) {
            //add the device to the vector
            if(!vecDevices.contains(btDevice)){
               vecDevices.addElement(btDevice);}
         }
         @Override
```

```
    public void serviceSearchCompleted(int transID, int respCode) {
        synchronized(lock){
            lock.notify();}
    }
    @Override
    public void inquiryCompleted(int discType) {
        synchronized(lock){
            lock.notify();}
    }//end method
    @Override
    public void servicesDiscovered(int i, ServiceRecord[] srs) {

        throw new UnsupportedOperationException("Not supported
yet."); //To change body of generated methods, choose Tools | Templates.
    }
}
```

Appendix E

Java code for transient authentication (Token side)

```
package Token;
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.security.PublicKey;
import java.util.Scanner;
import java.util.Vector;
import java.util.concurrent.TimeUnit;
import static Token.EncryptionUtil.PUBLIC_KEY_FILE;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.UUID;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;
```

```
/**
* A simple SPP client that connects with an SPP server
*/
public class SampleSPPClient implements DiscoveryListener{
//object used for waiting
private static Object lock=new Object();
//vector containing the devices discovered
private static Vector vecDevices=new Vector();
private static String connectionURL=null;
public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
SampleSPPClient client=new SampleSPPClient();
//display local device address and name
LocalDevice localDevice = LocalDevice.getLocalDevice();
System.out.println("Address:
"+localDevice.getBluetoothAddress());
System.out.println("Name: "+localDevice.getFriendlyName());
//find devices
DiscoveryAgent agent = localDevice.getDiscoveryAgent();
System.out.println("Starting device inquiry…");
agent.startInquiry(DiscoveryAgent.GIAC, client);
try {
synchronized(lock){
lock.wait();}
}catch (InterruptedException e) {
e.printStackTrace();}
System.out.println("Device Inquiry Completed. ");
```

```
        //print all devices in vecDevices
        int deviceCount=vecDevices.size();
        if(deviceCount <= 0){
           System.out.println("No Devices Found .");
           System.exit(0); }
        else{
           //print bluetooth device addresses and names in the format [ No.
address (name) ]
           System.out.println("Bluetooth Devices: ");
           for (int i = 0; i < deviceCount; i++) {
              RemoteDevice
remoteDevice=(RemoteDevice)vecDevices.elementAt(i);
              System.out.println((i+1)+".
"+remoteDevice.getBluetoothAddress()+"
("+remoteDevice.getFriendlyName(true)+")");
           }}
        System.out.print("Choose Device index: ");
        BufferedReader          bReader=new           BufferedReader(new
InputStreamReader(System.in));
        String chosenIndex=bReader.readLine();
        int index=Integer.parseInt(chosenIndex.trim());
        //check for spp service
        RemoteDevice
remoteDevice=(RemoteDevice)vecDevices.elementAt(index-1);
        connectionURL=("btspp://"+remoteDevice.getBluetoothAddress()+"
:1");
        if(connectionURL==null){
           System.out.println("Device    does    not    support    Simple    SPP
Service.");
```

```
   System.exit(0); }
try{
//connect to the device and send a encrypted PW
StreamConnection
streamConnection=(StreamConnection)Connector.open(connectionURL);
System.out.print("Enter Your Password:");
Scanner sc=new Scanner(System.in);
String pw=sc.nextLine();
int x=0;
byte[] ciperText;
String send;
ObjectInputStream inputStream = null;
inputStream = new ObjectInputStream(new
FileInputStream(PUBLIC_KEY_FILE));
final PublicKey publicKey = (PublicKey) inputStream.readObject();
do{
  x=0;
  ciperText=EncryptionUtil.encrypt(pw, publicKey);
  for(int y=0;y<256;++y){
     if(ciperText[y]==(byte)10 ||ciperText[y]==(byte)13 ){
       x=1;
       break;
     }}
}while(x==1);
send=new String(ciperText,"ISO-8859-1");
OutputStream outStream=streamConnection.openOutputStream();
```

```
PrintWriter pWriter=new PrintWriter(new
OutputStreamWriter(outStream));
for (int u=0;u<2;){
pWriter.write(send+"\r\n");
pWriter.flush();
++x;
if(x>60){
  TimeUnit.MILLISECONDS.sleep(5000);}
}
streamConnection.close();
}catch(IOException e){
   System.err.println("\nconnection failed");}
}//main
//methods of DiscoveryListener
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass
cod) {
   //add the device to the vector
   if(!vecDevices.contains(btDevice)){
vecDevices.addElement(btDevice);
   }}
//implement this method since services are not being discovered
public void servicesDiscovered(int transID, ServiceRecord[]
servRecord) {
   if(servRecord!=null && servRecord.length>0){
connectionURL=servRecord[0].getConnectionURL(0,false);}
synchronized(lock){
lock.notify();}
}
```

```
//implement this method since services are not being discovered
public void serviceSearchCompleted(int transID, int respCode) {
synchronized(lock){
lock.notify();}
}
public void inquiryCompleted(int discType) {
synchronized(lock){
lock.notify();}
}}
```