THE DESIGN AND DEVELOPMENT OF SECURE PASSWORD SYNCHRONIZATION AND QUERYING SYSTEM FOR ENTERPRISE NETWORKS

by

ZAFER GÜREL

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

August 2004

THE DESIGN AND DEVELOPMENT OF SECURE PASSWORD SYNCHRONIZATION AND QUERYING SYSTEM FOR ENTERPRISE NETWORKS

APPROVED BY:

Asst. Prof Albert Levi ………………………….
(Thesis Advisor)

Asst. Prof Erkay Savaş ………………………….
(Thesis Co-advisor)

Asst. Prof Hüsnü Yenigün ………………………….

Asst. Prof Ahmet Onat ………………………….

Asst. Prof İlker Hamzaoğlu ………………………….

DATE OF APPROVAL: ………………………….

ABSTRACT

Organizations that run large computer networks should also provide maintenance for the computers on these networks. Nowadays, it is a common practice to outsource this maintenance task to specialized service firms.

These service firms may not be considered trustworthy. Therefore, the local administrator password of a local machine that a maintenance technician needs to access should be changed periodically. Consequently, the technician needs a way to learn the current local administrator password of each computer.

In this thesis, a secure password synchronization and querying system is presented. In this system, the local administrator passwords of computers are changed periodically in synchronization with a server managing the system. The maintenance technicians can learn the current password of a computer by querying the server. For synchronization and querying mechanisms, we propose three secure protocols that employ symmetric and asymmetric encryption techniques.

Moreover, in this thesis, the proposed protocols are implemented as a software product and the performance of the system is evaluated by simulating the system. The average of the number of successful synchronizations stays constant when the number of computers is increased from 3,000 to 20,000 in the simulation. An increase in the number of computers doesn’t change the behavior of the system. In addition, it is shown that the system can be configured to survive under rough network conditions. The implementation details and the performance evaluation of the system are presented in the thesis.

ÖZET

Geniş bilgisayar ağları işleten organizasyonlar, bu ağlardaki bilgisayarların bakımını sağlamalıdırlar. Son zamanlarda, bu bakım işini uzman servis firmalarına devretmek genel bir iş pratiği haline gelmiştir.

Servis firmaları güvenilir bulunmayabilir. Bu yüzden bakım teknisyeninin bilmesi gereken yerel bilgisayarların yönetici şifreleri periyodik olarak değiştirilmelidir. Ayrıca, teknisyenin bir bilgisayarın mevcut yerel yönetici şifresini öğrenmesinin bir yolu olmalıdır.

Bu tezde, güvenli bir şifre senkronizasyonu ve sorgulama sistemi sunulmaktadır. Bu sistemde, bilgisayarların yerel yönetici şifreleri, sistemi idare eden bir sunucu ile senkronizasyon içinde belli aralıklarla değiştirilmektedir. Teknisyenler, bir bilgisayarın mevcut şifresini sunucuyu sorgulayarak öğrenebilirler. Senkronizasyon ve sorgulama mekanizmaları için simetrik ve asimetrik şifreleme tekniklerinin kullanıldığı üç güvenli protokol önermekteyiz.

Bu tezde önerilen protokoller bir yazılım ürünü olarak gerçekleştirilmiştir. Ortalama başarılı senkronizasyon sayısı, bilgisayar sayısı 3.000’den 20.000’e artırıldığında sabit kalmaktadır. Sistemin davranışı, bilgisayar sayısındaki artıştan etkilenmemektedir. Ayrıca, sistemin kötü ağ koşullarına rağmen çalışmaya devam edecek şekilde ayarlanabileceği de gösterilmiştir. Uygulama detayları ve sistemin performans değerlendirilmesi bu tezde sunulmuştur.

ACKNOWLEDGEMENTS

I would like to deeply thank my advisor Albert Levi and co-advisor Erkay Savaş for their assistance and guidance they have given to me. I would like to thank to the members of the thesis jury, Hüsnü Yenigün, Ahmet Onat, and İlker Hamzaoğlu for their invaluable contribution and insights.

I would also like to thank to my friends, Cem Atacık and Ali İnal for their support and significant contributions throughout my thesis. Moreover, I would like to thank all of my friends and my family members that have been tolerant to me while I have been too busy with my thesis.

Finally, I would like to thank to my parents, Fatma and Cengiz Gürel who are great individuals and parents. Without their support and sympathy, I could not have made it through the graduate school.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THE DESIGN AND DEVELOPMENT OF SECURE PASSWORD SYNCHRONIZATION AND QUERYING SYSTEM FOR ENTERPRISE NETWORKS

by

ZAFER GÜREL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2004

THE DESIGN AND DEVELOPMENT OF SECURE PASSWORD SYNCHRONIZATION AND QUERYING SYSTEM FOR ENTERPRISE NETWORKS

APPROVED BY:

Asst. Prof Albert Levi (Thesis Advisor)	………………………….
Asst. Prof Erkay Savaş (Thesis Co-advisor)	………………………….
Asst. Prof Hüsnü Yenigün	………………………….
Asst. Prof Ahmet Onat	………………………….
Asst. Prof İlker Hamzaoğlu	………………………….

DATE OF APPROVAL: ………………………….

ABSTRACT

Organizations that run large computer networks should also provide maintenance for the computers on these networks. Nowadays, it is a common practice to outsource this maintenance task to specialized service firms.

These service firms may not be considered trustworthy. Therefore, the local administrator password of a local machine that a maintenance technician needs to access should be changed periodically. Consequently, the technician needs a way to learn the current local administrator password of each computer.

In this thesis, a secure password synchronization and querying system is presented. In this system, the local administrator passwords of computers are changed periodically in synchronization with a server managing the system. The maintenance technicians can learn the current password of a computer by querying the server. For synchronization and querying mechanisms, we propose three secure protocols that employ symmetric and asymmetric encryption techniques.

Moreover, in this thesis, the proposed protocols are implemented as a software product and the performance of the system is evaluated by simulating the system. The average of the number of successful synchronizations stays constant when the number of computers is increased from 3,000 to 20,000 in the simulation. An increase in the number of computers doesn’t change the behavior of the system. In addition, it is shown that the system can be configured to survive under rough network conditions. The implementation details and the performance evaluation of the system are presented in the thesis.

ÖZET

Geniş bilgisayar ağları işleten organizasyonlar, bu ağlardaki bilgisayarların bakımını sağlamalıdırlar. Son zamanlarda, bu bakım işini uzman servis firmalarına devretmek genel bir iş pratiği haline gelmiştir.

Servis firmaları güvenilir bulunmayabilir. Bu yüzden bakım teknisyeninin bilmesi gereken yerel bilgisayarların yönetici şifreleri periyodik olarak değiştirilmelidir. Ayrıca, teknisyenin bir bilgisayarın mevcut yerel yönetici şifresini öğrenmesinin bir yolu olmalıdır.

Bu tezde, güvenli bir şifre senkronizasyonu ve sorgulama sistemi sunulmaktadır. Bu sistemde, bilgisayarların yerel yönetici şifreleri, sistemi idare eden bir sunucu ile senkronizasyon içinde belli aralıklarla değiştirilmektedir. Teknisyenler, bir bilgisayarın mevcut şifresini sunucuyu sorgulayarak öğrenebilirler. Senkronizasyon ve sorgulama mekanizmaları için simetrik ve asimetrik şifreleme tekniklerinin kullanıldığı üç güvenli protokol önermekteyiz.

Bu tezde önerilen protokoller bir yazılım ürünü olarak gerçekleştirilmiştir. Ortalama başarılı senkronizasyon sayısı, bilgisayar sayısı 3.000’den 20.000’e artırıldığında sabit kalmaktadır. Sistemin davranışı, bilgisayar sayısındaki artıştan etkilenmemektedir. Ayrıca, sistemin kötü ağ koşullarına rağmen çalışmaya devam edecek şekilde ayarlanabileceği de gösterilmiştir. Uygulama detayları ve sistemin performans değerlendirilmesi bu tezde sunulmuştur.

ACKNOWLEDGEMENTS

I would like to deeply thank my advisor Albert Levi and co-advisor Erkay Savaş for their assistance and guidance they have given to me. I would like to thank to the members of the thesis jury, Hüsnü Yenigün, Ahmet Onat, and İlker Hamzaoğlu for their invaluable contribution and insights.

I would also like to thank to my friends, Cem Atacık and Ali İnal for their support and significant contributions throughout my thesis. Moreover, I would like to thank all of my friends and my family members that have been tolerant to me while I have been too busy with my thesis.

Finally, I would like to thank to my parents, Fatma and Cengiz Gürel who are great individuals and parents. Without their support and sympathy, I could not have made it through the graduate school.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The amount of information shared over computer networks is increasing enormously as the sizes of computer networks increase and more business is carried out electronically. The organizations use computers much more than the past in order to increase their effectiveness.

Today, numerous big or even mid-sized organizations have computer networks with thousands of computers, which we will mention as Enterprise Networks throughout the thesis. As the computer becomes an inextricable part of businesses, organizations do not tolerate failures in their computer systems. Therefore, maintenance of computer systems and shortening repair time upon a failure are vital issues for organizations.

Some organizations carry out the maintenance of their computer systems by their IT departments, whereas others prefer outsourcing this task because of the following reasons:

- Outsourcing helps organizations to reduce costs and to focus on their core businesses.
- Since fast and accurate solutions are required in maintenance of a computer system, a specialized service firm is a better alternative than an IT department.

In order to carry out maintenance operations on a particular computer, a maintenance technician needs to gain administrative rights by using local administrative account credentials. Generally, the same username and password are used for each computer or a group of computers in the network to help the maintenance staff to easily recall the password for all computers. IT departments may be considered trustworthy whereas a service firm may not. Here lies the crucial requirement: The local administrative account username and password of a computer in the enterprise network must be changed periodically in order to prevent long-term hazards in case of a compromise. However, the maintenance technician still needs to know the current local administrator account username and password.

We propose a secure password synchronization, management, and querying system, called Enterprise-Level Sensitive Information Security System (ESIS). Local administrator passwords are very sensitive information in an enterprise network, but there may be other types of information as sensitive as the local administrator password and must be secured. Therefore, we have chosen the term, "Sensitive Information" in naming our system in order to cover all types of sensitive information including the "local administrator password". Although we focus only on administrator passwords, the proposed security protocols can easily be utilized in exchange and synchronization of other types of sensitive information, as well.

The local administrator passwords are changed periodically under the control of a server and in a synchronized fashion. Additionally, a maintenance technician can query the server to learn the current password of a computer in the network. The password synchronization and querying facility are provided by the following secure protocols that have been designed in the scope of this thesis:

- ESIS Secure Synchronization Protocol (ESSP)
- ESIS Client Initialization Protocol (ECIP)
- ESIS Secure Querying Protocol (ESQP)

In Section 2, the necessary background information about cryptography and security is given in order to clarify the details of the protocols. The overview of the proposed system and the secure protocols that the system depends on are detailed in Section 3. Details of the prototype that has been developed in order to implement the solution are given Section 4, and finally the performance evaluation of the system are detailed in Section 5.

2 BACKGROUND

Identification of users in a computer network is crucial. Consider a large enterprise network where so many computers and electronic devices are inter-connected to each other. In such a network, there are so many remote services provided such as printing, file sharing, remote access, etc. Only authorized users can use these services. For instance, when a user wants to check out his/her e-mails, he/she logs in by typing his/her username and password in the login form. If this information is correct, which means that the e-mail server authenticates the user as a result of successful matching of the entered username and password and the ones in the user database, he/she gets authorized to use the e-mail service. In fact, authorization and authentication are two separate concepts, but they are very close to each other in most applications. Authentication is the process of confirming the correctness of the claimed identity. Authorization is the approval, permission, or empowerment for someone or something to do something. Authentication is often a prerequisite to allowing access to resources in the system, i.e. for authorization.

2.1 Cryptography Basics

Authentication protocols and systems are based on cryptographic primitives such as encryption and hashing algorithms. In the ESIS System, we have used symmetric and asymmetric encryption, code obfuscation techniques as well as message authentication codes and hash functions. In this subsection, we briefly discuss these primitives.

2.1.1 Definitions

Cryptography is the art of making messages secret, so anyone who intercepts the message cannot understand it. Cryptanalysis is the art of breaking ciphertext by using special methods. Cryptology is the study of mathematical building blocks of cryptanalysis and cryptography. Cryptography deals with cryptosystems in which messages are encrypted and decrypted.

Encryption is the process of converting a message into an encoded form in order to hide the content from outsiders. The original message is described as plaintext or cleartext in cryptography terminology. The encrypted message is called ciphertext. Decryption is the reverse of encryption process. It converts ciphertext to plaintext. Generally in encryption and decryption processes, a key is used. A ciphertext which is encrypted by an encryption key cannot be decrypted without knowing the decryption key. The details of these encryption algorithms will be given later in this chapter. In Figure 2.1, encryption and decryption processes are depicted.

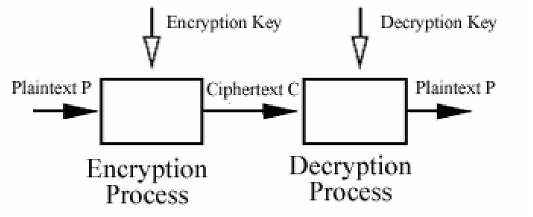


Figure 2.1. Encryption and decryption

Encryption/Decryption algorithms are split into two categories:

- Symmetric (secret-key) algorithms
- Asymmetric (public-key) algorithms

2.1.2 Symmetric Key Cryptography

In symmetric cryptographic algorithms, the same secret key is used in encryption and decryption processes. Today’s symmetric algorithms work on blocks of data or one bit at a time. They are called block ciphers and stream ciphers, respectively.

The most widely used symmetric block cipher is DES (Data Encryption Standard) [1]. It was published by NIST (National Institute of Standards and Technology) first in 1977. Until 2000’s, DES was the standard for symmetric encryption. DES operates on 64-bit blocks and its key length is 56. DES has been proved as a secure algorithm as more study has been done on it.

With the increase in computational power of computers, DES key length became too small to resist against brute-force attacks. In 1999, NIST published FIPS 46-3 [1] which is the new version of DES standard. In this standard, it was recommended that old version of DES should only be used in legacy systems. In the new systems, instead of DES, 3DES (triple DES) is recommended. 3DES utilizes DES algorithm and runs it three times in a row. It is resistant to cryptanalysis and it has a key length of 112-bit or 168-bit. However, 3DES is slow in software (DES was designed primarily for hardware) and its block length is 64 (Larger block size would be more secure). Therefore, NIST has decided to replace this standard.

In 1997, NIST started a contest by issuing a call for proposals for the new Advanced Encryption Standard (AES). NIST selected the Rijndael algorithm which was developed by Dr. Joan Daemen and Dr. Vincent Rijmen. In November 2001, with the publication of FIPS PUB 197 [2], AES became the new standard symmetric encryption algorithm. Rijndael or AES algorithm has the following characteristics as described in [3]:

- Very resistant against all known attacks
- High speed on various platforms
- Simple design

The AES algorithm is a block cipher algorithm which operates on 128, 192 or 256-bit blocks. Three different-sized symmetric keys of 128, 192 or 256 bits are used. The AES algorithm involves, *r* standard rounds (*r* may be 10, 12 or 14 depending on the block and key lengths). The relation between the number of rounds and key length is shown in Table 2.1. The details of the AES algorithm can be found in [2].

Key Length (in bits)	Number of rounds (r)
128	10
192	12
256	14

Table 2.1. Key lengths and number of rounds in AES

In ESIS, AES is employed in the synchronization and querying protocols as the symmetric encryption algorithm. The AES key length is 128 bits in ESIS.

2.1.3 Public Key Cryptography

The idea of public key cryptography has been first proposed in the seminal paper of Diffie and Hellman [4]. In public key cryptography, there is no shared secret key between two parties, which is the main difference between public key cryptography and symmetric key cryptography. In a public key cryptography based system, each principal has its own public key and private key pair. Public key can be obtained by all of the principals in the system whereas private key is just known by its owner and secret. A message encrypted by using public key can only be decrypted by private key, which provides secrecy of the message. The opposite situation is also valid. A message encrypted by using private key can only be decrypted by public key, which provides authenticity of the message, because only the owner of the private key can encrypt the message.

The most widely known public key algorithm is RSA which was developed by Rivest, Shamir and Adleman [5]. In RSA, encryption and decryption are the same processes. The strength of RSA depends on the computational infeasibility of factorization of large numbers. The factorization of large numbers is a complex and hard problem, but continuing increase in computing power and refinement of factoring algorithms decreases the amount of duration required to crack RSA. In the near future, key lengths of 1024-bit and 2048-bit seem reasonably sufficient as discussed in [3].

