CHAPTER 2 LITERATURE SURVEY

2.1 Introduction

Development of a software system of any size and complexity requires careful planning and extensive research. The development of Enhanced Password-Based Authentication Protocol (E-PAP) is no exception. To ensure that the prerequisite knowledge is sufficient, research was conducted in several areas. The resulting information is summarized in the following chapters.

2.1 What is Authentication?

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. This is distinct from the assertion of identity (known as *identification*) and from deciding what privileges assign to that identity (*authorization*). While all three are important, authentication is the most risky from the perspective of network security. In private and public computer networks (including the Internet), authentication is commonly done through the use of logon passwords.

Authentication gives assurance of identity. It is the means of gaining confidence that people or things are who or what they claim to be. The legitimate owner of an identity is known as a *principal* [50]. Principals of various physical forms may need to be authenticated, for example, people, pieces of equipment, or running applications on computer systems.

Authentication relates to a scenario where some party (a *claimant* [50]) has presented a principal's identity and claims to *be* that principal. Authentication enables some other party (a *verifier* [50]) to gain confidence that the claim is legitimate. Authentication methods are based upon any of the following principles:

- The claimant demonstrates knowledge of something, e.g., a password. (something you know)

- The claimant demonstrates possession of something, e.g., a physical key or card. (something you have)
- The claimant exhibits some required immutable characteristic, e.g., a fingerprint. (something you are)

Besides above three main principles, a few authors also include [50]:

- Evidence is presented that the claimant is at a particular place (possibly at some particular time).
- The verifier accepts that some other party, who is trusted, has already established authentication.

2.1.1 ISO 7498

The OSI Security Architecture (ISO 7498-2) distinguishes between data origin authentication and (peer) entity authentication:

- *Entity authentication*: An identity is presented by a remote party participating in a communication connection or session. ISO 7498-2 defines entity authentication as 'the corroboration that an entity is the one claimed'.
- *Data origin authentication*: An identity is presented along with a data item. It is claimed that the data item originated from the principal identified.

Entity authentication may be either *unilateral* or *mutual*. With unilateral authentication, just one party to a communication activity authenticates to the other. With mutual authentication, the parties at both ends authenticate each other.

2.1.2 ISO/IEC 9798

In recent years, ISO/IEC JTC1/SC27 has been working on a multi-part standard, ISO/IEC 9798, specifying a general-purpose set of authentication protocols. The four parts published so far have covers the following area.

- ISO/IEC 9798-1: 1997 (2nd edition) - General model.
- ISO/IEC 9798-2: 1994 - Protocols based on symmetric encipherment.
- ISO/IEC 9798-3: 1994 - Protocols based on digital signatures.

- ISO/IEC 9798-4: 1995 - Protocols based on data integrity mechanisms.

The protocols specified in these standards have been specified for use in a variety of application domains. As such they have been designed to be as 'robust' as possible, such as they have been designed to resist all known attacks in Section 2.3.

A fifth part of ISO/IEC 9798 is currently awaiting publication. ISO/IEC 9798-5 will specifying a general-purpose set of authentication protocol about Zero Knowledge protocols which is explained in Section 2.4. The future version of E-PAP System follows the ISO/IEC 9798-5 specifications.

Entity authentication can only be achieved for a single instant in time. Typically, a mutual authentication protocol is used at the start of a connection between a pair of communicating entities.

If other securities (e.g. confidentiality, integrity) is required for information subsequently exchanged during the life of the connection, then other cryptographic mechanisms will need to be used, e.g. encipherment or the use of Message Authentication Codes (MACs), to protect that data. The keys needed for these cryptographic operations can be agreed and/or exchanged as part of the authentication protocol.

2.1.3 Authentication Protocol Model

Entity authentication is typically achieved using an authentication exchange mechanism. Such a mechanism consists of an exchange of messages between a pair of entities, and is usually called an authentication protocol. Figure 2-1 shows a typical authentication protocol model.

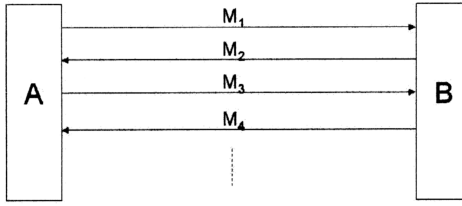

Figure 2-1: An authentication protocol model

A and B are a pair of communicating entities. A and B wish to use an authentication protocol to provide either unilateral or mutual authentication. The protocol consists of a (finite) sequence of messages: M_1, M_2, ... More specifically, suppose that A starts the protocol and sends B the message M_1. B then sends A the message M_2. A then sends B the message M_3, and so on until the protocol is completed. Note that messages are sent by A and B in strict alternation. In addition, A or B will not send message M_i until message M_{i-1} has been correctly received, where all cryptographic checks performed on the message give valid results.

Requirement I

A will want to be sure that:

- M_2, M_4, ... were all sent by B (as received),
- M_2, M_4, ... are 'fresh', or in other word, not replays of old messages,
- M_2, M_4, ... were intended for A, and not for any other entity, and
- M_2, M_4, ... were only generated by B after M_1, M_3, ... (respectively) were received correctly by B.

Requirement II

In a similar manner, B will want to be sure that:

- M_1, M_3, ... were all sent by A (as received),
- M_1, M_3, ... are 'fresh', or in other word, not replays of old messages,
- M_1, M_3, ... were intended for B, and not for any other entity, and

- M_3, M_5, ... were only generated by A after M_2, M_4, ... (respectively) were received correctly by A.

After completion of the protocol, both parties will have 'belief' requirements regarding the messages exchanged. It is important to observe that A and B may well not be sure of these 'beliefs' until after completion of the protocol [62].