In ESIS, RSA is employed in the secure querying protocol. A session key is shared between two parties by using RSA encryption and decryption techniques. Moreover, RSA private key encryption and a hash function are used together to digitally sign technician's password stored in his/her token.

2.1.4 Cryptographic Hash Functions

A cryptographic hash function is an algorithm that computes a value based on a data object thereby mapping the data object to a smaller thumbprint as mentioned in [6]. To explain in more detail, a cryptographic hash function computes a fixed-size data from a message and this fixed-size data, which is called message digest, must have the following properties:

1. **One-way**: It must be computationally infeasible to compute the message from the message digest.
2. **Weak collision resistance**: It must be computationally infeasible to find another message that yields the same digest with a given message when it is hashed.
3. **Strong collision resistance**: It must be computationally infeasible to find a pair of messages that yields the same message digests.

Some of standard hash functions are SHA-1 [7] and RipeMD-160 [8]. Hash functions are used in a variety of applications. The followings are some of these applications:

- **Message Authentication Code (MAC)**: In constructing a message authentication code, hash functions are used.
- **Digital Signature**: In order to digitally sign a large document, it is hashed at first and the digest is signed for performance results.

- **Timestamps**: A timestamp is included in a message digest in order to know when the corresponding hashed document has been signed.
- **Virus Checking**: Digest of an unmodified version of a file can be used in order to reveal that this file can be infected by a virus or modified by comparing the digest of the possible infected file.
- **Securing Passwords**: In the password based authentication systems, digests of passwords are stored in a file or a database, so that a hacker cannot learn the passwords by just obtaining a copy of the password file or database.

In the ESIS System, hash functions are used in the protocols in order to create message authentication codes and to store thumbprints of passwords.

2.1.5 Digital Signatures

Public-key algorithms can be utilized to create digital signatures. A digital signature is a piece of data that is generated using a private key on a message. The corresponding public key is used to verify this signature on a message. The signature is also used to check the integrity of the message.

Another purpose of using digital signatures is to build a relationship between a public key and its owner. Owner's information and public key are digitally signed by a trusted authority. The resulting signed document is called a certificate. Certificates are verified to learn third party’s public keys in an authentic way. A third party key learnt in this way is trusted because it's signed by a trusted authority.

There are three important standard algorithms for generation and verification of digital signatures: Digital Signature Algorithm (DSA) [9], RSA [5], and Elliptic Curve DSA (ECDSA) [10].

2.1.6 Message Authentication Code

A message authentication code reduces a message to a fixed small block of data but it brings the notion of using a secret key. MAC functions are as fast as hash functions and in general a MAC function is irreversible like a hash function. In an authentication system, the MAC is appended to the message to provide integrity and authentication.

Hash-based MAC (HMAC) [11] is used in order to generate message authentication codes. In the secure protocols of the ESIS System, the participants that have the same symmetric key use HMAC function in order to authenticate the protocol messages.

2.1.7 Cryptographic Random Number Generators

Cryptographic Random Number Generators (RNGs) are used to generate random number sequences to be used in cryptographic applications. Some examples are listed below:

- In some key distribution systems, nonces are used to prevent replay attacks. Nonces are sent to the recipient as a part of the message. Since they are random numbers, the sent message cannot be replayed.
- In authentication protocols, session key generation is carried out by using a random number generator.
- RSA public-key encryption uses random number generators in order to generate the public and private key pair.

The ESIS System make use of RNGs in order to generate random binary sequences. These binary sequences are used in the generation of new passwords and timestamps.

They are different from normal random number generators which are available in many programming languages or operating systems in terms of high resistance to cryptanalysis. They can be used to generate cryptographic keys.

In the ideal case, random numbers are based on truly random sources such as physical noise generators, the most significant bits of an audio input or keystrokes on a computer keyboard. But in cryptographic applications, such sources are often unavailable. Therefore, pseudo-random numbers are used instead of truly random numbers. Pseudo-random numbers are outcomes of deterministic algorithms and not statistically random but they are successfully tested to verify that they are close to true randomness. Despite the determinism in their calculation, they are useful as it is stated in [12]:

> *For practical purposes we are forced to accept the awkward concept of "relatively random" meaning that with regard to the proposed use we can see no reason why they will not perform as if they were random (as the theory usually requires.). This is highly subjective and is not very palatable to purists, but it is what statisticians regularly appeal to when they take "a random sample" - they hope that any results they use will have approximately the same properties as a complete counting of the whole sample space that occurs in their theory.*

2.2 Authentication

Authentication is the process of determining whether a participant’s identity is correct. In computer systems, participants are allowed to use resources after they are authenticated.

We can differentiate between two types of authentication: machine-by-machine authentication (machine authentication) and human-by-machine authentication (user authentication) as discussed in [13]. These two types of authentication are depicted in Figure 2.1.

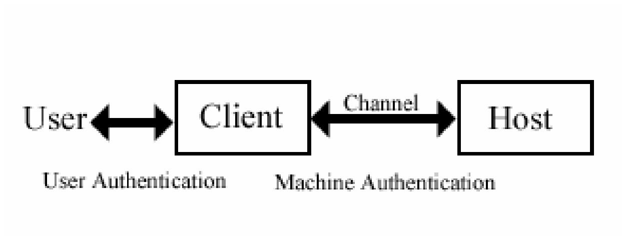


Figure 2.2. User authentication and machine authentication

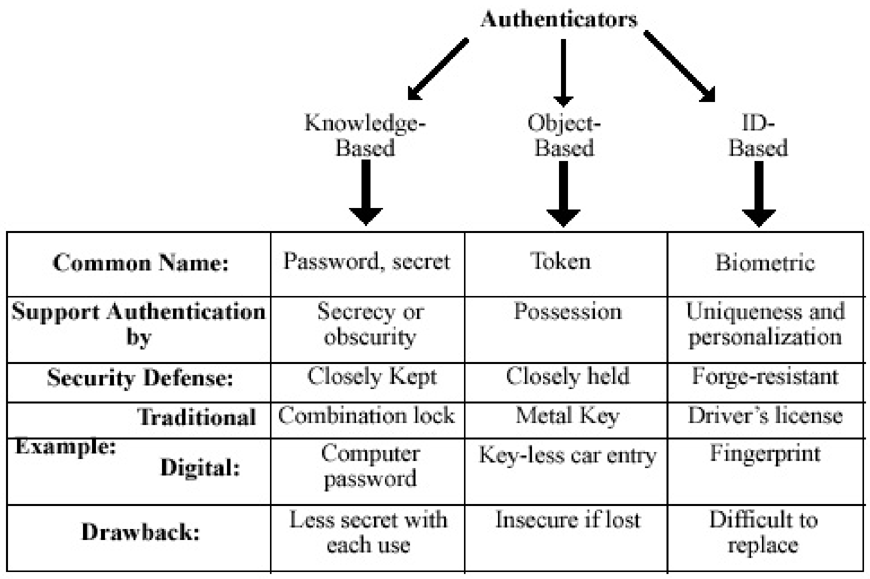
Machine authentication is securer than user authentication. In machine authentication, two machines use strong encryption algorithms with long and unpredictable encryption keys in message exchange, so it is infeasible for a malicious person to crack the message exchange between them. However, in user authentication a person uses a password instead of a long key because he/she cannot remember a long sequence of bits. This password is then converted into a sequence of bits by applying a hash function. The result of this function is again unpredictable for a human but the password is not. People tend to choose passwords that may be guessed easily as described in [14]. Therefore, user authentication is the weakest link in the whole authentication chain.

2.2.1 Authenticator Types

Before the Internet usage wasn’t wide-spread as it is today and computers were generally single working machines and not connected to a network, people had to remember a single password to login to their computers. But with the rise of Internet and the computer systems and applications based on TCP-IP networking, number of passwords that a person possesses has increased as well. The most of the user authentication systems depend solely on password-usage. The password is the simplest but the most widely used way of user authentication but it has drawbacks and security risks. There are other types of user authentication methods which are harder to implement but securer than passwords. These are biometric and token based methods.

In [13], authenticators illustrated in Figure 2.3 are categorized as follows:

1. **Knowledge-Based Authenticators** (What You Know): A knowledge-based authenticator is a secret phrase or a single secret word. A memorized password is an example to this type of authenticator.
2. **Object-based Authenticators** (What You Have): Object-based authenticators are physically owned things such as specially designed temper-resistant digital tokens or smart cards.
3. **ID-based Authenticators** (Who You Are): ID-based authenticators are unique to one person, so that they can identify their owners. Biometrics such as fingerprint, signature or eye scan are in this category.



Common Name:	Password, secret	Token	Biometric
Support Authentication by	Secrecy or obscurity	Possession	Uniqueness and personalization
Security Defense:	Closely Kept	Closely held	Forge-resistant
Example: Traditional	Combination lock	Metal Key	Driver's license
Example: Digital:	Computer password	Key-less car entry	Fingerprint
Drawback:	Less secret with each use	Insecure if lost	Difficult to replace

Figure 2.3. User authentication is split into three categories

The password. The most widely used way of authentication is the password [15]. The term password means a personal identification number (PIN), a phrase or a single word. Although it is a very convenient authenticator, it is the least secure way of user authentication and this has proven by the studies [14, 16, 17, 18].

It is obvious that password-based authentication systems include security risks and threats. Because of the risks in password-based authentication systems, NIST have published a document [19] including guidelines identifying 10 characteristics in designing a password system. Some of the characteristics are as follows:

- length
- character set
- lifetime
- source, i.e. user or administrator generated
- ownership, preferably individual
- distribution
- storage
- entry e.g. keyboard, display, possibility of being overlooked
- etc.

Other Authenticators. A token is a device that you have in an authentication process and performs the authentication. It can be a smart card, a USB memory stick or your credit card. The token simply defines the object-based authenticators. Your possession of the token gets you authenticated.

A biometric is a feature of your body that can distinguish you from other people. Biometrics include fingerprint, iris, retina, face, voice, and hand. A biometric shows who you are with no doubt so that non-repudiation never occurs.

Object-based (token) and id-based (biometric) authenticators are more secure in some conditions but sometimes they are not as feasible as the password for some conditions, too. In every use of a password, it gets more insecure to use it. You can lose your digital token. A biometric is difficult to replace (you cannot replace your eye). Also, as shown in Table 2.2 and Table 2.3 [13], every authenticator has its own security risks in certain circumstances.

Every authenticator has its own advantages and disadvantages. For example, it may not be convenient or feasible to implement a token or biometric based computer login system for every user in a large corporate network. Such a system requires a substantial initial investment for the electronic devices such as smart cards, smart card readers, digital tokens or biometric reader devices for every workstation in the network. The total cost includes not only this initial investment but also the constant maintenance and administrative costs.

Attacks	**Authenticators**	**Examples**	**Typical Defenses**
Client Attack	Password	Guessing, exhaustive search	Large Entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts, theft of object requires
	Biometric	False match	Large entropy; limited attempts
Host Attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	1-time passcode for each session
	Biometric	Template theft	Capture device authentication
Eaves-dropping, Theft and Copying	Password	Shoulder surfing	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multi-factor authentication
	Token	Theft, counterfeiting hardware	Multi-factor authentication; tamper resistant/evident hardware token
	Biometric	Copying biometric	Copy-detection at capture device ad capture device authentication
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode per session
	Biometric	Replay stolen biometric template response	Copy-detection at capture device and capture device authentication via challenge-response protocol
Trojan Horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device; client or capture device within trusted security perimeter
Denial of Service	Password, token, biometric	Lockout by multiple failed authentications	Multi-factor with token

Table 2.2. Some potential attacks on authenticators

In the real life, while designing a security system process, the choice of the authentication system depends on the cost, convenience and the required security level of the system. It can easily be claimed that cost and security level are directly proportional whereas security level and convenience are inversely proportional [20].

Security Issues	**Authenticators**	**Examples**	**Typical Defenses**
Non-repudiation	Password, token	Claim lost or stolen authenticator	Personal liability, two-factor with biometric (e.g. signature)
	Biometric	Claim copied biometric	Capture device authentication
Compromise Detection	Password, biometric	Stolen password or copied biometric	"Last login" displayed to user to detect anomaly
	Token	Lost or stolen token	User notes physical absence
Administrative and Policy-Registration/Enrollment	Password	Initial password registration	Delivery to pre-established e-mail address
	Token	New token registration	Delivery to pre-established physical address
	Biometric	Biometric enrollment	In-person with picture ID
Administrative and Policy-Reset and Recovery	Password	Forgotten password	Secondary authenticator (e.g., date of birth)
	Token	Lost token	Delivery to pre-established physical address
	Biometric	Compromised biometric	Not much option but to revert to password

Table 2.3. Other security issues about authenticators

Multi-factor Authenticators. These three types of authenticators can be used together to increase the security level of the authentication system. For example, a bank credit card cannot be used to withdraw money from an ATM machine without knowing the corresponding password. In Table 2.4 [13], the security advantages and drawbacks of combining authenticators are shown.

Authenticator Combination	**Security Advantage**	**Convenience Drawback**	**Example**
Knowledge- and Object-Based	Lost/stolen token protected by password	Must carry token and memorize password	PIN-enabled bank card
Object- and ID- Based	Lost/stolen token protected by ID	Must carry token, but not ID if it is a biometric	Photo-ID
Knowledge- and ID-Based	Two factors provide security in case either compromised	Have to memorize password and have ID	Password and biometric for computer access
Knowledge-, Object-, and ID- Based	A third factor to provide security in case two other factors are compromised	Have to memorize password, carry token, and have ID	Military applications requiring photo-ID checked by guard, plus password

Table 2.4. Combining authenticators

One of the authentication protocols developed in the scope of this study relies on the knowledge and object based multi-factor authenticator. Knowledge-based authenticator is a username-password combination whereas object-based authenticator is a token. These authenticators are mutually dependent to each other because they should be used together in authentication.

In our study, each username in the authentication system is unique. Moreover, a unique binary string is stored in every user’s token. There is a one-to-one relationship between usernames and binary strings in the tokens. The combination of the binary string, the username, and the password represent an individual principal in the authentication system. Therefore, compromise of one authenticator is not sufficient to break the system. Also, we have designed this unique string to be an encrypted form of the digest of the user’s password. This design choice has the following advantages:

- Even if a hacker steals a token, the password cannot be extracted out of the token since it is hashed.
- Even if a hacker manages to create a new user account (a username and a password) in the system, he will still need a token to authenticate himself/herself, which is practically impossible. A hacker cannot produce this binary string corresponding to this password, because the digest is encrypted with the server’s private key.
- If a user loses his/her token, replacing the corresponding password with a new one and giving the user another token carrying the encrypted form of the new digest thwarts a hacker who will try to use the lost token when he gets the related password.

2.3 Code Obfuscation

The word *obfuscate* means “to confuse”. Code Obfuscation refers to confusing the program code. In other words, *Code Obfuscation* is the technical term for preventing others from reading the code. Obfuscation serves to increase the difficulty of decompilation, usually forcing someone who wants that information to use more costly forms of reverse engineering.

The idea of code obfuscation is to transform the program code in order to make it more difficult to understand without changing the functionality the program provides.

Security of obfuscation relies on sophistication of transformation, power of deobfuscation algorithms and resources available to deobfuscator.

The code obfuscation methods are categorized as follows:

- **Layout Obfuscation**: Aimed at making the code unreadable
- **Data Obfuscation**: Aimed at obscuring data and data structures like migrate storage, aggregation, ordering and encoding.
- **Control Flow Obfuscation**: Aimed at obfuscating the flow of execution which makes code more frustrating
- **Preventive Transformation:** This transformation type intends to stop decompilers and deobfuscators from working.

The advantages provided by code obfuscation methods are as follows:

- Code obfuscation preserves platform independency to an extent unlike most of other source code protecting techniques. The obfuscated program is packaged with an obfuscator program that can run the obfuscated application in different versions of a particular operating system.
- The obfuscator program also compresses the obfuscated application, which results in a smaller-sized application. Therefore, an obfuscated program can be downloaded faster on a network.
- Since the obfuscator program also encrypts the obfuscated application, there is no need to a specific hardware for encryption.

The followings are the disadvantages of code obfuscation:

- Since the obfuscated application should be deobfuscated before it is executed, the execution time of the application increases.
- Obfuscating a program needs special complex tools that are difficult to use.
- The program development effort increases when a code obfuscation phase is integrated into the programming development cycle.

- Since code obfuscation techniques use complex algorithms, more processing power may be needed in order to use these techniques.

In the ESIS System, the client application is obfuscated in order to prevent attackers to learn the hidden seed value. The use of this seed value is described in Section 3.3.2.