2.1.4 Cryptographic Mechanisms

Authentication protocols require the use of a combination of either shared secrets (keys or passwords) or signature/verification key pairs, and accompanying cryptographic mechanisms. These are used to ensure that the recipient of a protocol message knows:

- where it has come from (origin checking),
- that it has not been interfered with (integrity checking).

Note that cryptographic mechanisms (by themselves) cannot provide freshness checking, for example, the verification that a protocol message is not simply a replay of a previously transmitted (valid) protocol message, protected using a currently valid key.

A variety of different types of cryptographic mechanism can be used to provide integrity and origin checking for individual protocol messages. This dissertation considers three main possibilities:

- by using encipherment,
- by using Message Authentication Code (MAC) mechanism,
- by using digital signature.

Encipherment

To protect a message in a protocol, the sender enciphers it with a secret key shared with the recipient. The recipient can then verify the origin of the message using the following process. The recipient first deciphers the message and checks that it 'makes sense'; if this is the case then the recipient reasons that it must therefore have

been enciphered using the correct secret key, and since only the genuine sender knows this key, it must therefore have been sent by the claimed originator.

This reasoning makes a number of assumptions about the nature of the encipherment algorithm and the capabilities of the recipient. Firstly, if this process is to be performed automatically by a computer or machine, then definition of 'makes sense' for a computer have to be made, especially as the contents of the message might include random session keys and random 'challenges'.

An interceptor is assumed to be unable of manipulating an enciphered message (without knowledge of the key used to encipher it) in such a way that it still 'makes sense' after decipherment. This constrains the type of encipherment algorithm that is suitable for use in this application; for example, stream ciphers are usually unsuitable for use as part of an authentication protocol.

The usual solution for defining 'makes sense' is the addition of deliberate 'redundancy' (according to some agreed formula) to the message prior to encipherment. The presence of this redundancy can then be automatically checked by the recipient of the message (after decipherment).

One common method of adding redundancy to a message is to calculate a *Manipulation Detection Code* (MDC), a sort of checksum dependent on the entire message, and append it to the message prior to encipherment. The MDC calculation function will typically be a public function.

MAC Mechanism

There are a variety of possible types of MAC (Message Authentication Code) mechanism. All MAC mechanisms have the following properties.

- a MAC mechanism is a function which takes as inputs a secret key and a message, and
- gives as output a fixed length MAC which is appended to the message as a 'cryptographic check value'.

The recipient of a protocol message checks it by re-computing the MAC, using a shared secret key, and comparing it with the value appended to the message. The appended MAC provides both:

- origin checking, because the originator and verifier are assumed to be the only possessors of the secret key, and hence are the only ones capable of computing a valid MAC for a message, and
- integrity checking, because, without the secret key, the new MAC corresponding to a changed message cannot be calculated.

The 'standard' way of computing a MAC is by using a block cipher, e.g. DES, in Cipher Block Chaining (CBC) mode. The result of enciphering the last block forms the MAC. This is a CBC-MAC.

Alternatively, a *one-way function* can be used as the basis for a MAC mechanism. A one-way function is a function which is simple to compute yet computationally infeasible to invert (given an input it is easy to compute the corresponding output, but given an arbitrary output it is computationally infeasible to find an input which gives this output). To use such a function as a MAC mechanism involves concatenating the message to be protected with a secret key and using this as the input to the one-way function. The resulting output is then the MAC. An example of this type of MAC is provided by the HMAC scheme.

Digital Signatures

A digital signature function is an example of a public key scheme (an asymmetric cryptographic technique). The fundamental idea of asymmetric cryptography is that keys come in matching pairs made up of a public key and a private key, and that knowledge of the public key of a pair does not reveal knowledge of the private key.

The pair of keys for a digital signature algorithm are:

- a private 'signing key' (which defines the signature transformation), and
- a public 'verification key' (which defines the verification transformation).

A signature will typically function somewhat like a MAC, in that the possessor of the private key can use it to derive the digital signature of a message, which is then appended to the message. The recipient then uses the public verification key to check the signature, thereby enabling the recipient to check the origin and integrity of a message.

2.1.5 Freshness Mechanisms

Providing origin and integrity checking for messages is not all that is required for a authentication protocol. An authentication protocol needs a means of checking the 'freshness' of protocol messages to protect against replays of messages from previous valid exchanges.

There are two main methods of providing freshness checking:

- the use of time-stamps (either clock-based or 'logical' time-stamps),
- the use of nonces or challenges (as in the challenge-response protocols discussed in Section 2 of these course notes).

Clock-based Protocol

Clearly the inclusion of a date/time stamp in a message enables the recipient of a message to check it for freshness, as long as the time-stamp is protected by cryptographic means. However, in order for this to operate successfully all entities must be equipped with securely synchronised clocks. It is non-trivial to provide such clocks (such as the clock drift of a typical workstation can be 1-2 seconds/day).

Every entity receiving protocol messages will need to define a time acceptance 'window' on either side of their current clock value. A received message will then be accepted as 'fresh' if and only if it falls within this window. This acceptance window is needed for two main reasons:

- clocks vary continuously, and hence no two clocks will be precisely synchronised, except perhaps at some instant in time, and
- messages take time to propagate from one machine to another, and this time will vary unpredictably.

Logical Time-stamps

One alternative to the use of clocks is for every pair of communicating entities to store a pair of sequence numbers, which are used only in communications between that pair. For example, for communications between A and B, A must maintain two counters: N_{AB} and N_{BA} (B will also need to maintain two counters for A). Every time A sends B a message, the value of N_{AB} is included in the message, and at the same time N_{AB} is incremented by A.

Every time A receives a message from B, then the sequence number put into the message by B (N say) is compared with N_{BA} (as stored by A), and:

- if $N > N_{BA}$ then the message is accepted as fresh, and
- N_{BA} is reset to equal N, if $N \leq N_{BA}$ then the message is rejected as an 'old' message.