3 THE DESIGN OF THE ESIS SYSTEM

In this chapter, we propose a secure synchronization system as a solution to the problem of managing local administrator passwords in an enterprise network. The problem is defined in Section 3.1. An overview of the proposed system is given in Section 3.2. In Sections 3.3 and 3.4 , we describe the details of the system. We propose three secure protocols in Section 3.5, which constitute the base of the proposed system.

3.1 Problem Definition

A large enterprise computer network may consist of thousands of computers and may be formed of subnetworks that may be geographically separated. The maintenance of such a large network requires a work force specialized on computer hardware and software. Some companies in public and private sectors carry out maintenance work by in-house IT departments established for this purpose, whereas some other companies and organizations outsource this work to computer support and maintenance firms that have more experience in this field. Companies and organizations outsource their maintenance tasks in order to focus on their core businesses and to decrease their operational costs.

Maintenance of a distributed enterprise computer network involves the following operations:

- Configuration of network equipments such as routers, switches.
- Installation, configuration, and repair of operating systems, database, email servers, and other software

- Installation, configuration, and repairing computer peripherals such as modems, ethernet or video cards, etc.

In order to carry out maintenance operations specific to a computer, a maintenance technician needs to gain administrative rights by using administrative account credentials. In Windows operating systems (Win NT, Win2000, WinXP), an administrative account can be a local or a domain[1] account. A local administrator account can only be used for a computer but a domain administrative account has administrative access to all of the computers in the domain.

The username and password in encoded forms of the local administrator account are stored in the local computer, whereas domain account information is stored in a central domain server. In many cases, the technician needs the local administrator password because domain passwords cannot be verified against the domain server if the computer is not connected to the network. There might be different policies about the maintenance of local administrator passwords.

Every computer may be assigned a different random local administrator password. This method would prevent the whole network from being compromised when only one local administrator password has been revealed. However, it is practically impossible for a technician to memorize and recall all of the passwords of the computers in an enterprise network. Therefore, a technician will need to carry a list of the computers and their corresponding local administrator passwords on either paper or digital media. Changes in the local administrator passwords should be reflected to every technician's list. On the other hand, a static list of passwords would be very insecure as stated in [21].

Another method is to group the computers into a few numbers (the group size may be between 3 and 7 computers, depending on the network size) of groups and every group can be assigned a different local administrator password. As a result, the number

[1] A Windows Domain is a logical grouping of computers that share common security and user account information. This information is stored in a master directory database (SAM) which resides on a Windows server designated as a domain controller.

of passwords a technician should recall is decreased to a level where every technician can remember the passwords easily. Although this method is quite convenient and feasible, compromise of a group password results in compromise of all the computers in that group.

A more convenient, but very insecure, method would be assigning a unique password or generic passwords generated by using a simple rule to the computers. The generic password may be generated by adding the first three characters of the computer name at the end of a constant string such as “clientpwd”. However, compromise of the unique password or the rule used in generating a password results in the compromise of all the computers.

Methods described above are usually implemented in organizations lacking a secure automation system that periodically updates passwords and provide a query interface to technicians to obtain the current local administrator password of a computer. Such organizations encounter the following security risks when they outsource computer maintenance:

- Technicians are authorized users that can perform every operation on the computers without being tracked. Since technicians use the same local administrator account to login a specific computer and there is no relation between a technician and a local administrator account, activity log recorded by the operating system of the computer cannot be used for tracking technicians.
- There is a common business fact that secret information for an organization must not be known by outsiders. Technicians working for the computer support firm may change frequently due to the job rotation in the service firm. Therefore, local administrator passwords should be changed according to the change in the technician staff, which is a difficult and tedious operation to perform in an enterprise network.
- Passwords cannot be changed instantly when a technician quits his/her job.
- Technicians may use memory aids (a note paper, a PDA) to remember the passwords. A memory aid can be lost or stolen which prevents all of the passwords from being secret.

- By the use of social engineering techniques as described in [22], a hacker may obtain the passwords and use these passwords until they expire. Since passwords aren't changed frequently, a hacker would have plenty of time to crack the whole network.

In order to avoid the risks mentioned above, a secure automation system should be employed in an enterprise network to manage the local administrator passwords. The system should provide the following benefits:

- Changing local administrator passwords periodically and securely without a human intervention
- Querying facility for technicians to learn the current local administrator password of a computer
- Tracking of technicians' querying activities

In the scope of this study, we have designed and developed a secure automation system that provides the benefits mentioned above. It comprises of secure synchronization and query applications and protocols. It is called Enterprise-Level Sensitive Information Security (ESIS) system and is implemented on Windows platform. Details of the system are given later in this chapter.

3.2 Overview of the System

We propose a secure automation system, called Enterprise-Level Sensitive Information Security (ESIS) system, as a solution to the problems stated in the previous section. Local administrator passwords are very sensitive information in an enterprise network, but there may be other types of information as sensitive as the local administrator password and must be secured. Therefore, we have chosen the term, "Sensitive Information" in naming our system in order to cover all types of sensitive information including the "local administrator password". Although our focus is on local administrator passwords, the proposed security protocols can be easily utilized in exchange and synchronization of other types of sensitive information, as well.

ESIS is a secure automation system that provides the following benefits as mentioned in the previous chapter:

- Changing local administrator passwords periodically and securely without a human intervention
- Querying facility for technicians to learn the current local administrator password of a computer
- Tracking of technicians' querying activities

In the ESIS system, computers that are called ESIS Clients[2] change their local administrator passwords in synchronization with a central server that is called ESIS Server[3]. As shown in Figure 3.1, synchronization between clients and the server is achieved by two protocols, ESSP (ESIS Secure Synchronization Protocol) and ECIP (ESSP Client Initialization Protocol).

[2] The term, "ESIS Client" is used to define a client computer in the ESIS System. The same term may be used for the software application running on client computers in the ESIS System throughout the text.

[3] The term, "ESIS Server" is used to define the central server that manages synchronization in the ESIS System. The same term may be used for the software application running on this server throughout the text.

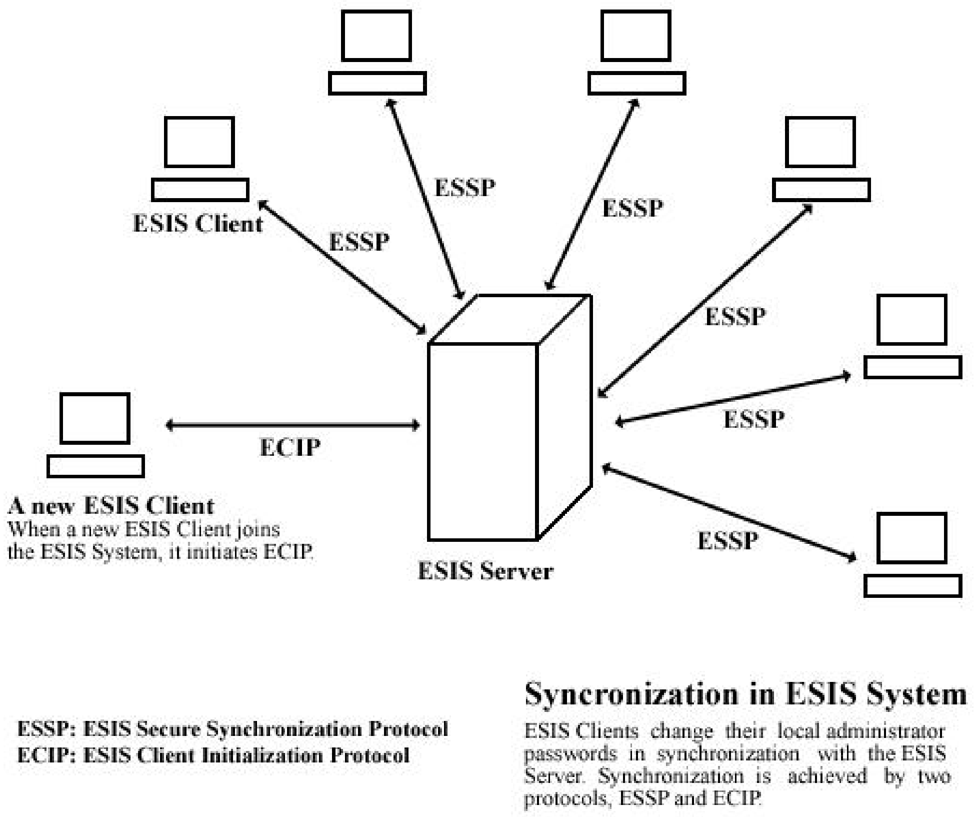

ESIS Client
ESSP
ESSP
ESSP
ESSP
ESSP
ESSP
ESSP
ECIP
A new ESIS Client
When a new ESIS Client joins the ESIS System, it initiates ECIP.
ESIS Server
Syncronization in ESIS System
ESIS Clients change their local administrator passwords in synchronization with the ESIS Server. Synchronization is achieved by two protocols, ESSP and ECIP.
ESSP: ESIS Secure Synchronization Protocol
ECIP: ESIS Client Initialization Protocol

Figure 3.1. Synchronization in ESIS System

The routine synchronization messaging between the ESIS server and clients is performed by ESSP in a secure way. The software applications, ESIS Server application running on the ESIS Server and ESIS Client application running on ESIS Clients, use ESSP in synchronization between them. The client software application initiates ECIP when it is first started; synchronization with the ESIS server is initiated with this protocol.

Every technician has a unique username and a password associated with this username and they are given tokens in which the tiny querying software application, ESIS Teller is located. There are also some data to be used in authentication by the application in this token. In order to learn the current password of an ESIS Client, a technician needs to authenticate himself/herself by using the querying application in the token. Authentication and querying operations are accomplished by a secure querying protocol, ESQP (ESIS Secure Querying Protocol). Authentication of a technician is a multi-factor authentication in which a password-username combination (What-You-

Know type authenticator) and a token (What-You-Have type authenticator) are used together. Querying local administrator passwords is performed as shown in Figure 3.2.

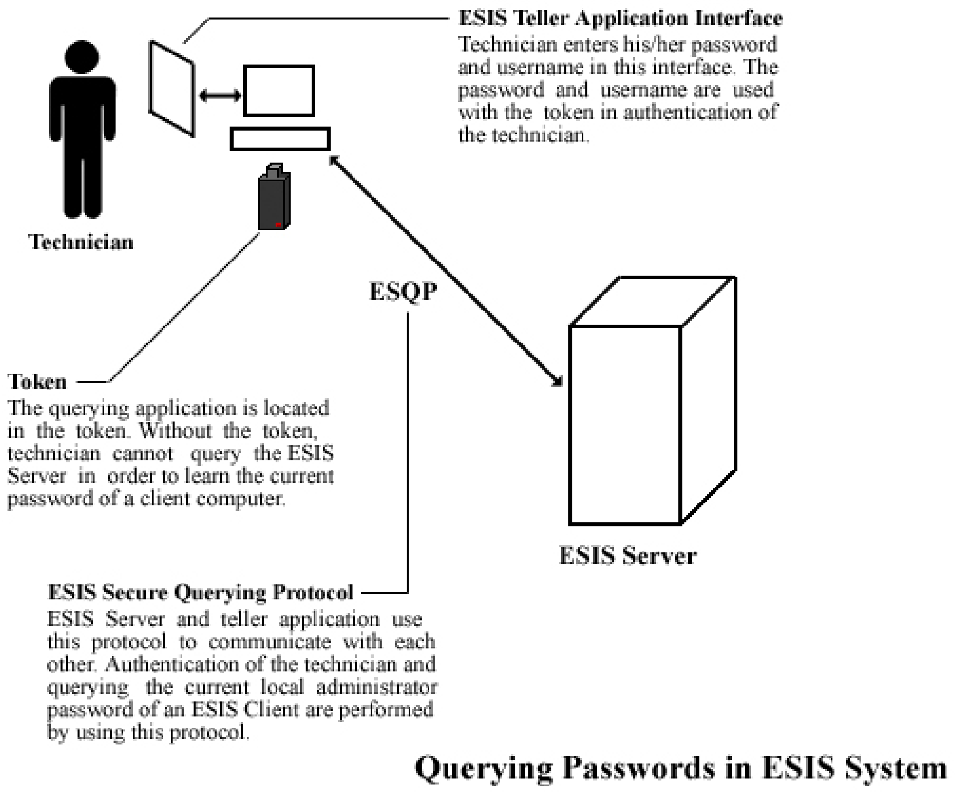

ESIS Teller Application Interface
Technician enters his/her password and username in this interface. The password and username are used with the token in authentication of the technician.
Technician
ESQP
Token
The querying application is located in the token. Without the token, technician cannot query the ESIS Server in order to learn the current password of a client computer.
ESIS Server
ESIS Secure Querying Protocol
ESIS Server and teller application use this protocol to communicate with each other. Authentication of the technician and querying the current local administrator password of an ESIS Client are performed by using this protocol.
Querying Passwords in ESIS System

Figure 3.2. Querying local administrator passwords in the ESIS System

3.3 System Components and Their Functions

The details of the ESIS System components and their functions will be given in this section before delving into the details of the protocols and the applications that have been designed and developed in the scope of this study.

There are three main components of the system: the client, the server, and the technician. The server manages the synchronization of these local administrator passwords and provides tracking features. There are many clients. These are the computers in the network of which local administrator passwords are to be updated periodically. The clients run the ESIS Client Application to change their passwords periodically in synchronization with the server. The technician is the only user that interacts with the system in order to learn the current password of a particular client.

The definition of the system components are based on their physical existences. In fact, there are software applications and protocols in the background, but we have chosen this way to be able to clearly separate the components from each other.

3.3.1 Server

The server, called ESIS Server, handles the following operations in the ESIS System:

- **Management of synchronized password generation**: The clients in the system automatically change their local administrator passwords under the control of the ESIS Server. ESIS Server application communicates with the ESIS Client applications using the secure synchronization protocol, ESSP as depicted in Figure 3.1. It sends each client a different 256-bit random seed value. A client uses this value in generation of a new password that will be replaced with the current password at the time specified by the server. The details will be given later in this section.
- **Responding to the technician's querying requests:** An authorized technician can query the server to learn the current administrator password of a client. He/she uses a tiny (≈300kB) software application that is called ESIS Teller Application and placed in his/her token in order to query the ESIS Server application. The protocol used between the server and the technician's application is a secure protocol named ESQP which is shown in Figure 3.2.

- **Tracking technicians' querying activities**: Since technicians must query the server in order to learn the current administrator password of a client computer, he/she needs to enter his/her username and password in the login form interface provided by the ESIS Teller Application. The server logs all of the queries performed by technicians with their usernames and the date of querying, so that every query made by a technician can be reported easily because every technician possesses a unique username.

In the server, the following data are stored:

- **Client Data**: For each client in the system, data such as client id, IP address, current password, and expiration time of the current password are stored. There are also other fields that are used in the ESIS Server Application and all of the parameters are shown in Table 3.1.
- **System Configuration Parameters**: These are the configuration parameters for the ESIS Server Application, which are used to tune the system performance. These parameters are shown in Table 3.2.
- **Server's RSA Public and Private Key Pair**: The secure querying protocol, ESQP, involves public key encryption operations. An RSA public and private pair to be used in these operations is stored in the server.
- **Technician Accounts:** Technicians' usernames, digests of their passwords and token ids are stored. Authentication of the technicians using the system is performed by using these data.
- **Tracking Data:** Technicians' querying activities are logged by the ESIS Server Application. Activity data involves the client id, the technician's username, IP address of the computer that the technician uses to query the server, and the querying date and time.

Client Data	
Client ID:	This is a unique identifier. Two clients cannot be represented by the same identifier.
IP Address:	The ESIS Server uses an IP address in order to initiate the secure synchronization protocol, ESSP with the corresponding client.
Current Password:	All the current local administrator passwords of the clients in the system are stored in the server. The server changes the current password of the corresponding client in synchronization with the client.
Seed:	This is a 256-bit binary sequence which is used as the symmetric encryption key in the secure synchronization protocol, ESSP. This value is updated in each successful synchronization messaging with a new seed that is generated by XORing a random 256-bit binary sequence with the current seed. When a new seed is sent to the client, a new synchronization is started at the server and the client side and the state of the client is set to 1. The initial value of the seed is located in each ESIS Client Application. It is obfuscated in the ESIS Client Application code and used in the ESIS Client Initialization Protocol, ESQP that will be explained in detail later in this chapter.
Activation Time of the Synchronization:	This is the starting time of the synchronization phase.
Expiration Time of the Current Password:	Current local administrator password will expire at the expiration time specified by this value.
Synchronization State of the Client:	An integer value represents the state of a client as follows: **Awaiting (0)**: The client has changed its password and is now waiting to be connected and sent a new seed value to start the synchronization phase **Aware (1)**: The client has been sent a seed value. **Lost (2 or an even integer)**: When the server cannot send the new seed to the client, the state of the client is updated by incrementing the current state value by 2. **Dead**: The state of a particular client that cannot be synchronized for a specified number of times in a row becomes Dead. The server stops attempting to synchronize with that client. The state of a client that joins the network is 0.
Number of Password Generation Rounds in a Synchronization Phase:	After the synchronization protocol is carried out between the client and the server, a synchronization phase begins at the activation time specified by the server. In this phase, each side changes the local administrator password a number of times. This number is the number of password generation rounds in a synchronization phase.
Number of Remaining Rounds in a Synchronization Phase:	As the server and the client updates the local administrator password, the number of the remaining rounds is decremented by one. This value is equal to the number of password generation rounds in the beginning of a synchronization phase.
Initial Seed of a Client:	As mentioned in the definition of the seed value, this value is the initial seed value that is obfuscated in the client application code.
Last Synchronization Time:	This is the time of the last successful update of a local administrator password of the client.