Before proceeding note that all time-stamp protocols (both clock-based and logical time-stamp based) have problems in providing the property that M_i was only generated by peer after M_{i-1} was received correctly by peer.

Nonce-based

Nonce-based (or *challenge-response*) protocols use quite a different mechanism to provide freshness checking. One party, say A, sends an other party, say B, a nonce (**n**umber used **once**) as a challenge. B then includes this nonce in the response to A. Because the nonce has never been used before, at least within the lifetime of the current key, A can verify the 'freshness' of B's response (given that message integrity is provided by some cryptographic mechanism).

Note that it is always up to A, the nonce provider, to ensure that the choice of nonce is appropriate or it has not been used before.

The main property required of a nonce is the *one-time* property. Thus, if that is all that is ever required, A could ensure it by keeping a single counter and whenever a

nonce is required, for use with any other party, the current counter value is used (and the counter is incremented).

2.2 *Terminology and Background*

2.2.1 The Players

To help demonstrate protocols, this dissertation enlisted the aid of several parties (Table 2-1).

Table 2-1: Dramatis Personae

Alice, A	User or Client, first participant in all protocols
Bob, B	Host or Server, second participant in all protocols
Eve	Eavesdropper
Mallory	Malicious active attacker

2.2.2 Password and Verifier

Password and *verifier* [19] corresponds to conventional private and public keys albeit a few dissimilar properties. Unlike typical private keys, the password has limited entropy as it constrained by the memory of the user. *Plaintext-equivalent* authentication mechanism requires the server to store a copy of the user's password or private key.

Since a verifier is easily computed from the password, but deriving the password from the verifier is computationally infeasible. Therefore verifier has similar mathematical properties to a public key. Instead of being a publicly-known quantity, however, the verifier is kept secret by the server. *Verifier-based* authentication mechanism requires only a verifier to be stored. Verifier-based protocols have a significant advantage over plaintext-equivalent protocol. A system that uses plaintext-equivalent authentication becomes instantly compromised if the password database is revealed, since every user's password is stored there. An example is UNIX system store user's password in `/etc/passwd`. A database of verifiers, on

the other hand, can be protected just as easily and effectively as a database of plaintext-equivalent passwords, except that failure of the said protection is not as catastrophic if only verifiers are compromised.

2.2.3 Known Plaintext and Verifiable Text

A message contains a *Known plaintext* [16] if a cryptanalyst is presented with a piece of ciphertext and can predict all or part of the plaintext before the message is decrypted. Any predictable information may constitute known plaintext whether or not its position in the message was known in advance. For example, if one sends an encrypted message about this dissertation to someone else, a reasonable guess would be the message contains words such as "University of Malaya" or "authentication protocol".

The mechanism of cryptanalyst may like this. Suppose an attacker has obtained $\{p,q\}_r$ (which has information p and q and encrypted by password r or public key r) and he knows the value p. One way to exploit this information is by guessing r^{-1} and then compute $\{\{p,q\}_r\}_{r^{-}}$. If the result contains p, the attacker knows that his guess is correct. In public-key cryptosystem, it is difficult to guess the private key. However, he knew $\{p,q\}_r$, p and the public key r, therefore he can guess a value of q', compute $\{p,q'\}_r$ and compare it to $\{p,q\}_r$.

Verifiable text [16] could be a plaintext, a ciphertext, or the result of a chain of computation that may or may not involve encryption or decryption. Recall that in order to discover a poorly chosen secret, an attacker takes a number of guesses and, together with his knowledge of the system, performs computations to see whether he can rediscover something that he already knows or can recognize. Often if he discovers a match, the guess was correct with a high probability.

Let $P\{y|x\}$ denote the probability that y is true given that x is true. Let t be a threshold probability. For a poorly chosen secret s, v should be considered as "verifiable text" if

- the attacker can recognize v when he sees it,

- there exists a function *f()* such that $v = f(s)$ and it is feasible for the attacker to compute $f(s')$ for a sufficient number of values of s'.

2.2.4 Poorly Chosen and Well Chosen

Poorly chosen describes an encryption key derived from a user-chosen password. *Well chosen* describes an encryption key chosen at random from a large key space. Probability of an attacker successfully guessing a well-known random chosen key is presumably low than a poorly user chosen password [5]. A guideline for a well chosen password can be found at Appendix A.

2.2.5 Symmetric and Asymmetric Cryptography

Symmetric cryptography is an encryption system that uses the same key for encryption and decryption, sometimes referred to as Secret-Key Cryptography. Examples of symmetric cryptography include DES (Data Encryption Standard), Blowfish, RC5 and IDEA [4].

Asymmetric cryptography is an encryption system that uses different keys, for encryption and decryption. The two keys have an intrinsic mathematical relationship to each other, sometimes also called Public-Key Cryptography. Examples of asymmetric cryptography include RSA, Pohlig-Hellmaan, Rabin, ElGamal, McEliece and LUC [4].

2.2.6 Public Key Infrastructure (PKI)

A public key infrastructure (PKI) consists of protocols, services, and standards supporting applications of public-key cryptography. The term PKI, which is relatively recent, is defined variously in current literature. PKI sometimes refers simply to a trust hierarchy based on public key certificates [22], and in other contexts embraces encryption and digital signature services provided to end user applications. A middle view is that the public key infrastructure consists of [30]:

- A certificate authority (CA) that issues and verifies digital certificates. A certificate includes the public key or information about the public key
- A registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor
- One or more directories where the certificates (with their public keys) are held (usually in an ITU X.500 standard directory)
- A certificate management system

Among the services likely to be found in a PKI are the following:

- key registration: issuing a new certificate for a public key
- certificate revocation: canceling a previously issued certificate
- key selection: obtaining a party's public key
- trust evaluation: determining whether a certificate is valid and what operations it authorizes
- Key recovery (suggested as a possible aspect of a PKI)

In public key cryptography, a public and private key are created simultaneously using the same algorithm (a popular one is known as RSA) by a certificate authority (CA). The private key is given only to the requesting party and the public key is made publicly available (as part of a digital certificate) in a directory that all parties can access. The private key is never shared with anyone or sent across the Internet.

A number of products are offered that enable a company or group of companies to implement a PKI. The acceleration of e-commerce and business-to-business commerce over the Internet has increased the demand for PKI solutions. Related ideas are virtual private networks (VPNs) and the IP Security (IPSec) standard. Among PKI leaders are:

- RSA, which has developed the main algorithms used by PKI vendors:
- Verisign, which acts as a certificate authority and sells software that allows a company to create its own certificate authorities
- GTE CyberTrust, which provides a PKI implementation methodology and consultation service that it plans to vend to other companies for a fixed price

- Check Point, which offers a product, VPN-1 Certificate Manager, that is based on the Netscape Directory Server
- Xcert, whose Web Sentry product checks the revocation status of certificates on a server, using the Online Certificate Status Protocol (OCSP)
- Netscape, whose Directory Server product is said to support 50 million objects and process 5,000 queries a second; Secure E-Commerce, which allows a company or extranet manager to manage digital certificates; and Meta-Directory, which can connect all corporate directories into a single directory for security management

Therefore, efforts to define a PKI generally presume there will eventually be one, or, increasingly, that multiple independent PKIs will evolve with varying degrees of coexistence and interoperability. In this sense, the PKI today can be viewed akin to local and wide-area networks in the 1980's, before there was widespread connectivity via the Internet. As a result of this view toward a global PKI, certificate formats and trust mechanisms are defined in an open and scaleable manner, but with usage profiles corresponding to trust and policy requirements of particular customer and application environments. For instance, it is usually accepted that there will be multiple "root" or "top-level" certificate authorities in a global PKI, not just one "root", although in a local PKI there may be only one root. Accordingly, protocols are defined with provision for specifying which roots are trusted by a given application or user.

Efforts to define a PKI today are underway in several governments as well as standards organizations. The U.S. Department of the Treasury and National Institute of Standards and Technology (NIST) both have PKI programs [23],[24], as do Canada [25] and the United Kingdom [26]. NIST has published an interoperability profile for PKI components and specifies algorithms and certificate formats that certification authorities should support. Standards bodies working on a PKI includes the IETF's PKIX and SPKI working groups [27],[28] and The Open Group [29]. Most PKI definitions are based on X.509 certificates, with the notable exception of the IETF's SPKI.

2.3 Security Threat

2.3.1 Active, Passive and Replay Attack

Active attack is an attempt to improperly modify data, gain authentication or gain authorization by inserting false packets into the data stream or by modifying packets transiting the data stream.

Passive Attack is an attack on an authentication system that inserts no data into the stream, but instead relies on being able to passively monitor information being sent between other parties. This information could be used a later time in what appears to be a valid session.

Replay attack is an attack on an authentication system by recording and replaying previously sent valid messages (or parts of messages). Any constant authentication information, such as a password or electronically transmitted biometric data, can be recorded and used later to forge messages that appear to be authentic.

2.3.2 Brute-force Attack

Brute-force attack, sometimes called *guessing* attack, which is performed through an iterative guessing and verification on the weak secret, is surprisingly successful [5]. The goal of a guessing attack is to find out the shared weak secret. If the shared secret is known to others, it is clear that secure communication is impossible. When performing a large number of guesses, guessing attacks are most effective. For example, all of the words in a machine-readable dictionary, which may be sold by some hacking club in CD-ROM, can be made automatically and each guesses verified without raising an alarm. A brute force cracker simply tries all possible passwords until it gets the password. From a cracker perspective, this is usually very time consuming. However, given enough time and CPU power the password will eventually be cracked. Most modern brute force crackers allow a number of options to be specified, such as maximum password length or characters to brute force with. Such attacks are classified into off-line and on-line methods.

For *on-line guessing attacks*, an attacker have access to another participant who shares the weak secret legitimately. Therefore, it is not difficult to notice an on-line guessing attack. After guessing a target password, the attacker tries to use a guessed password iteratively in on-line manners, for instance by replaying eavesdropped messages or by impersonating as other users with the guessed password. For impersonation, the attacker should construct a bogus message through the guessed password. Using the messages resulted from on-line guessing attack, the attacker can proceed with guessing in off-line manners [6].

For *off-line guessing attacks*, an attacker eavesdrops an protocol messages and stores them in a local storage for verification. Therefore, other participants cannot notice the off-line attack. After guessing a password included in the eavesdropped messages, the attacker tries to verify the guessed password iteratively in off-line manners. A common method is to compare the eavesdropped messages with the artificial messages that are reconstructed by using the guessed password [5].

2.3.3 Dictionary Attacks

A dictionary password cracker simply takes a list of dictionary words, and one at a time encrypts them to see if they encrypt to the one way hash from the system. If the hashes are equal, the password is considered cracked, and the word tried from the dictionary list is the password [61].

Some of these dictionary crackers can "manipulate" each word in the wordlist by using filters. These rules/filters allow a user to change "idiot" to "1d10t" and other advanced variations to get the most from a word list.

2.3.4 Classification of Attacker

Depending on the power and ability of attackers, different kinds of attacks are possible. Normally, attackers are classified into three categories [13].

- Active Attacker

Active attacker is the most powerful attacker who has the ability to intercept packets and modify the packet or insert his own packets.

- Eavesdropping Attacker

  Next in the power is the eavesdropping attacker who has the power to eavesdrop on other legitimate sessions.