Time of Last Successful Sending of a Seed:	This is the time the server has successfully finished the synchronization protocol.
Time of The Next Seeding Attempt:	This is the time the server starts the synchronization protocol with the client

Table 3.1. ESIS Client data

ESIS System Configuration Parameters	
Seeding Interval	At each seeding interval, the server application starts the synchronization protocol, ESSP with the client at state 0.
Expiration Interval	This is the length of the interval between password change rounds in a synchronization phase in seconds.
Password Update Interval	At each password update interval, the server updates the local administrator passwords of the clients whose corresponding expiration times are later than the present time.
Password Length	The generated password is at this length.
Maximum Number of Seeding Attempts	If a client cannot be reached by the client, the server tries to reach that client at most that number of times. If the server cannot reach a client that times in a row, the state of that client becomes Dead.
Lost Client Seeding Interval	If a client cannot be reached by the client, the next time the server tries to reach that client until that amount of time elapses. The details about the implementation are in Section 4.1.1.1.

Table 3.2. ESIS System configuration parameters

3.3.2 Client

The client is a computer running ESIS Client Application to replace the current password with a different password generated by a seed value, which has been sent by the server, periodically and in synchronization with the server. In the client initialization phase, the client connects to the server in order to establish the secure initialization protocol. The protocol messages are encrypted with a 256-bit AES key called $seed_{obf}$ that is obfuscated in the code to prevent it from being revealed as a result of a reverse-engineering attempt.

The client may be disconnected from the network because of a software or hardware problem, which results in a broken connection between the server and the client. In this case, the synchronization between the server and the client halts, which means the client stops changing its current password. Since the last current password of the client stored in the server is the valid one, the technician who has queried the server to learn the password can login to the client computer.

The ESIS Client Application runs in the background on every client. This application is a socket application listening on a specific port to get the synchronization protocol packets from the server. We have implemented this application as a Windows service application that will be examined in Section 4.1.2.

3.3.3 Technician

The technician is the only user that interacts with the ESIS System as shown in Figure 3.2. Every technician knows/has the followings:

- **Username:** Every technician is assigned a unique username. Every username is registered in the server.
- **Password:** Each username is associated with a password. The digests of the passwords are stored in the server. The technician must know his/her username/password pair.
- **Token:** To strengthen the authentication process in the secure querying protocol, ESQP, a token is employed together with a username and a password. Every technician is given a token. As mentioned in the previous chapter, this type of authentication is called multi-factor authentication and it is integrated in the system in order to increase the security level. A tiny querying software application, the public key of the server, and encrypted form of the digest of the password are located in token before it is given to a technician. Since the digest of the technician's password is encrypted with the RSA private key of the server, a hacker cannot produce this value in order to use a different password. Moreover, a hacker cannot learn the password even if he/she steals a token.

When there is a problem in one of the client computers, a technician plug his/her token into a computer from which the ESIS Server can be reached over the enterprise network. Then, he/she uses the token application in order to connect and query the server to obtain the password of the client computer. In our implementation, the token is a USB memory stick that can be easily found in a computer store and the software application is a pure Win32 application which has been developed to run on Windows operating systems.

3.4 Synchronized Password Generation

Synchronized password generation takes place at the client and server sides. The client changes its password at the same time when the server updates the password of the client in the database with the new password.

The ESIS Server controls the synchronization of local administrator passwords of the ESIS Clients. The synchronization between the client and the server consists of the following three phases:

- ESIS Client Initialization
- Password Synchronization Phase
- Delivering Synchronization Parameters

After ESIS Client Initialization, the client gets the initial synchronization parameters. At the time specified by the server, the password synchronization phase starts. After this phase has been completed by the server and the client in a synchronized way, the server initiates the secure synchronization protocol, ESSP, to send the parameters of the next synchronization password generation phase. The second and the third phases follow each other as shown in Figure 3.3.

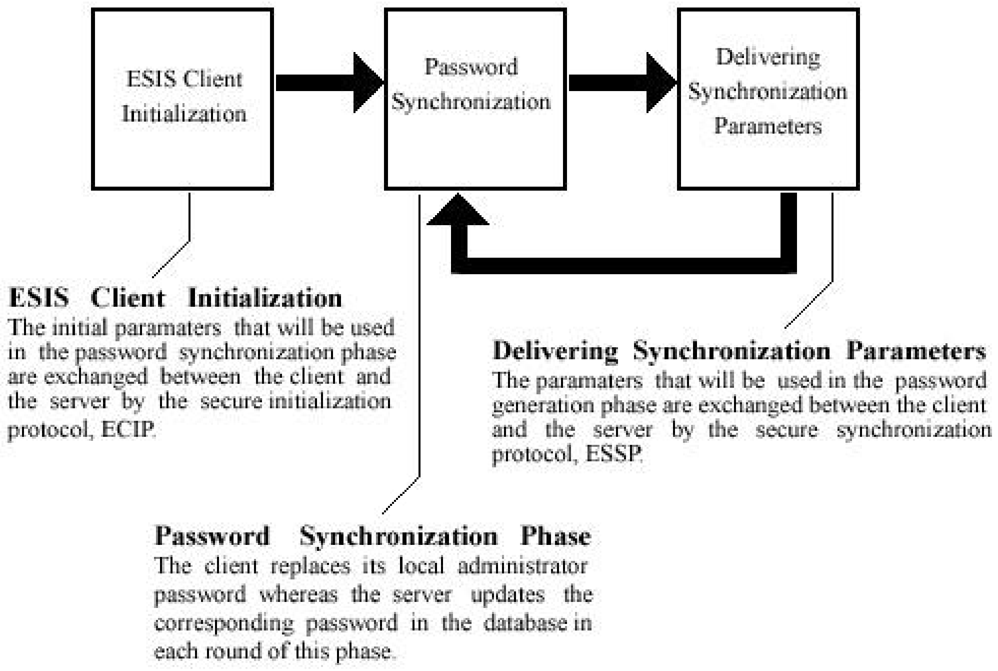

ESIS Client Initialization
Password Synchronization
Delivering Synchronization Parameters
ESIS Client Initialization
The initial paramaters that will be used in the password synchronization phase are exchanged between the client and the server by the secure initialization protocol, ECIP.
Delivering Synchronization Parameters
The paramaters that will be used in the password generation phase are exchanged between the client and the server by the secure synchronization protocol, ESSP.
Password Synchronization Phase
The client replaces its local administrator password whereas the server updates the corresponding password in the database in each round of this phase.

Figure 3.3. Synchronization phases

The first phase, ESIS Client Initialization, is only initiated by the client when the ESIS Client Application is started. To restart the application on the client will cause the client to be initialized again. Therefore, a technician can restart the client application in order to start synchronized password generation again between the client and the server if a synchronization problem occurs. Since we have implemented the ESIS Client Application as a Windows service, it starts running automatically when the computer is restarted. This is a useful property because when a technician cannot login with the current password he/she has obtained from the server, he/she restarts the computer in order to establish the synchronization between the server and then queries the server for the local administrator password.

The Password Synchronization Phase follows the Client Initialization and Delivering Synchronization Parameters phases. After the initialization of a particular client, password synchronization is accomplished in that client and the server after a specified amount of time after the server sends the synchronization parameters to that client. This loop as shown in Figure 3.3 lasts until the client application is restarted, which causes the corresponding client to be initialized.

In the following sections, three phases of the synchronized password generation and the client synchronization states will be described in detail. For the sake of clarity, ESIS Client Initialization and Delivering Synchronization Parameters phases will be explained before the Password Synchronization Phase because it depends on the synchronization parameters delivered to the client in those phases.

3.4.1 Synchronization States of Client

The clients may be in different states at different times. The state of a client that has just received the synchronization parameters is different from a client that has completed the password synchronization phase. The synchronization states are represented by an integer value in the server. There are four types of client synchronization state in the ESIS System as follows:

- **Awaiting**: The clients entering the Client Initialization phase or completing the Password Synchronization phase are in this state. Periodically, the server sends the synchronization parameters to the clients that are in the Awaiting state. This state is represented by 0 in the database.
- **Aware**: After a successful Delivering Synchronization Parameters or a Client Initialization phase, the synchronization state of a client becomes Aware. The Aware state is kept until the Password Synchronization Phase is completed by the server and the client. This state is represented by 1 in the database.
- **Lost:** The server and a particular client cannot be synchronized successfully after the server initiates the secure protocol (Delivering Synchronization Parameters phase). This may occur because of general network errors or hardware or software problems in the client. In this case, the server sets the synchronization state of the client to the Lost state, which is represented by an even integer greater or equal to 2.
- **Dead:** If the server cannot reach a client in the Lost state for a number of times that is defined as a system parameter, Maximum Number of Seeding Attempt Times, its state becomes Dead. The integer value representing this state is two times the Maximum Number of Seeding Attempt times.

3.4.2 ESIS Client Initialization Phase

ESIS Client Initialization and its relation with the password synchronization phase is depicted in Figure 3.4. The steps in this phase are described as follows:

- **Init_ECP()**: The ESIS Client joins the ESIS System, which means the ESIS Client Application is started on the client. As the application starts, the client initiates the secure initialization protocol, ECIP.
- **SendSynchParams(clientid$_k$, seed$_0$, t$_m$, r, interval)**: The server receives the initialization request message that includes kth client's id. If the server authenticates the client id, it sends the corresponding client the following synchronization parameters:

  seed$_0$: This is the initial 256-bit pseudo-random binary sequence that will be used in generation of the local administrator password. This seed will be the symmetric encryption key in the next secure synchronization protocol, ESSP. The seed is generated by the server as shown below:

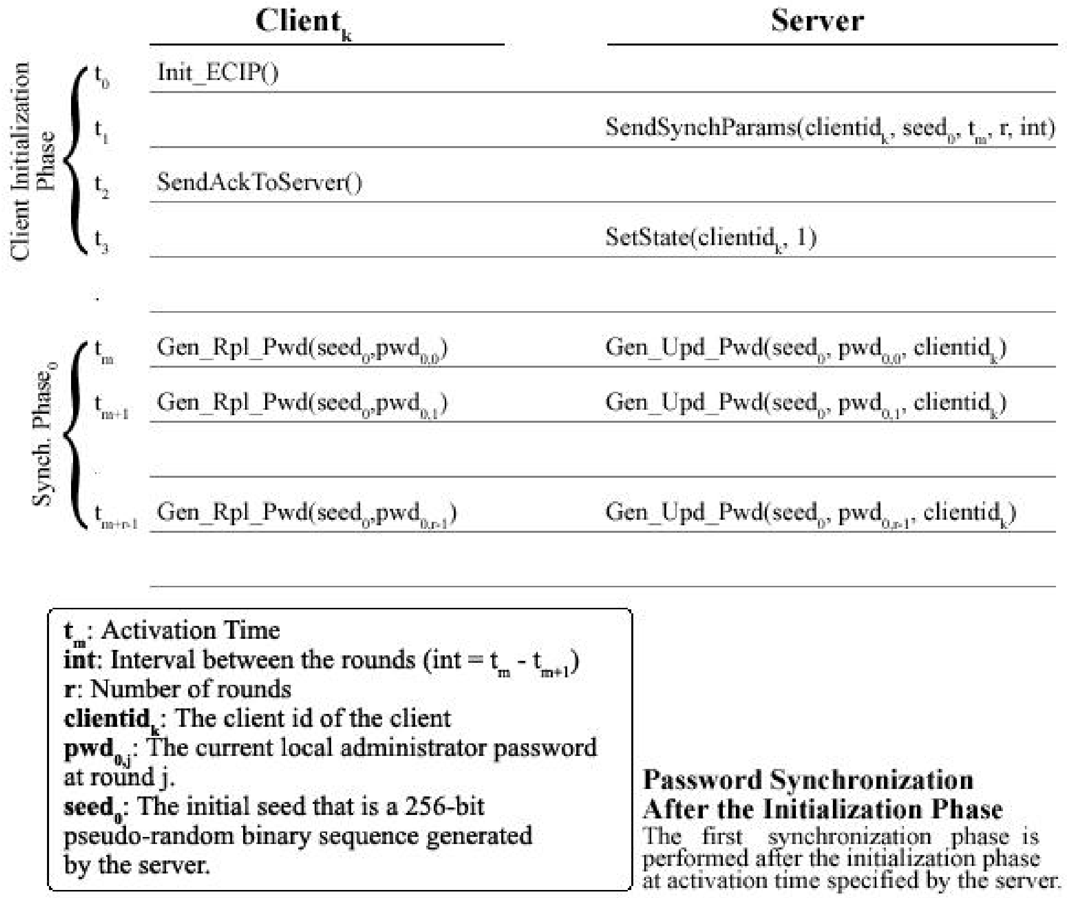
  $$seed_0 = seed_{obf} \oplus seq_0$$

  where $\oplus$ is bitwise XOR operator. seed$_{obf}$ is a 256-bit binary sequence obfuscated in the client application code whereas it is stored in the database as the initial seed of the client. seq$_0$ is a pseudo-random value generated by the server.

  t$_m$: This is the activation time at which the password synchronization is started.
  r: This is the number of password synchronization rounds. The local administrator password will be changed r times in the scope of the corresponding synchronization phase.

  interval: This is the duration between the synchronization phase rounds in seconds.
- **SendAckToServer()**: If the client authenticates server's response, it sends an acknowledgement message to the server.
- **SetState(clientid$_k$,1):** If The server receives the acknowledgement message and authenticates it successfully, it updates the state of the client to Aware state (1)

in the database, which means the client has been sent synchronization parameters and is ready for the synchronization phase.



Client$_k$
Server
Client Initialization Phase
t_0
Init_ECIP()
t_1
SendSynchParams(clientid$_k$, seed$_0$, t_m, r, int)
t_2
SendAckToServer()
t_3
SetState(clientid$_k$, 1)
Synch. Phase$_0$
t_m
Gen_Rpl_Pwd(seed$_0$,pwd$_{0,0}$)
Gen_Upd_Pwd(seed$_0$, pwd$_{0,0}$, clientid$_k$)
t_{m+1}
Gen_Rpl_Pwd(seed$_0$,pwd$_{0,1}$)
Gen_Upd_Pwd(seed$_0$, pwd$_{0,1}$, clientid$_k$)
t_{m+r-1}
Gen_Rpl_Pwd(seed$_0$,pwd$_{0,r-1}$)
Gen_Upd_Pwd(seed$_0$, pwd$_{0,r-1}$, clientid$_k$)
t_m: Activation Time
int: Interval between the rounds (int = t_m - t_{m+1})
r: Number of rounds
clientid$_k$: The client id of the client
pwd$_{0,j}$: The current local administrator password at round j.
seed$_0$: The initial seed that is a 256-bit pseudo-random binary sequence generated by the server.
Password Synchronization After the Initialization Phase
The first synchronization phase is performed after the initialization phase at activation time specified by the server.

Figure 3.4. Password synchronization after ESIS Client Initialization

This phase depends on the secure client initialization protocol, ECIP and it will be given later in this chapter.

3.4.3 Delivering Synchronization Parameters Phase

This phase is initiated by the server and depends on the secure synchronization protocol, ESSP that will be explained later in this chapter.

The Password Synchronization Phase follows this phase as shown in Figure 3.3 and the relation between them is shown in Figure 3.5.