- Querying Attacker

  The least powerful attacker is the querying attacker. The only capability this attacker possesses is to initiate sessions with one party (Alice) while pretending to be other party (Bob).

2.4 Interactive Proof and Zero Knowledge

Informally, an interactive proof is a protocol between two parties in which one party, called the *prover*, tries to prove a certain fact to the other party, called the *verifier*. An interactive proof usually takes the form of a challenge-response protocol, in which the prover and the verifier exchange messages and the verifier outputs either "accept" or "reject" at the end of the protocol. Apart from their theoretical interest, interactive proofs have found applications in cryptography and computer security such as identification and authentication.

It is useful for interactive proofs to have the following properties, especially in cryptographic applications:

- **Completeness**: The verifier always accepts the proof if the fact is true and both the prover and the verifier follow the protocol.
- **Soundness**: The verifier always rejects the proof if the fact is false, as long as the verifier follows the protocol.
- **Zero knowledge**: The verifier learns nothing about the fact being proved (except that it is correct) from the prover that he could not already learn without the prover, even if the verifier does not follow the protocol (as long as the prover does). In a zero-knowledge proof, the verifier cannot even later prove the fact to anyone else. Note that not all interactive proofs have this property.

A typical round in a zero-knowledge proof consists of a "commitment" message from the prover, followed by a challenge from the verifier, and then a response to the challenge from the prover. The protocol may be repeated for many rounds. Based on the prover's responses in all the rounds, the verifier decides whether to accept or reject the proof.

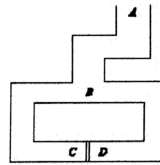

Figure 2-2: Ali Baba's Cave [9].

Let consider an intuitive example called Ali Baba's Cave [9] (Figure 2-2). Alice wants to prove to Bob she knows the secret words that will open the portal at C-D in the cave, but she does not wish to reveal the secret to Bob. In this scenario, Alice's commitment is to go to C or D. A typical round in the proof proceeds as follows: Bob goes to A and waits there while Alice goes to C or D. Bob then goes to B and shouts to ask Alice to appear from either the right side or the left side of the tunnel. If Alice does not know the secret words (e.g., "Open Sesame"), there is only a 50 percent chance she will come out from the right tunnel. Bob will repeat this round as many times as he desires until he is certain Alice knows the secret words. No matter how many times the proof repeats, Bob does not learn the secret words. There are a number of zero-knowledge and interactive proof protocols in use today as identification schemes. Zero knowledge do not leak any information about the password to the legitimate host (except the fact that the party at the other end really does know it) [9].

2.5 The Best Authentication System

What is the best authentication system in the world? The answer for this question is very simple: a system that combines three authentication methods mentioned in

Section 2.1 (something you have, you know and you are) together. A system that uses more than one authentication method is called a multi-factor authentication system. If each of authentication methods has 1% failure, that multi-factor authentication system will only have 0.0001% (1% x 1% x 1%) failure. However, this dissertation is not interested with multi-factor authentication system and only focuses on single-factor authentication system. (Section 2.6, 2.7 and 2.8)

2.6 Why password is used?

Most authentication protocols assume that a strong remote authentication is possible only if a cryptographically long or random secret is shared between the participants, in the light of an external threat of a dictionary attack or brute-force attack [35]. But in most systems, user-chosen secrets, such as passwords, are used for authentication. Clearly, cryptographically long passwords or long secrets chosen at random would be better for security only if ordinary users could remember them. But most users want to use a small secret and choose an easy-to-remember password because well-chosen passwords or large passwords are arguably weaker since they can't be memorized.

Public key technology presumed that a majority of personal computers and workstations would contain smartcard readers. This is not the case nowadays. Deploying public key applications without smartcards readers requires storing users private keys and X.509 certificates on their hard disks. However, storing such information on individual PCs is risky because this data can be stolen from the PC. It can also result in higher administration costs and support calls (e.g. when user data is lost due to a hard disk crash, accidental erasure or upgrade to a new computer).

The quantity of password variants equals to N^M, where N is the quantity of characters in a charset (the quantity of characters that may be virtually present in a password) and M is the maximum length of a password. By dividing this number by the speed of a search we obtain T - the time for a complete search, that is, in the 'worst case' time we have to wait to obtain a password. In the 'best case' the very first password will be the correct one. The probability of finding a password in time t

equals to $p=t/T$, for example, a probability to find the password in half of time indicated equals to 50% [51].

A password usually only contains letters. In this case the quantity of characters in a charset is 26 or 52, depending on usage of registers - both of them or just one. Some operating systems don't make any difference between lower-case and upper-case letters. Note that in the first case, the search time is 2^M times, not just 2 times. With an 8-characters' long password the difference would amount to 256 times, which is really significant. Table 2-2 estimates what the aforementioned formulas actually mean.

Table 2-2: Password search time with respect to the password and charset size [51].

Password length / charset	26 (no case, letters only)	36 (no case, letters&digits)	52 (case sensitive)	96 (all printable)
4	< 1 minute	< 1 minute	1 minute	13 minutes
5	< 1 minute	10 minutes	1 hour	22 hours
6	50 minutes	6 hours	2.2 days	3 months
7	22 hours	9 days	4 months	23 years
8	24 days	10.5 months	17 years	2287 years
9	21 months	32.6 years	881 years	219,000 years
10	45 years	1159 years	45,838 years	21 million years

The search speed is supposed equal to 100,000 passwords per second (a very decent speed). Normally, minimum password length for current applications is 6 or 8 characters with case sensitive. This means an attacker have to use 2.2 days to 17 years to get a correct password by using offline brute-force attack with a search speed of 100,000 password per second. Please note that if a system can only be attacked online, such as E-PAP System, the search speed will depend on the client and server processing speed plus total transmission time (including delay). Typically, it can only achieve 1 password per second. Consequently, all the figures in Table 2-2 have to be multiplied by 100,000. In other words, an authentication protocol that can prevent offline brute-force attack is theoretically unbreakable at the moment and this is the focused of this dissertation.