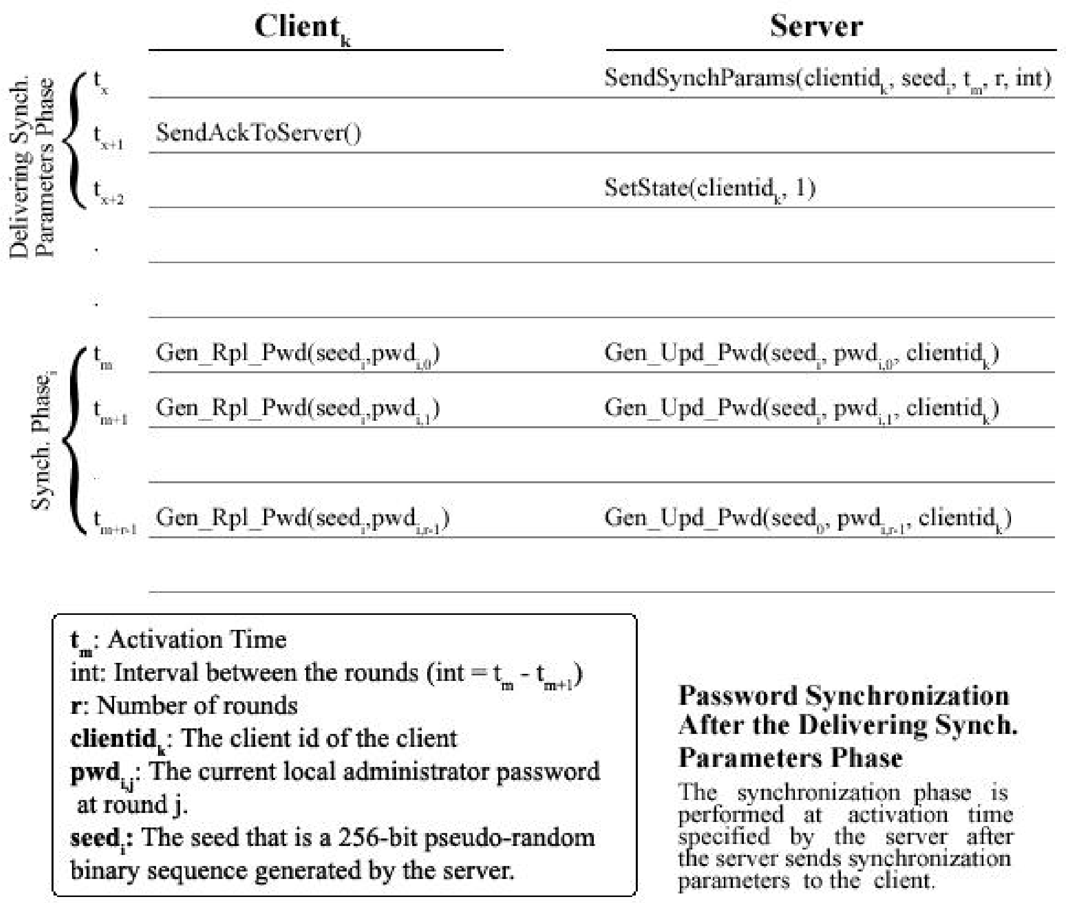

Client$_k$
Server
Delivering Synch. Parameters Phase
t_x
SendSynchParams(clientid$_k$, seed$_i$, t_m, r, int)
t_{x+1}
SendAckToServer()
t_{x+2}
SetState(clientid$_k$, 1)
Synch. Phase$_i$
t_m
Gen_Rpl_Pwd(seed$_i$,pwd$_{i,0}$)
Gen_Upd_Pwd(seed$_i$, pwd$_{i,0}$, clientid$_k$)
t_{m+1}
Gen_Rpl_Pwd(seed$_i$,pwd$_{i,1}$)
Gen_Upd_Pwd(seed$_i$, pwd$_{i,1}$, clientid$_k$)
t_{m+r-1}
Gen_Rpl_Pwd(seed$_i$,pwd$_{i,r-1}$)
Gen_Upd_Pwd(seed$_0$, pwd$_{i,r-1}$, clientid$_k$)
t_m: Activation Time
int: Interval between the rounds (int = t_m - t_{m+1})
r: Number of rounds
clientid$_k$: The client id of the client
pwd$_{i,j}$: The current local administrator password at round j.
seed$_i$: The seed that is a 256-bit pseudo-random binary sequence generated by the server.
Password Synchronization After the Delivering Synch. Parameters Phase
The synchronization phase is performed at activation time specified by the server after the server sends synchronization parameters to the client.

Figure 3.5. Password synchronization after Delivering Synchronization Parameters

The synchronization parameters that are sent to the client in this phase are the same with those sent in the client initialization phase, but it is initiated by the server whereas the client initialization phase is initiated by the client. In this phase, the server generates the seed value that will be used in the i^{th} synchronization phase, which is calculated as follows.

$$seed_i = seed_{i-1} \oplus seq_i$$

where $\oplus$ is bitwise XOR operator, and seq_i is a pseudo-random value.

3.4.4 Password Synchronization Phase

In the first and the third phases, the server sends the synchronization parameters to the client. Since the client and the server know the parameters, the generated password in this phase becomes the same at both sides.

The password synchronization phase comprises one or more rounds as shown in Figure 3.4 and Figure 3.5. The server sends the number of rounds to the client among the synchronization parameters. In these rounds, the server and the client perform the following operations as shown in Figure 3.5:

- **Gen_Rpl_Pwd(seed$_i$, pwd$_{i,j}$)**: The client replaces its current local administrator password with the new one generated by this function. The new password is generated as follows:

$$\mathrm{pwd}_{i,j+1} = \mathrm{pwd}_{i,j+1} \oplus \mathrm{seed}_i$$

- **Gen_Upd_Pwd(seed$_i$, pwd$_{i,j}$, clientid$_k$)**: The new password is generated in the same way as in the function, Gen_Rpl_Pwd. The server updates the local administrator password of the client in the database and also sets the state of the corresponding client to the Awaiting state (0).

The password synchronization starts at the activation time, t_m as shown in Figure 3.5. Therefore, system clock of all the clients and the server must be synchronized in order to establish synchronization in the ESIS System. In an enterprise network, this can be achieved by providing a time synchronization service. If this service fails, the password synchronization in the ESIS System will fail accordingly. Hence, this service should be robust and scalable.

3.5 Protocols

The building blocks of a secure authentication system are carefully designed and flawless security protocols. The ESIS System comprises of three protocols, ESIS Secure Synchronization Protocol (ESSP), ESIS Secure Querying Protocol (ESQP), and ESIS Client Initialization Protocol (ECIP). The local administrator password synchronization between the server and the clients depends on the ESSP and the ECIP as shown in Figure 3.1.

In Section 3.5.1, all of the abbreviations used in the protocol definitions are explained. In Sections 3.5.2, 3.5.3, and 3.5.4, the secure protocols employed in the ESIS System are explained in detail.

3.5.1 Abbreviations

The following list of abbreviations are used in the protocols.

T: Technician

C: Client computer

un: Technician's user name

$\mathbf{pwd_{technician}}$: The password that the technician enters in the software.

$\mathbf{h(pwd_{token})}$: The hashed form of the password which is located the technician's token.

TS: A timestamp value used in the protocols in order to thwart replay attacks[4] on them.

$\mathbf{E_k}$: The symmetric key encryption routine using an AES key k.

$\mathbf{HMAC_{key}}$: The message authentication routine using a symmetric key

tid: Token Id

[4] A hacker can intercept an encrypted message in order to send it to the destination later for unauthorized access. This type of an attack is called replay attack.

E_{pub}: The RSA public key encryption routine of the server using its public key.

E_{priv}: The RSA private key encryption routine of the server using its private key.

compName: The name of the client computer whose password is requested. (Example: abc.companydomain.com)

locAdmPwd: Current local administrator password of a client. The technician needs the current local administrator password of a client in order to login to it.

nextSeed: The seed value that will be used in the next numberOfRounds password generations.

activationTime: The starting time of a password synchronization phase.
numberOfRounds: The number of rounds of a password synchronization phase.

interval: The time in seconds between the rounds in a password synchronization phase.

clientID: An identifier string that uniquely identifies the computer in the network. This could be the same as the computer name such as abc.companydomain.com.

$seed_{obf}$: A 256-bit binary sequence which is obfuscated in the ESIS Client Application code and stored in the server. This is employed as the symmetric key in the client initialization protocol, ECIP.

K_{ses}: A symmetric key used during a session between two communicating parties (a session key).

E_{ses}: The symmetric key encryption routine using the session key K_{ses}.

3.5.2 ESIS Client Initialization Protocol (ECIP)

Due to the various network or hardware problems, a client can disconnect from the network causing that the server cannot reach the client and cannot initiate the synchronization protocol (ESSP) with the client. The server attempts to reach the client for several times but after a specified number of times, the server stops attempting to reach and initiate the ESSP with the client. In this case, if the client service application is started at the client side, this client will never be contacted by the server because it’s known as dead by the server. Therefore, the client needs to initiate another protocol to establish the synchronization again. This protocol is the ESIS Client Initialization Protocol. The client service application initiates this protocol whenever it is started. In fact, this protocol can also be used as a notification protocol, which means that client has joined the network as shown in Figure 3.1 . The protocol is as follows:

1. C → S: $HMAC_{seedobf}$(clientId,TS) , clientId , TS
2. S → C: $E_{seedobf}$ (M_1) , $HMAC_{seedobf}$ (M_1)
3. C → S: $HMAC_{seedobf}$ (M_2)

In the first step, the client authenticates itself by sending its client id, a timestamp value, and the authentication code of the whole message. $seed_{obf}$ is used as the symmetric key in HMAC and AES encryption operations.

As a result of successful authentication of the client, the server prepares the message M_1 including the synchronization parameters. M_1 is as follows:

M_1 = TS, nextSeed, activationTime, clientID, numberOfRounds, interval

The synchronization parameters nextSeed, activationTime, numberOfRounds, and interval are explained in Section 3.4.2. The timestamp, TS is put in the messages in order to thwart replay attacks. Binding the message to the time prevents a hacker from using the same message later for unauthorized access. Later, the server refuses the message because of its invalid timestamp. The server encrypts the message and generates authentication code of the message by using the client's $seed_{obf}$ as the symmetric key. The server sends the encrypted message and the message authentication code to the client in the second step of the protocol. Encryption and computing the message authentication code provide confidentiality and authentication of the message.

After the client receives and authenticates M_1, the client prepares the acknowledgement message, M_2 in order to send to the server. The client becomes ready for the password synchronization phase as shown in Figure 3.4. M_2 is as follows:

M_2 = TS+1, nextSeed, activationTime, clientID, numberOfRounds, interval

Since the server knows how M_2 is computed and the symmetric key, it can authenticate the message. On the other hand, M_1 may also be used as the acknowledgement message but in this case, a hacker can obtain it in the second step of the protocol and use it to pretend like the client. Therefore, to differentiate the acknowledgement message, timestamp in M_1 is incremented. A hacker cannot replay this message because the encryption key, $seed_{obf}$ is just known to the client and the server. If the server authenticates M_2, it sets the state of the client to the Aware state and updates the synchronization data of the corresponding client.

3.5.3 ESIS Secure Synchronization Protocol (ESSP)

As shown in Figure 3.5, the server delivers the synchronization parameters to the clients using the secure synchronization protocol, ESSP.

The synchronization protocol involves two steps:

1. $S \rightarrow C$: $E_{seed}(M_1)$, $HMAC_{seed}(M_1)$
2. $C \rightarrow S$: $HMAC_{seed}(M_2)$

Steps of the protocol are the same as the second and third steps of the client initialization protocol, ECIP. The contents of M_1 and M_2 are given in Section 3.5.2.

In the first step of the protocol, the server sends the synchronization parameters to the client in the Awaiting state. The server encrypts the message, M_1, including the synchronization parameters and obtains the message authentication code of M_1. The symmetric key used in encryption and message authentication function is the current seed value that is known to both client and server. M_1 is as follows:

$$M_1 = \text{TS, nextSeed, activationTime, clientID, numberOfRounds, interval}$$

The client decrypts the ciphertext in order to get M_1 before computing its message authentication code. It authenticates the server by comparing the message authentication codes, the computed one and the other sent in the first step of the protocol. After a successful authentication, the client sends the message authentication code of M_2 as an acknowledgement message. M_2 is as follows:

M_2 = TS+1, nextSeed, activationTime, clientID, numberOfRounds, interval

Practically, the message authentication code of M_1 may be sent to the server as the acknowledgement message allowing a hacker to pretend like the client because he/she can obtain it in the previous (first) step of the protocol. Hence, another practical but a secure way is to increment the timestamp in M_1 in order to get M_2. Since the current seed is just known to the client and the server, the message authentication code of M_2 cannot be replayed by an attacker. Afterwards, the server sets the state of the client to the Aware state and updates the synchronization data of the corresponding client if it authenticates the client in the second step of the protocol.

The size of the encrypted message is 128 bytes whereas the size of the message authentication code is 20 bytes. Since there are two message authentication codes and one encrypted message sent in the protocol, the total size of the messages exchanged between the participants is 168 bytes (1344 bits).

3.5.4 ESIS Secure Querying Protocol (ESQP)

Technicians can query the server with the application located in their tokens. The messaging between the token application and the server application is performed by the secure querying protocol, ESQP as shown in Figure 3.2. The protocol ESQP consists of the following steps:

1. T → C: E_{pub}(un, $pwd_{technician}$, TS, K_{ses}, tid) , E_{priv}(h(pwd_{token}))
2. C → T: E_{ses}(TS)
3. T → C: E_{ses}(compName, TS)
4. C → T: E_{ses}(locAdmPwd,TS,compName,expirationTime)
5. T → C: E_{ses}("kill", TS)

The first step is the authentication or identification of the technician. The technician's username, timestamp value, token ID and a session key generated by the token application is encrypted with the public key of the server and then this encrypted message is sent to the server with the technician's password that is signed by the server's private key. The technician is authenticated when the two conditions are met:

1. $E_{pub}(E_{priv}(h(pwd_{token})) \overset{?}{=} h(pwd_{technician})$
2. $h(pwd_{technician}) \overset{?}{=} GetPassword(un, tid)$

where GetPassword(un, tid) is a function which returns the password associated with the username and token id which are stored in the database on the server.

If the first condition is satisfied, it is verified that the password digitally signed and embedded in the token and the one the technician uses are same. The digital signature of the password is embedded into the token by a system administrator that has privilege to use the private key of the server before it is given to a technician. Since nobody except the system administrator can reproduce the signature of the password without the knowledge of the private key, the first condition provides the server the ability to control the identity binding between the token and the technician. The technician may quit his/her job without giving his/her token back to the system administrator, but he/she must be prevented from using the system. This need can be satisfied by the second condition. If the system administrator changes the technician’s password or even deletes his/her entry from the database, the second condition fails and the technician is not authenticated.

The second step of the protocol is the identification of the server. In the second step the server encrypts the timestamp with the session key and sends it back. Since no one knows the session key, the encrypted message itself is an adequate indicator for the technician that the server has received the message and is the owner of the message.

After the authentication phase (steps 1 and 2) is completed, the technician now can query the server to obtain passwords of the computers. In the third step, the technician requests the password of a client computer from the server. The corresponding computer name is sent to the server together with the timestamp. The server receives the ciphertext and decrypts it in order to get the computer name and the timestamp. Since successful decryption provides authentication as well as confidentiality, the server can authenticate the message by decrypting it successfully. As a result of this request, the server finds the current password of the client computer from the database.

The fourth step of the protocol is the secure password transfer. The server sends the current local administrator password, the corresponding computer name, the timestamp value and the expiration time of the password in an encrypted form after it encrypts them by using the session key. The technician receives and authenticates the message by decrypting it.

The technician can use the current local administrator password of the computer until the expiration time; otherwise, he/she must query the server again in order to get the new current local administrator password and its expiration time. Moreover, the technician may need the current local administrator passwords of some other computers. Since the technician is already authenticated in the first and the second steps of the protocol, there is no need to repeat these steps in each query. So, only the third and the fourth steps are repeated through the session. Since the timestamp is put in the encrypted message, a hacker cannot perform a replay attack in order to get a response from the server. A session is closed automatically by the server after a period of time (Ex: 20 min.) in which the technician stays inactive or after a total amount of session time (Ex: 60 min.).

The technician may need to close the session whenever he wants after the fourth step. The fifth step is to deliberately close the session. The technician sends a “kill” command in an encrypted form. If the server decrypts the encrypted message successfully, it authenticates the message and closes the session. As in the previous steps, the timestamp is also sent together with the command in order to prevent replay attacks so that a hacker cannot close a session without knowing the session key. When the session is closed, the server updates the session record on the database.

3.6 Discussion

The security of the system depends not only on the secure protocols but also on the implementation details. Some open issues need to be addressed in the implementation in order to fill the security gaps of the system.

A real problem that may occur in the implementation is that the technician can close the client service application because he/she logs into the computer using the administrator password. Afterwards, he/she can even change the local administrator password. This problem may be observed by examining the relationship between the time when a client gets to the Dead state and the time when a technician queries the server for the local administrator password of that client. If they are very close to each other, this may be a clue showing that the technician has closed the service application.