This dissertation will propose an Enhanced Password-Based Authentication Protocol (E-PAP) which is based on password only as it is inexpensive, maintain both user convenience and a high level of security which is similar to a smartcard at the same time. In addition, no trusted third party such as a key server or arbitrator is needed and only the original two parties are involved in the authentication protocol. This authentication protocol does not require anything more than a memorized small secret password which are not vulnerable to dictionary attack, making them much easier to use, more convenient and less expensive to deploy than either biometric or token-based (smartcard) methods. In a well-designed and managed system, passwords are more resistant to theft than persistent stored keys or tokens. Most Internet protocols currently employ plaintext password for authentication (Section 3.2.1), and it will be replaced with more secure alternatives if it can be done transparently. By using E-PAP that is discussed in this dissertation, it fits perfectly into such architecture without introducing significant user-visible overhead like extra smartcard reader attachment or biometric recognition machine.

2.7 Disadvantages of using Smartcard

Smart cards have been associated with security since they provide a partial solution to the need for authentication and non-repudiation. To the extent that a card is tamper-resistant, it can be used to store important secrets such as DES keys or RSA private keys. However, can smartcard be used in all scenarios without any problems or inconveniences? Does smartcard have no disadvantages? In some user environment, the inconveniences and cost of smartcards may outweigh the potential benefits of using it. Section 2.7.1 to 2.7.9 will discuss some limitations, problems and inconveniences of smartcards.

2.7.1 Equipment's Physical Limitations

Smartcard reader introduce equipment's physical limitations such as shortage of slots (either PCMCIA or IRQs) to the laptop or desktop personal computer. In addition, smartcards have a limited storage capacity. Some smartcards would only allow a

single key update before its memory was used up [1]. On the contrary, E-PAP allows an unlimited number of password or key updates.

2.7.2 Mobility Problem

A smartcard-based system doesn't automatically allow user mobility. User mobility is only possible if every machine that the user accesses has a smartcard reader attached. The machine must support the same smartcard reader interfaces standard or use the same proprietary smartcard reader. Similarly, to use the same machine sequentially, multiple users must all use the same smartcard technology [1].

2.7.3 Expensive

In addition, smartcard technology can be expensive. For example, the base price for a simple smartcard reader is US$25 (RM95.50) and significantly more if it has built-in security features such as key-pads, a PCMCIA-reader can cost up to US$250 (RM955), and the interface software requires an additional expenditure. The cards themselves cost between US$10 (RM38.20) and US$30 (RM114.60) each. Smartcard supplier may charge more than US$100 for the software license fee for each smartcard. While these costs should decrease with time, the initial investment in smartcard technology can still cost more than US$100,000 (RM382,000) for an organization such as a hospital that has several thousand employees [1].

2.7.4 Slower Performance

Some smartcard implementations have slower performance than software based authentication in Pentium-based PCs, both during initial loading when a user logs on and during message signing and encryption. Smartcards are typically 5 to 100 percent slower during message signing and encryption, but they are up to an order of magnitude slower in the worst cases. Because card implementations are relatively new, they are more buggy than most other software. For example, the PKI smartcard interface may not recognize a PCMCIA smartcard reader if dynamically removed and reinserted it (even after releasing the port) and the laptop have to be rebooted

each time. It isn't always possible to freely move smartcards between machines unless the configurations are identical, and even then it can be a complex operation [1].

2.7.5 Physical attacks

The most obvious and direct attack on a smart card is a physical attack on the card itself. In the case of a stored-value card, the owner of a card may even carry out this sort of attack. Physical attacks attempt to reverse engineer the card and determine the secret key(s). The attacks include: voltage manipulation, temperature manipulation, chip removal (for easier probing), UV light attacks and microprobing. Figure 2-3 shows for a fully functional smartcard processor with its covering plastic removed for microprobing experiments, all tools necessary for this preparation were obtained at a cost of only US$30 in a pharmacy [64].

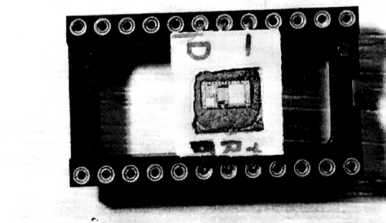

Figure 2-3: Removing a smartcard plastic cover. [64].

2.7.6 Advanced Attack Techniques

Armed with some special tools like focused ion beam (FIB) workstation, attacks on smartcards become much simpler and more powerful. A typical attack involves disconnecting almost all of the CPU from the bus, leaving only the EEPROM and a CPU component that can generate read accesses. For example, the program counter may be left connected in such a way that the memory locations will be accessed in order as the device is clocked (Figure 2-4). Read-out attack modifications on a security processor performed with a focused ion beam workstation allow easy access to secret EEPROM content with a single microprobing needle. Once this has been done, the attacker needs only a single microprobing needle or electro-optical probe to read the entire EEPROM contents. This makes the program much easier to analyse than in passive attacks, which typically yield only an execution trace; it also avoids the considerable mechanical difficulties of keeping several probes simultaneously located on bus lines that are perhaps a micrometer wide [64].

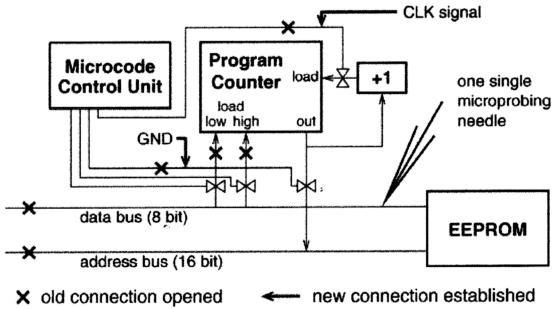

Figure 2-4: Read-out attack modifications with a FIB workstation [64].

2.7.7 Key compromise

Many smartcards carry secret crypto keys (usually DES or RSA private keys). If these keys are compromised, they can be used to load value onto the card, change card parameters, or bilk consumers by adjusting the card balance. Extreme care must be taken to guard crypto keys. Many researchers believe that giving a potential attacker a card with secrets on it that need to be protected from the attacker is a fundamentally flawed approach [63].

2.7.8 Risks of PKI

Security is a chain; it's only as strong as the weakest link. The security of any PKI-based system is based on many links and not all cryptographic. The immediate risk is on the part of PKI investors. The security risks are borne by anyone who decides to actually use the product of a commercial PKI [52].

2.7.9 Misbehaviour of Certificate Authorities

A more serious problem for smartcard is misbehavior of Certificate Authorities (CA) [34]. Let's assume Alice and Bob are two WWW users, wish to communicate securely using a standard browser. Perhaps they are engaged in an electronic commerce transaction. For a variety of reasons they enlist the help of a third party, CA, to introduce them. The protocol that they agree to use does not allow a CA to masquerade as either Alice or Bob without leaving evidence that may be used to show that CA has misbehaved. If a CA acts as a certification authority then with the existing certification schemes the following misbehavior is possible.

Since a CA, presumably, does not have access to Alice's private key she cannot forge a message signed under that key. However if Bob has never communicated with Alice he may not know her public key; in particular he is unlikely to have the public key certificate, signed by CA, certifying Alice's public key.

The CA can create a new certificate claiming that Alice has a different key and the corresponding private key can sign messages appearing to come from Alice, as

shown in Figure 2-5 and Figure 2-6. Alternatively if Bob wishes to communicate confidentially with Alice, the CA may use this new certificate to convince Bob to seal the message using the wrong key; the CA then unseals this message using the (wrong) private key.

Notation:

K_{XX}^{+} = Public key

K_{XX}^{-} = Private key

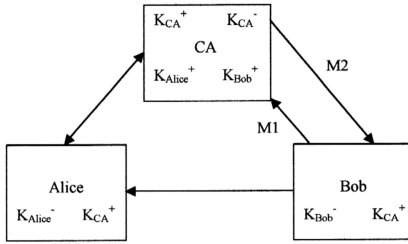

Figure 2-5: Forged Certificate [34].

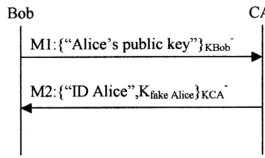

Figure 2-6: Exchanged Message [34].

Proponents of certification authorities might claim that the CA would be foolish to behave as described because if Bob does have Alice's certificate he would notice the misbehavior. However, any well designed certification scheme has to take into account the possibility that a certificate may need to be revoked - perhaps Alice has lost her private key and wishes to choose a new key.

If Alice detects this dishonest behavior but she has lost her genuine private key, she cannot prove it because the CA can always claim (cheating) to have issued the new (fake) public key after having received a request sent by Alice [34].

Another possible misbehavior by the CA is shown in Figure 2-7:

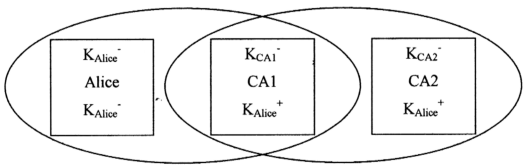

Figure 2-7: Register same user id and key in different domain [34].

Suppose Alice sends a certificate request to the certification authority CA1. Then CA1 could forward the request to another certification authority (CA2) specifying a new fake public key $K_{fakeAlice}^{+}$ and signing the request with the correspondent fake

private key $K_{fakeAlice^-}$, or she can try to register the same key under a fake user ID, or she can simply register the same user ID and key in a different domain.

The honest CA2 will issue a new valid certificate for Alice but binding the correct identity with a wrong key pair, which will be handled by the dishonest CA1, that can now masquerade as Alice in the domain 2. More important, even if the misbehavior is detected in the future, existing systems do not provide any facilities to build evidence that can be used as an irrefutable proof of the CA1's dishonest behavior.

2.8 Disadvantages of using Biometric

Biometric authentication is the process of proving ones identity via a physical measurement such as a fingerprint, facial recognition, iris scan, voice recognition and so on [2]. Usually, there are two types of biometric products in the market. The first one is *verification* where the user supplies a personal identification number or password coupled with some form of biometric characteristic that is then verified against the user's "Model-Template". The other is *identification* where the user supplies a biometric characteristic only, which is then verified against an on-line database to determine the user's identity. Identification products can eliminate alias use (multiple IDs for one user) and as such are much more difficult to develop [46].

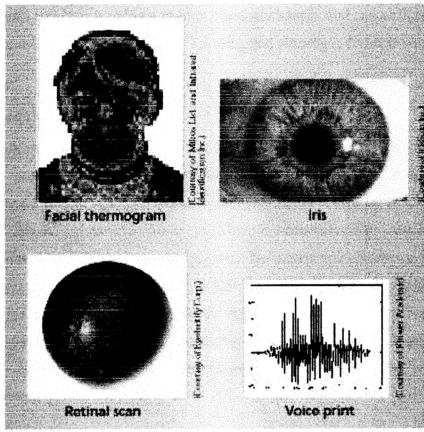

Figure 2-8: Examples of different biometric characteristics [2].