Another issue is that the technician may give his/her token, user name, and password to a third person. Giving all the authenticators to the third person allows him/her to use the ESIS System. This situation may be observed by analyzing the log data of the technicians' querying activities. If there are illogical entries in the log data such that a technician username and password might have been used at almost the same time but at different places, this shows a third person possesses technician's authenticators. This situation may be handled up to an extent by changing technicians' passwords frequently, so that the third person cannot use the system for a long time. However, since the content of the token must be updated when its owner's password is changed, there is a need for a practical method in order to update the tokens in a frequently manner.

4 APPLICATION

A set of software applications have been developed in the scope of this study:

- A server application, ESIS Server
- A Windows service, ESIS Client
- A Windows console application, ESIS Teller

In the following sections, these applications will be explained in detail.

4.1.1 The ESIS Server

ESIS Server has been developed in Visual Basic.NET to run on Microsoft.NET Framework. Microsoft.NET Framework is an application platform that runs on Windows based systems. The RDBMS has been chosen as MS SQL Server 2000. As shown in Figure 4.1, the server application consists of four concurrent threads; synchronization, password update, client initialization, and password query.

The ESIS Server Application has been developed as a Windows service application. Therefore, there is no need to start the application each time the server is restarted. Starting the service is the initialization of the application that results in starting the threads whereas stopping the service causes the threads to be stopped and destroyed.

In the Synchronization Thread, the server initiates the secure querying protocol, ESSP with the clients. The server updates the local administrator passwords of the clients in the Password Update Thread. The Client Initialization Thread handles the client initialization requests that the clients send in order to be synchronized with the server. The Password Query Thread handles technicians' query requests securely using the secure querying protocol, ESQP.

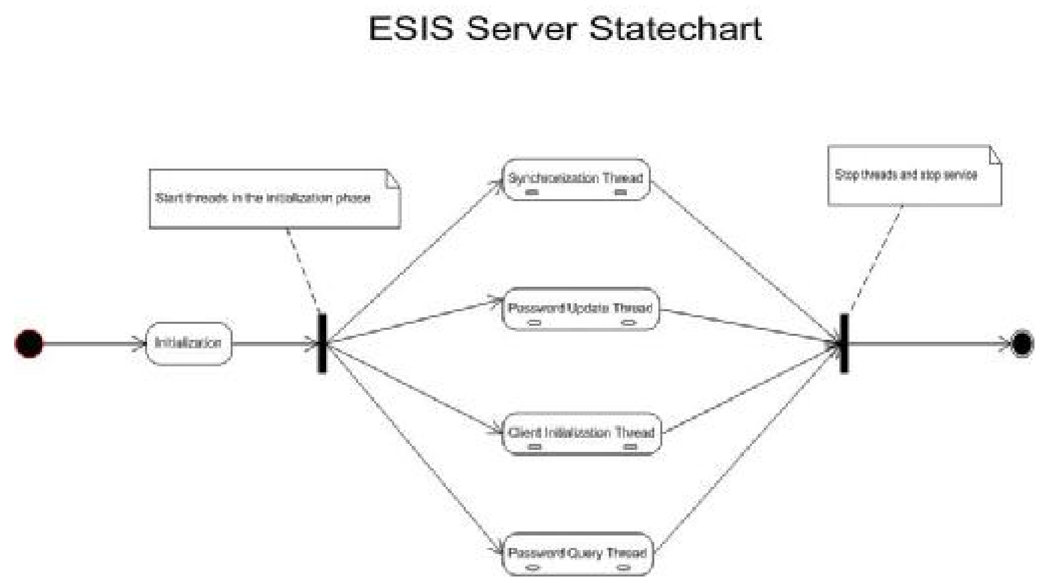


Figure 4.1. ESIS Server Statechart

4.1.1.1 Synchronization Thread

The Synchronization Thread handles delivering the synchronization parameters to the clients and performing the necessary actions after. The synchronization parameters are detailed in 3.4.

The server obtains the data of the clients that are in the Awaiting and the Lost states in order to send them the synchronization parameters.

Successful synchronization. This thread initiates the secure synchronization protocol, ESSP. If the protocol is performed successfully, the state of the client is set to the Aware state and the expiration time of the corresponding client is set to a specified time later than the present time as follows:

```
state(k) := 1
expirationTime(k) := NOW() + expirationTimeInterval
```

The expirationTimeInterval shown in the above pseudo-code is the system parameter, Expiration Time Interval and should be very carefully selected in order to keep the system working correctly.

Unsuccessful synchronization. The synchronization protocol may fail due to a general network error or misconfiguration in the client. The server performs the following pseudo-code when the synchronization protocol fails:

```
state(k) := state(k) + 2
```

The state of the client is set to the Lost state. This state is represented by an even integer greater or equal to 2 as shown above. The server attempts to reach a client in the Lost state for a number of times that is defined as the system parameter, Maximum Number of Seeding Attempts. For example, if this parameter is set to 10, the maximum value of a synchronization state of a client will be 20, which means the server could not send the synchronization parameters successfully to the client for 10 times in a row and has stopped attempting to reach it resulting in the state shift from Lost to Dead. This notion is useful because to divide the state value by 2 gives the number of unsuccessful synchronization attempts in a row between the server and the client.

If the system parameter, Seeding Interval is set to a large value like 12 hours, the unreachable clients will not be contacted until this amount of time elapses. So, another parameter is employed in the case of unsuccessful synchronization as follows:

```
nextSeedAttemptTime := NOW() + lostClientSeedInterval * state(k)
```

The system parameter, Lost Client Seeding Interval, is added to the present time after it is multiplied with the state of the client. This value is kept smaller than the seed interval, so that the clients in the Lost state can be synchronized earlier. The increasing value of a state indicates the high possibility of being a crucial problem in the client instead of a network problem and it is meaningless to attempt to reach it. Therefore, in order to increase the system performance, this parameter is multiplied with the state resulting in the increasing durations between the synchronization attempt times.

The thread is paused for a number of seconds that is defined as the system parameter, Seeding Interval following the delivering synchronization parameters to clients. The Seeding Interval is a significant parameter that can be used in performance optimization.

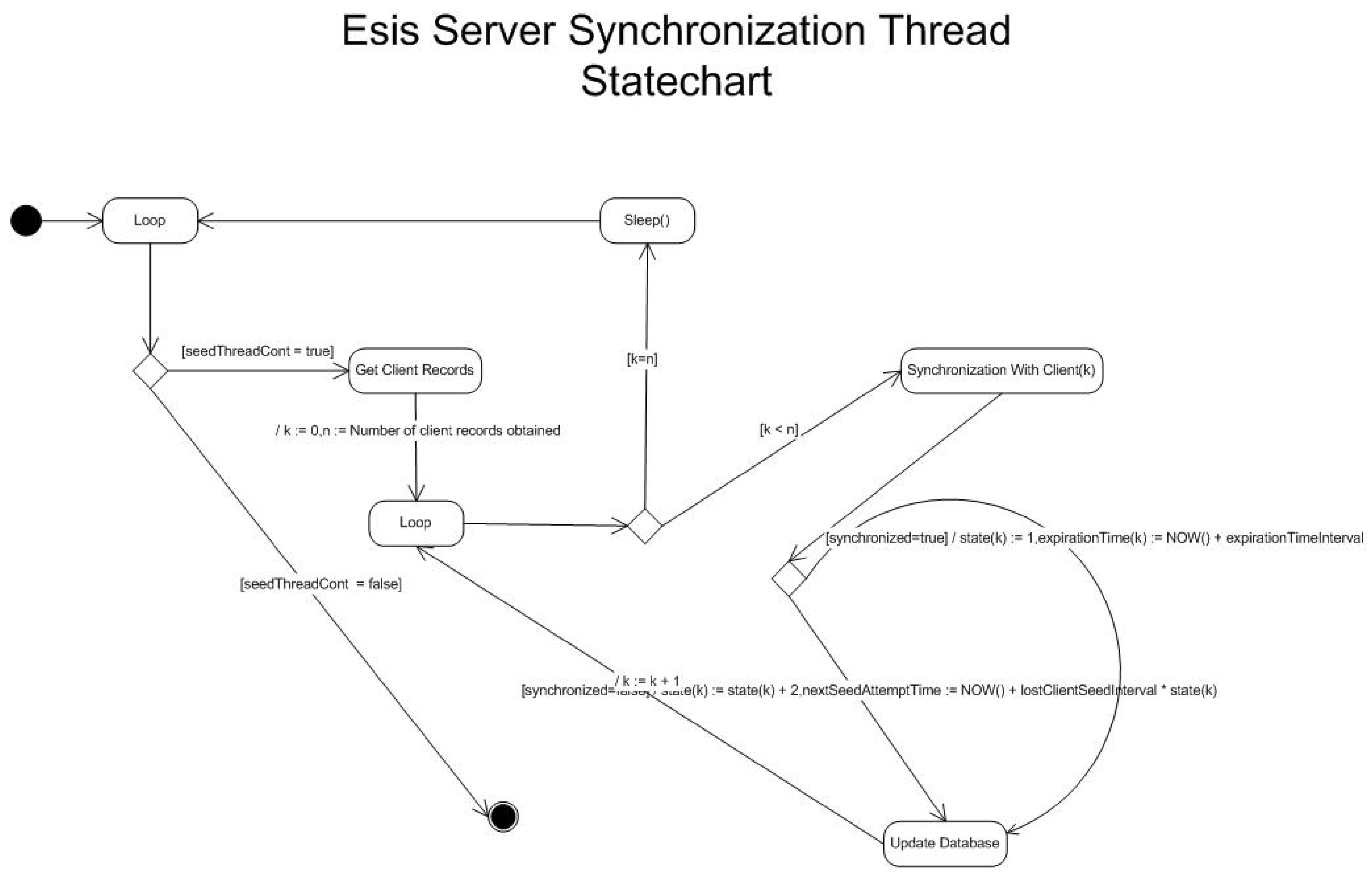

Esis Server Synchronization Thread Statechart
Loop
Sleep()
[seedThreadCont = true]
Get Client Records
[k=n]
Synchronization With Client(k)
/ k := 0,n := Number of client records obtained
[k < n]
Loop
[synchronized=true] / state(k) := 1,expirationTime(k) := NOW() + expirationTimeInterval
[seedThreadCont = false]
/ k := k + 1
[synchronized=false] / state(k) := state(k) + 2,nextSeedAttemptTime := NOW() + lostClientSeedInterval * state(k)
Update Database

Figure 4.2. ESIS Synchronization Thread Statechart

4.1.1.2 Password Update Thread

This thread handles the Password Synchronization Phase that is detailed in 3.4.4 at the server side.

ESIS Server periodically looks up in the database to find out the clients that are in the Aware state and have an expiration time value greater than or equal to the present time. This thread simply updates the local administrator password fields of those clients' database records with the newly generated ones. A new local administrator password is generated by a password generator routine that uses the current seed and the current local administrator password of the client. The same password generator routine is used by the service application at the client side which is as follows:

$$PWD_{new} := PWD_{current} \oplus SEED_{current}$$

where $\oplus$ is bitwise XOR operator.

While updating the local administrator password of a client, its state is also set to the Awaiting state if all of the rounds in the synchronization phase are finished. Therefore, it will be sent the new synchronization parameters for the next synchronization phase.

When the update operation is completed, the thread pauses for a number of seconds that is defined as the system parameter, Password Update Interval.

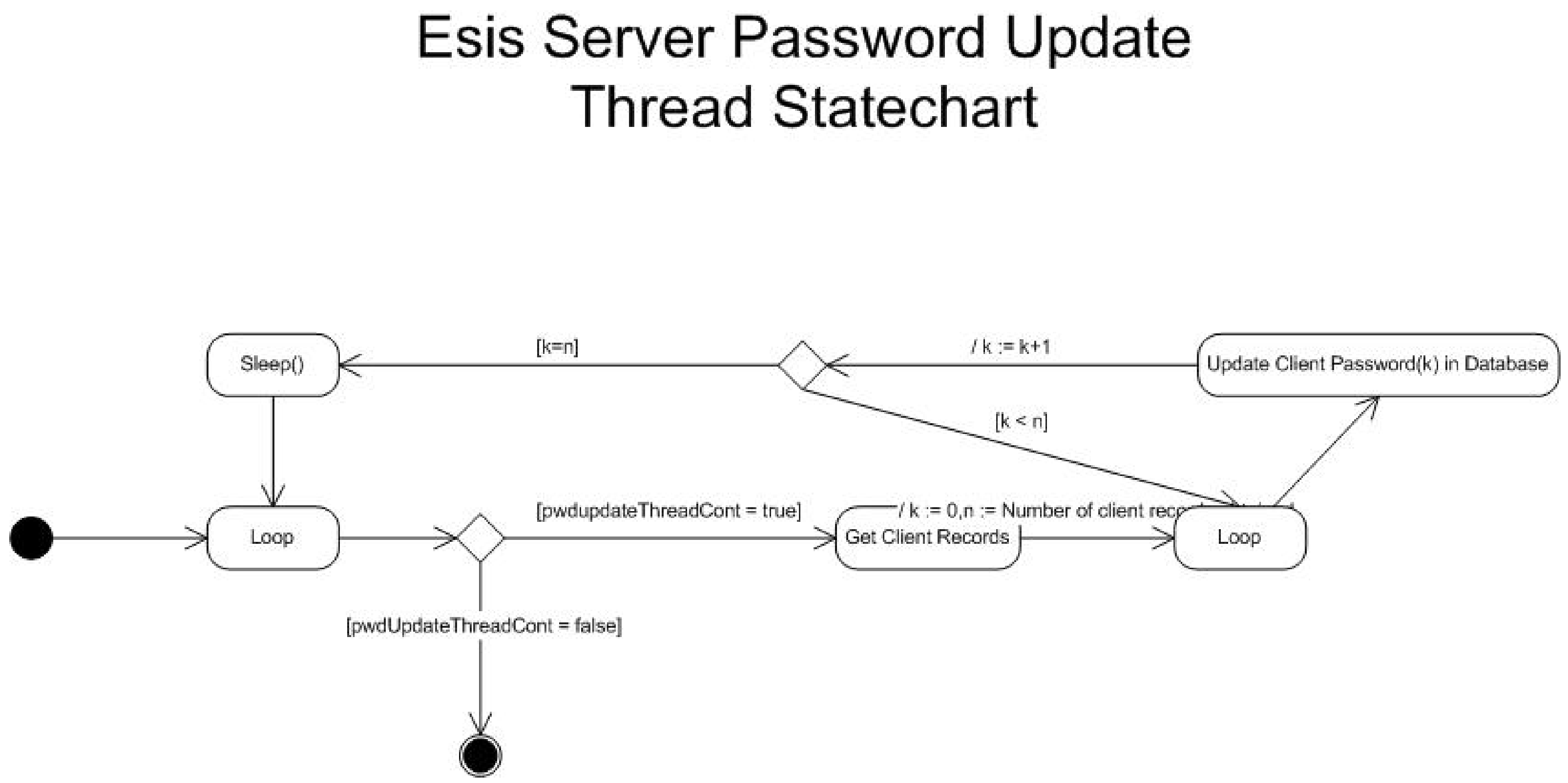

Esis Server Password Update Thread Statechart
Sleep()
[k=n]
/ k := k+1
Update Client Password(k) in Database
[k < n]
Loop
[pwdupdateThreadCont = true]
/ k := 0,n := Number of client reco
Get Client Records
Loop
[pwdUpdateThreadCont = false]

Figure 4.3. ESIS Server Password Update Thread Statechart

4.1.1.3 Client Initialization Thread

The server application listens on a specific port for client initialization requests in the Client Initialization Thread and it handles all of them. When the client service application is started, it initiates the client initialization protocol, ECIP in order to initialize itself and join the ESIS System.

The statechart of this thread is depicted in Figure 4.4. As shown in the figure, the result of the client initialization protocol is handled the same way as it is described in Section 4.1.1.1 where the Synchronization Thread is detailed.

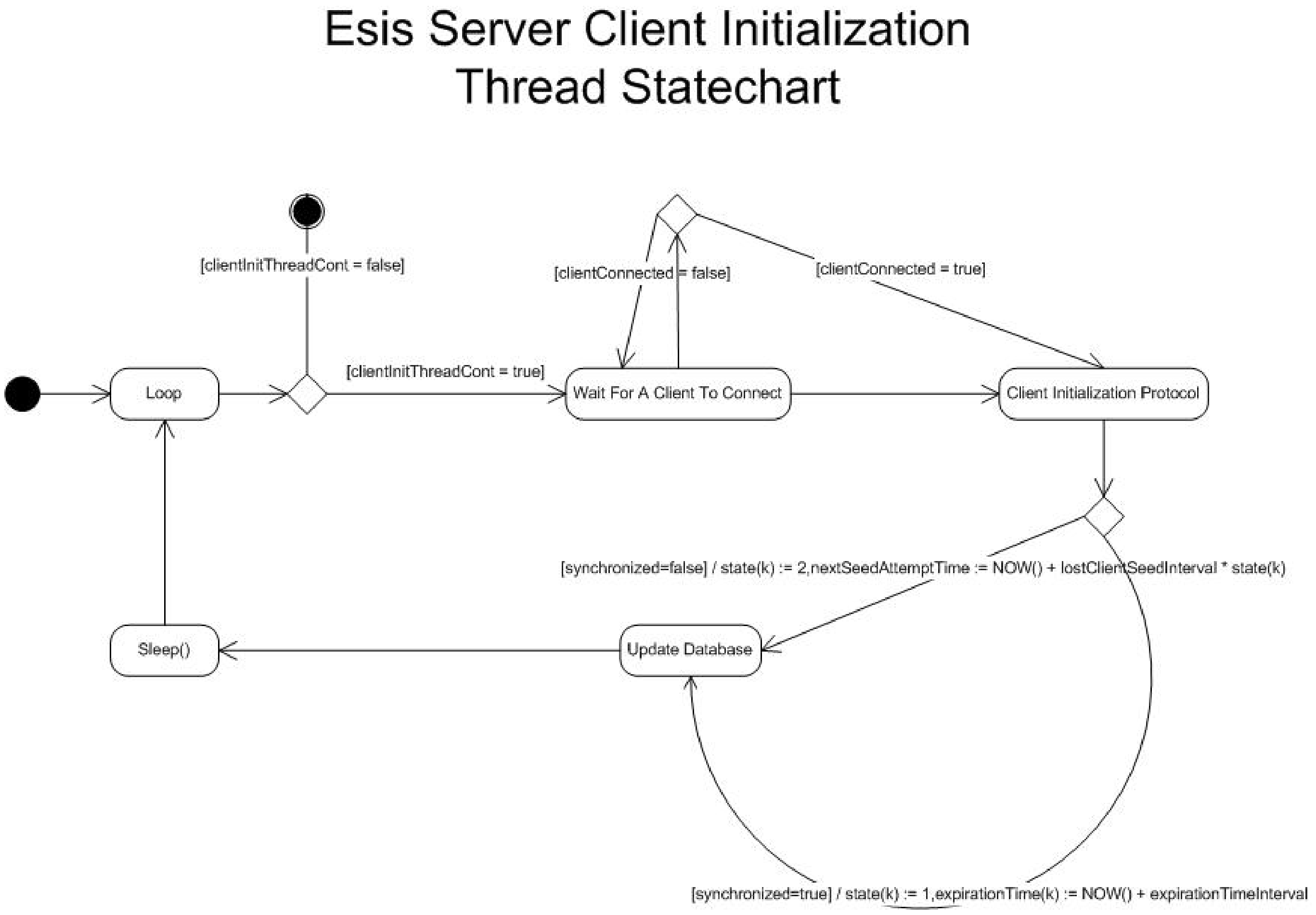

Esis Server Client Initialization Thread Statechart
[clientInitThreadCont = false]
[clientInitThreadCont = true]
Loop
[clientConnected = false]
[clientConnected = true]
Wait For A Client To Connect
Client Initialization Protocol
[synchronized=false] / state(k) := 2,nextSeedAttemptTime := NOW() + lostClientSeedInterval * state(k)
Sleep()
Update Database
[synchronized=true] / state(k) := 1,expirationTime(k) := NOW() + expirationTimeInterval

Figure 4.4. ESIS Server Client Initialization Thread Statechart

4.1.1.4 Password Query Thread

Technicians can query the server by using their tiny Win32[5] applications located on their tokens. The communication between the server and the token application is handled by the password query thread, which means that the ESQP is handled by this thread as shown in Figure 4.5.

When a token application connects to the server, it initiates the secure querying protocol (ESQP). The technician sends his password and username as well as the digital signature of his password located in the token. As you will remember, when the following conditions are met, the technician gets authenticated and can query the server.

1. $E_{pub}(E_{priv}(h(pwd_{token})) = h(pwd_{technician})$
2. $h(pwd_{technician}) = GetPassword(un, tid)$

The first condition guarantees that the technician is the real owner of the token because he cannot locate $E_{priv}(h(pwd_{token})$ in the token without the knowledge of the server private key. Server decrypts the message using its private RSA key and compares it with the digest of technician's password. If they are the same, it can be trusted that the token belongs to the technician.

The second condition guarantees that technician's password is correct or he is still employed. GetPassword() is a function which returns the password associated with the username and token id which are stored in the database on the server. If the technician quits his job, to change or to delete the username and password entry associated with him from the database is adequate to restrict his access to the system.

[5] Win32 applications are developed using the Win32 Application Programming Interface (Win32 API) provided by the Windows Operation System resulting in smaller and more portable applications.

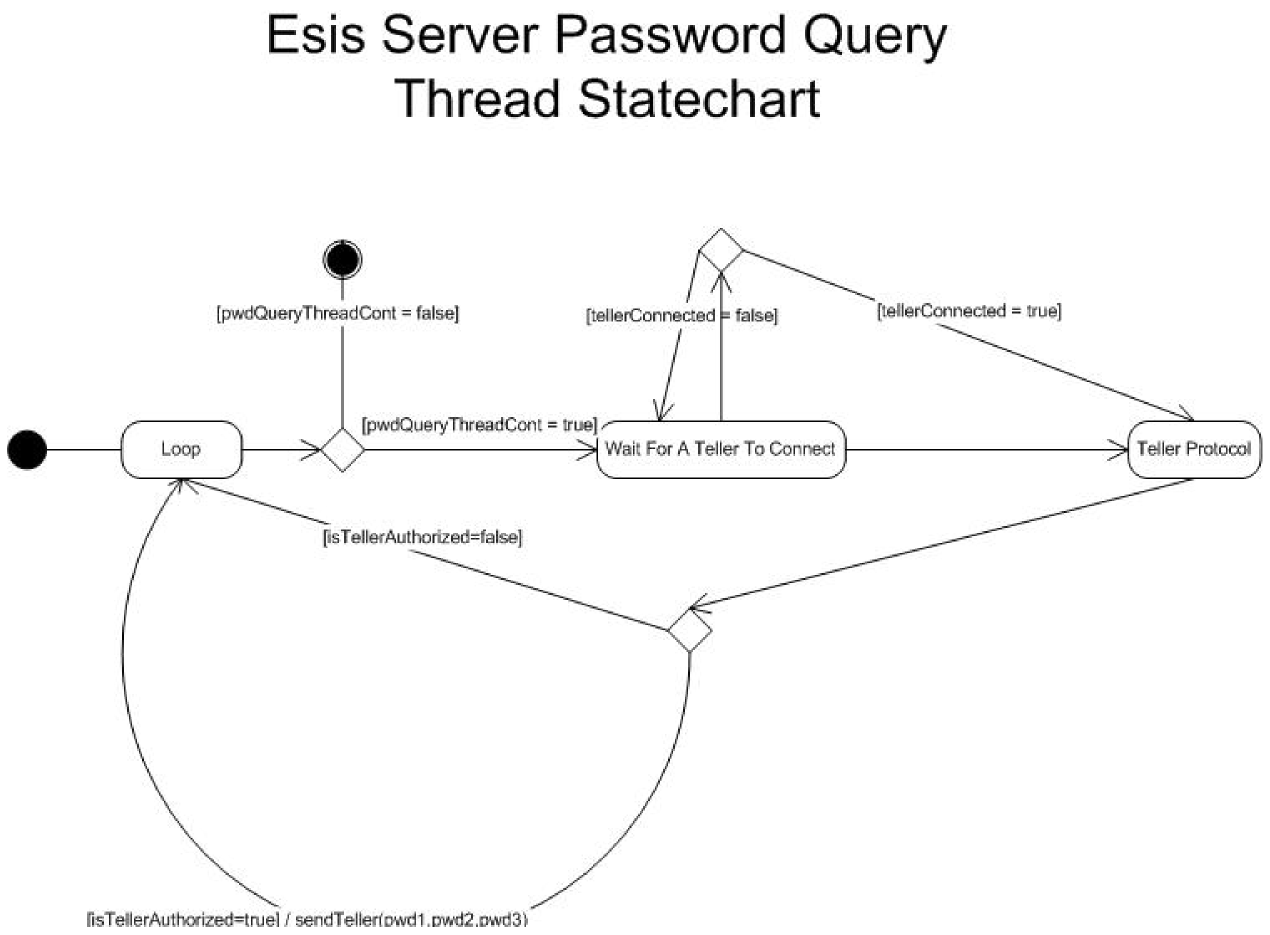

Esis Server Password Query Thread Statechart
[pwdQueryThreadCont = false]
[tellerConnected = false]
[tellerConnected = true]
[pwdQueryThreadCont = true]
Loop
Wait For A Teller To Connect
Teller Protocol
[isTellerAuthorized=false]
[isTellerAuthorized=true] / sendTeller(pwd1,pwd2,pwd3)

Figure 4.5. ESIS Password Query Thread Statechart

4.1.2 ESIS Client

ESIS Client has been developed in the C programming language. It is compatible with many of the Windows platforms[6]. ESIS Client is a Windows service application. In Windows operating systems, a service is an application that continuously runs in the background. The Unix name for a Windows service equivalent is daemon. The purpose of developing the client application as a Windows service is to eliminate the need to user-interaction to start the program. As the client computer starts, the service application starts automatically. In Figure 4.6, the statechart of the ESIS Client service application is depicted.

In the initialization step, the client initiates the client initialization protocol, ECIP. The client constantly initiates the protocol until it can authenticate itself to the server and establish synchronization by performing the rest of the protocol. If the protocol is completed successfully, the client gets the synchronization parameters in order to change its local administrator passwords at the given time.

The client waits for the activation time to change its current local administrator password. Since the number of rounds in a synchronization phase is one in our implementation, after the activation (replacing password with the newly generated one), the client starts listening on a specified socket and waits for the server to start the synchronization protocol, ESSP. The server connects to the client and starts the ESSP. If synchronization is successful, the client again waits for the activation time to change its current local administrator password, otherwise it keeps on listening on the specified port for the next ESSP session.

[6] Windows NT 4, Windows 2000, Windows XP

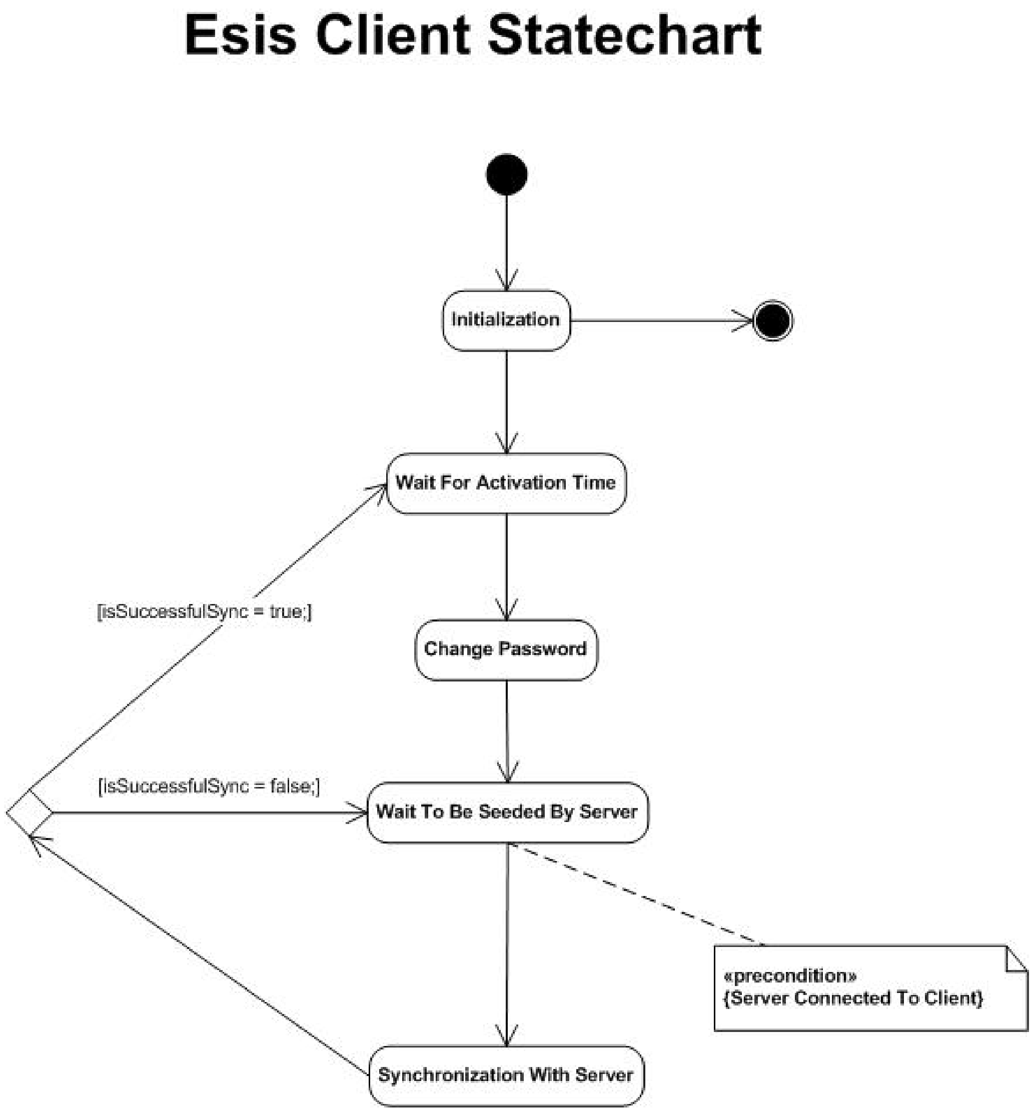

Esis Client Statechart
Initialization
Wait For Activation Time
Change Password
Wait To Be Seeded By Server
Synchronization With Server
[isSuccessfulSync = true;]
[isSuccessfulSync = false;]
«precondition»
{Server Connected To Client}

Figure 4.6. ESIS Client Statechart

5 PERFORMANCE EVALUATION

Various parameters affect the performance of the system. Some of these parameters are configurable; some of them are not. In Section 3.3.1, the list of configurable system parameters are presented, which are also listed below. We have used the parameters in Table 5.1 in our simulations in order to evaluate the performance of the system.

Simulation Parameters		
Parameter		**Descripton:**
n	Number of clients	This is the number of clients in the ESIS System.
$\mathbf{t_{exp}}$	Expiration Interval	The server application uses the expiration interval in order to calculate the next activation time of the client by adding it to the present time before it sends the synchronization parameters to the client.
$\mathbf{t_s}$	Seeding Interval	The duration between the synchronization phases initiated by the server is specified by this system parameter.
$\mathbf{t_{lcs}}$	Lost Client Seeding Interval	As a result of an unsuccessful synchronization phase, the corresponding client is contacted with the server after an amount of time specified by this parameter.
m	Maximum Number of Seeding Attempts	The server stops attempting to initiate the ESSP with a client after that number of attempts.
$\mathbf{P_s}$	Synchronization Success Probability	The probability of success of a single run of the synchronization protocol is defined by this parameter.

Table 5.1. Simulation Parameters

We have examined the secure querying protocol, ESSP in the simulations. The ESSP is the main and the most important component of the system. Almost all of the synchronization is carried out by the ESSP. Other protocols are used rarely in the system. Therefore, focusing on the ESSP in order to evaluate the performance is a valid approach.

5.1 Assumptions and Constant Parameters

In the implementation of the system, the Password Synchronization Phase consists of one round as mentioned in 4.1.2, so does in the simulation. The network bandwidth is constant at 100 Kbit/sec. The total size of the packets used in the ESSP as shown in Section 3.5.3 is 1324 bits. The total processing time of the protocol is 0.1 seconds. This value has been computed with a server and a client that have the same configuration of Pentium Celeron 1333 MHz processor and 512 MB Ram. Therefore, the total amount of time passing in an ESSP session between a client and a server, t_{essp} is as follows:

$$t_{essp} = 1324 / (100 * 1024) + 0.1 \approx 0.13 \text{ sec}$$

All of the simulations are performed for 20 days, which we believe is sufficient to show the system behavior correctly. The initial synchronization time of the clients are randomly distributed within the first day.

5.2 Tuning Expiration Interval

The server application as explained in Section 4.1.1 uses the expiration interval in order to calculate the next activation time of the client before it sends the synchronization parameters. This value is added to the present time and sent to the client. Setting this parameter to a large value results in the small number of synchronizations.

Eight cases are examined, in which the expiration interval is configured as 1, 2, 3, 6, 12, and 24 hours, respectively. Same values have been used for the other parameters as shown in Table 5.2.