Listed below are a few disadvantages of various biometric methods [45]:

- *Hand geometry* devices are subject to physical changes that makes them less than ideal for large database sizes, where identification versus verification is required. These devices are also typically large and, therefore, difficult to integrate into many applications.
- Photographs can fool simple *facial recognition* technology and thermal facial recognition is typically cost prohibitive, thereby limiting its application in mass-market applications.
- *Iris scanning* has remained costly, is subject to user motion, and requires large "Model Template" sizes for data storage.
- *Retinal scanning* has also remained expensive and is subject to user health concerns over infrared or laser scanning of the retina.
- *Signature verification* is subject to user physical changes over time and is susceptible to forgery.

- *Voice analysis* is subject to user physical changes and can be forged through the use of devices capable of recording and altering individual voices.

2.8.1 Accuracy Problem

Like smartcard implementation, biometric authentication that needs extra devices, such as recognition machine, has equipment physical limitation, mobility, performance and cost problem. Besides, one significant drawback of biometric is a lack of clear-cut, compared to password, where the comparison process is very clear and a definite Yes or No answer can be given. No matter what biometric is used, it is extremely unlikely that the template and the new offered image will produce an exact match. This is because recording of the image of verification will vary, subject sensor noise, limitations of the processing methods and more importantly, the variability in both biometric characteristic as well as its presentation, which Figure 2-9 illustrates.

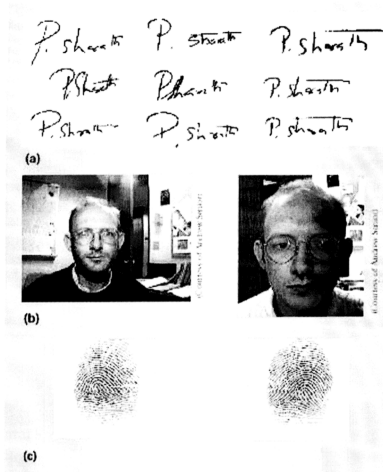

Figure 2-9: Variability is inherent in all signal readings, whether from (a) signature, (b) face or (c) fingerprint [2].

The level of accuracy of most devices is not 100% and a balance needs to be achieved between two types of error [62]:

- Type I error where the system fails to authenticate a valid user (a "false alarm") and
- Type II error: where the system accepts an impostor (a "false acceptance" or "impostor pass").

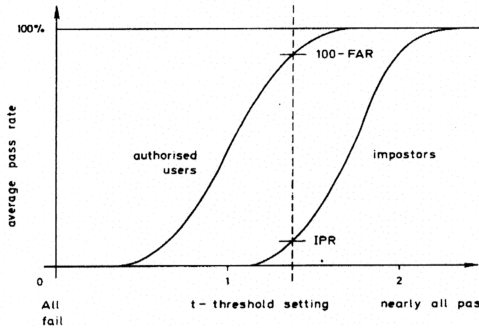

Figure 2-10: Pass Rate of Biometric [62].

Notes:

FAR: False Alarm Rate

IPR: Impostor Pass Rate

The performance of a typical biometric measurement system is shown in Figure 2-10. The effect of the choice of the threshold setting on the likelihood of Type I and Type II errors should be clear. A near ideal system has a large gap between the two curves.

2.8.2 Cost

Cost is tied to accuracy. Figure 2-11 roughly illustrates a comparison between cost and accuracy. The cost represented here is a typical incremental investment needed for acquiring a commercially available biometric sensor and for implementing an identity authentication system. The most costly technologies aren't necessarily the most accurate. Many applications like logging in to a PC are sensitive to the additional cost of including biometrics technology. Some applications (like laptop logon) cannot incorporate bulky biometric sensor hardware, which provides impetus for sensor miniaturization [2].

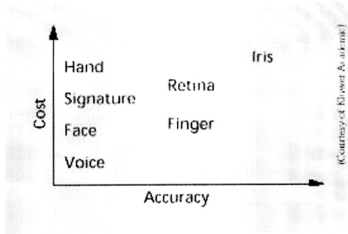

Figure 2-11: Cost versus accuracy [2].

2.8.3 Integrity

Authentication is unusable if the system cannot provide assurance that the legitimate owner indeed presented the characteristic. Data from multiple, independent biometric characteristics can serve to reinforce the identity of a subject. Multiple biometrics can alleviate several other practical problems in biometrics-based authentication.

For example, a fraction of the target population may either not actually possess a particular biometric identifier or may present a characteristic that does not tender any usable information, as Figure 2-12 illustrates. The information available in (a) is usable, whereas the information available in (d) is not. The other fingerprints, (b) and (c), are marginal. Furthermore, certain biometrics may not be acceptable to segments of the target population. Consequently, the integration of multiple biometric systems will become increasingly important [2].

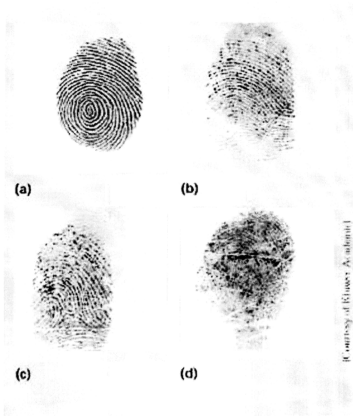

Figure 2-12: Usable, marginal and unusable fingerprint [2].

2.8.4 Ease of Use Problem

When using biometric, we have to analyse some practical implementations. How easy is it to use a given biometric system? Does the usage necessitate considerable user cooperation or is the acquisition of the characteristic too intrusive?

Does the system require a long training time? It is likely that obtrusive and cumbersome biometric authentication systems will be avoided much like we avoid systems requiring long passwords [2].

2.8.5 Ease of Development Problem

To foster improvements and encourage widespread deployment, biometric technology needs to be made easily accessible for system integration and

implementation. Integrating and harnessing biometrics technology is not easy in its present form. One of the reasons is the lack of industry-wide standards [2].