Parameter	Value
n	3,000
t_s	1 hour
t_{lcs}	15 min
m	5
P_s	0.9

Table 5.2. Simulation parameters used in tuning expiration interval

The next time a client should be synchronized gets earlier as the expiration interval is shortened, so that the total number of synchronization attempts made in the simulation time of 20 days increases naturally. Decreasing the expiration interval to a lower value results in increasing number of synchronization attempts as shown in Table 5.3 and Figure 5.1.

t_{exp}	**Number of Synchronization Attempts**		
	Successful	**Failed**	**Total**
1 hours	1271466	133987	1405453
2 hours	656849	72640	729489
3 hours	453563	47437	501000
6 hours	231,538	23,892	255,430
12 hours	117,587	12,291	129,878
24 hours	59,732	6,596	66,328

Table 5.3. Number of synchronization attempts due to t_{exp}

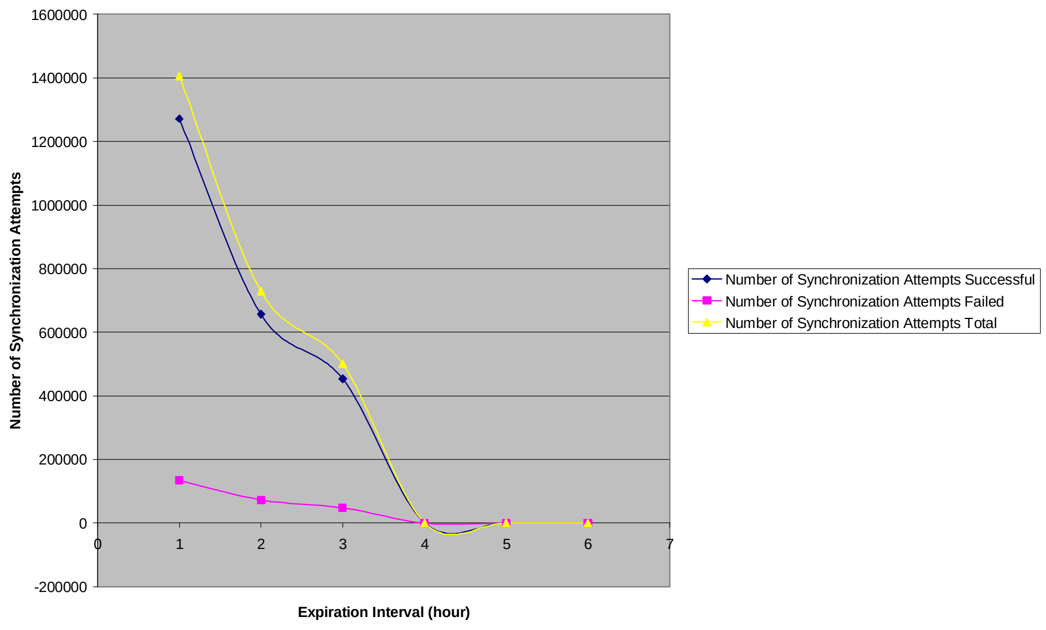


Figure 5.1. Number of synchronization attempts due to t_{exp}

The high number of the clients in the Awaiting state indicates high number of synchronization attempts. In Figure 5.2, Figure 5.4, Figure 5.6, Figure 5.8, Figure 5.10, and Figure 5.12, the number of Awaiting clients are shown due the varying t_{exp}. In the figures, the hours in the twenty-day simulation are located in the X axis whereas the number of awaiting clients are located in the Y axis.

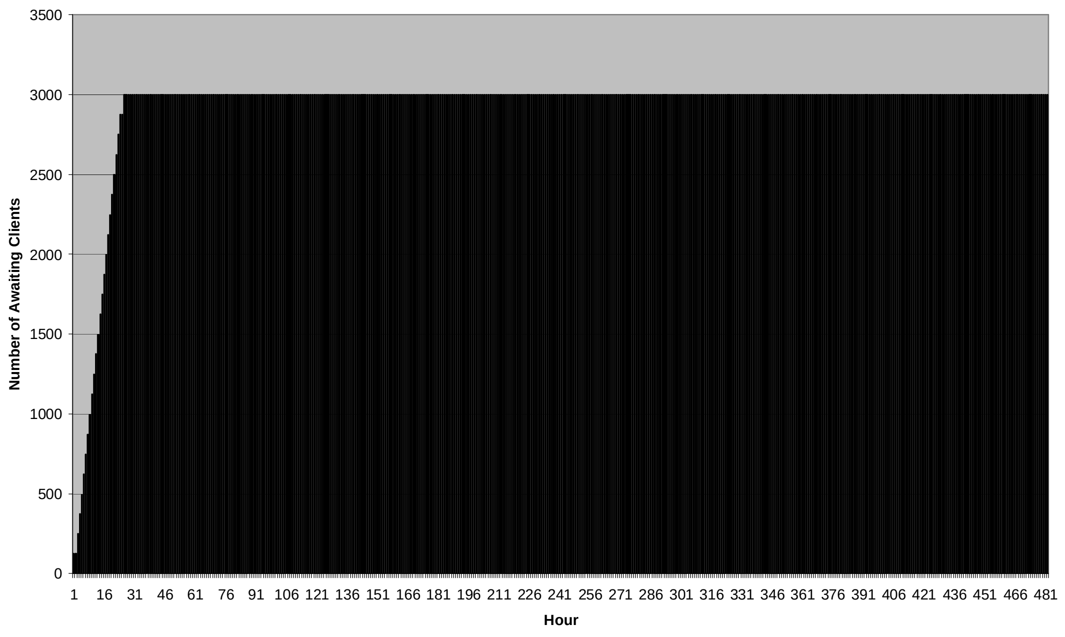


Figure 5.2. Tuning expiration interval to 1 hour (20-day period is plotted)

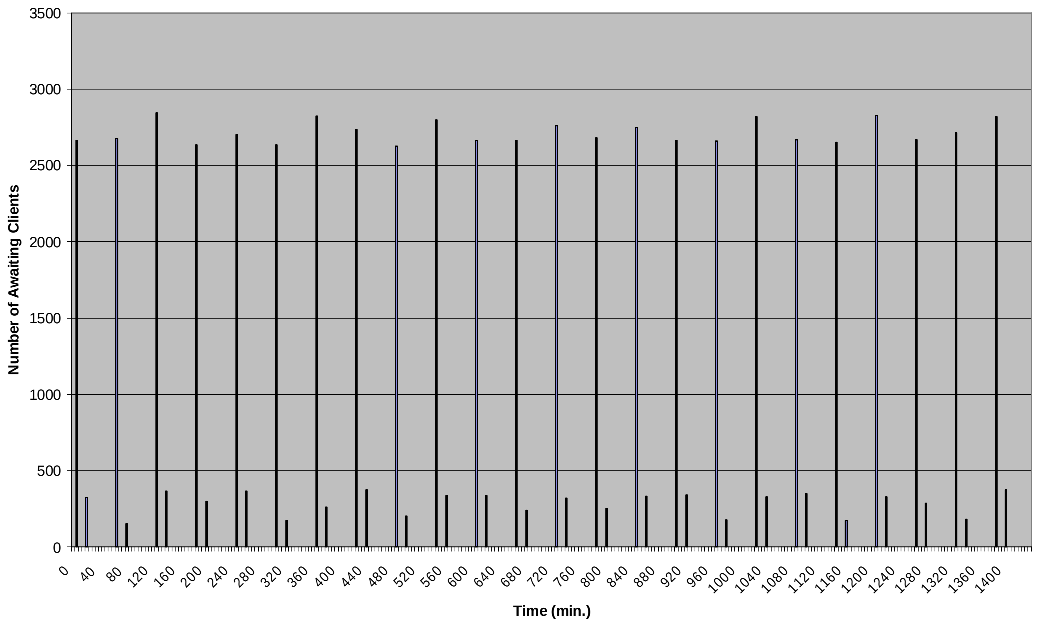


Figure 5.3. Tuning expiration interval to 1 hour (1-day period is plotted)

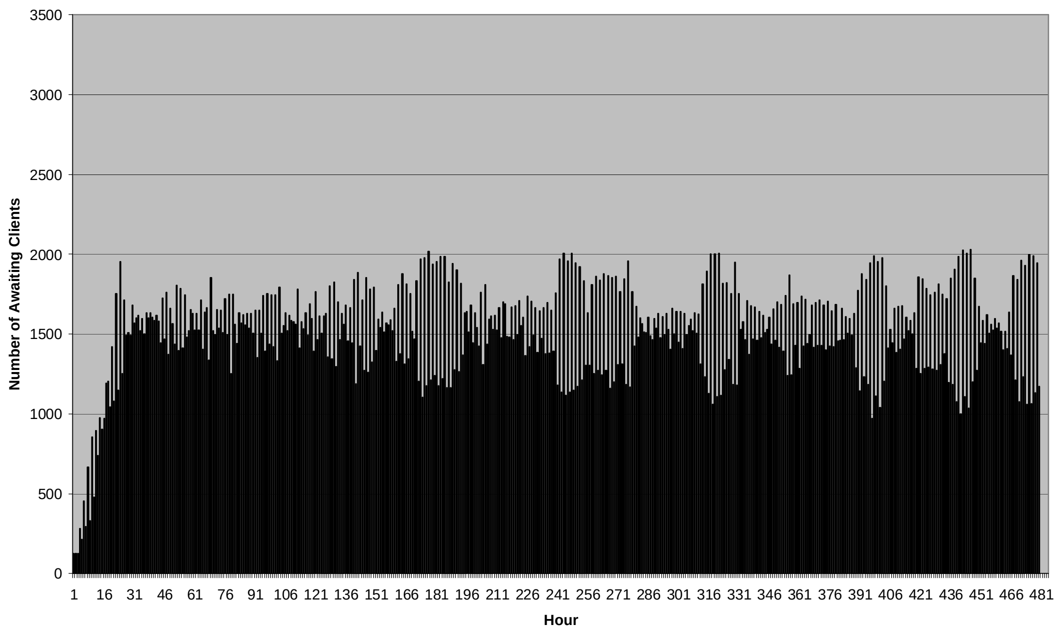


Figure 5.4. Tuning expiration interval to 2 hours (20-day period is plotted)

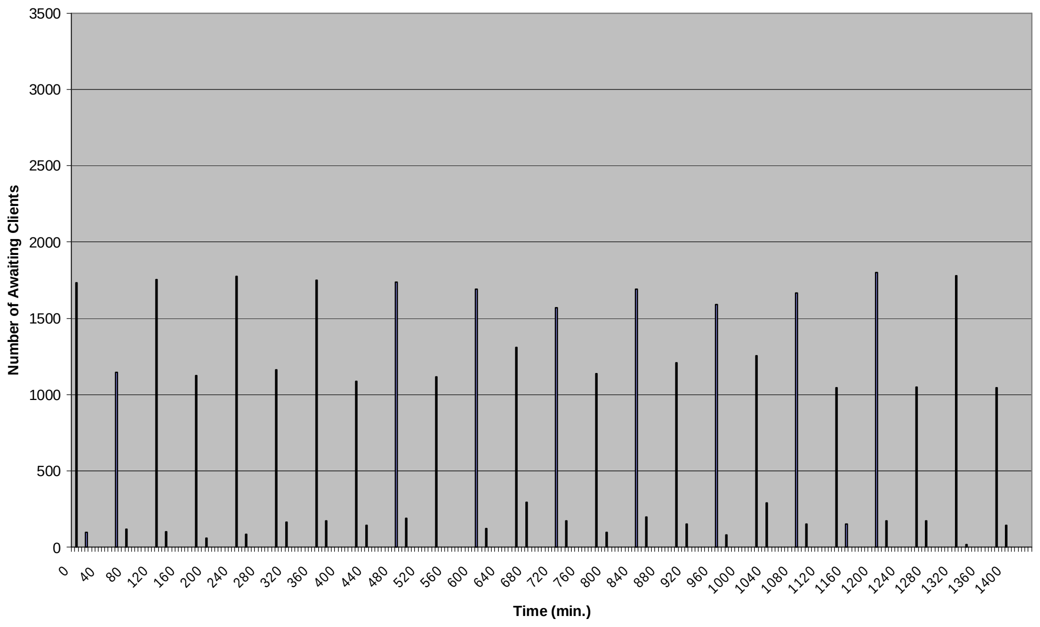


Figure 5.5. Tuning expiration interval to 2 hours (1-day period is plotted)

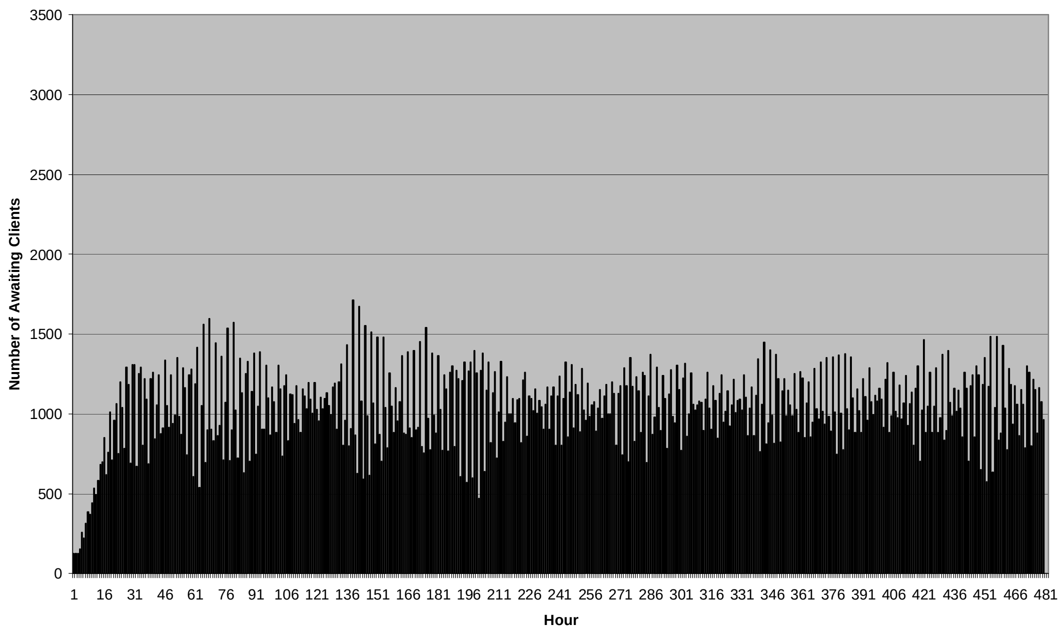


Figure 5.6. Tuning expiration interval to 3 hours (20-day period is plotted)

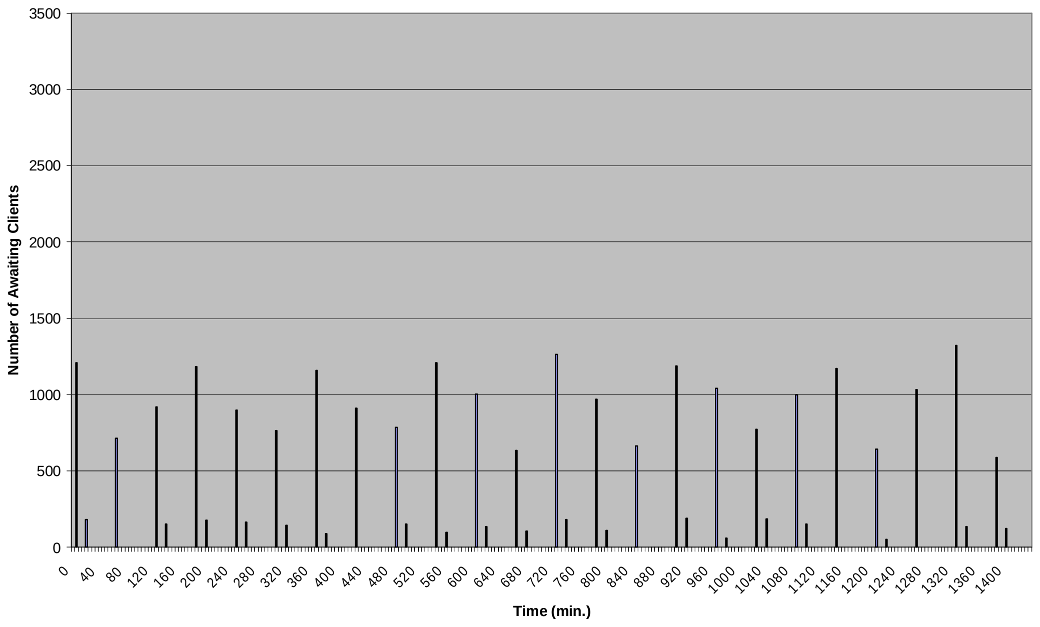


Figure 5.7. Tuning expiration interval to 3 hours (1-day period is plotted)

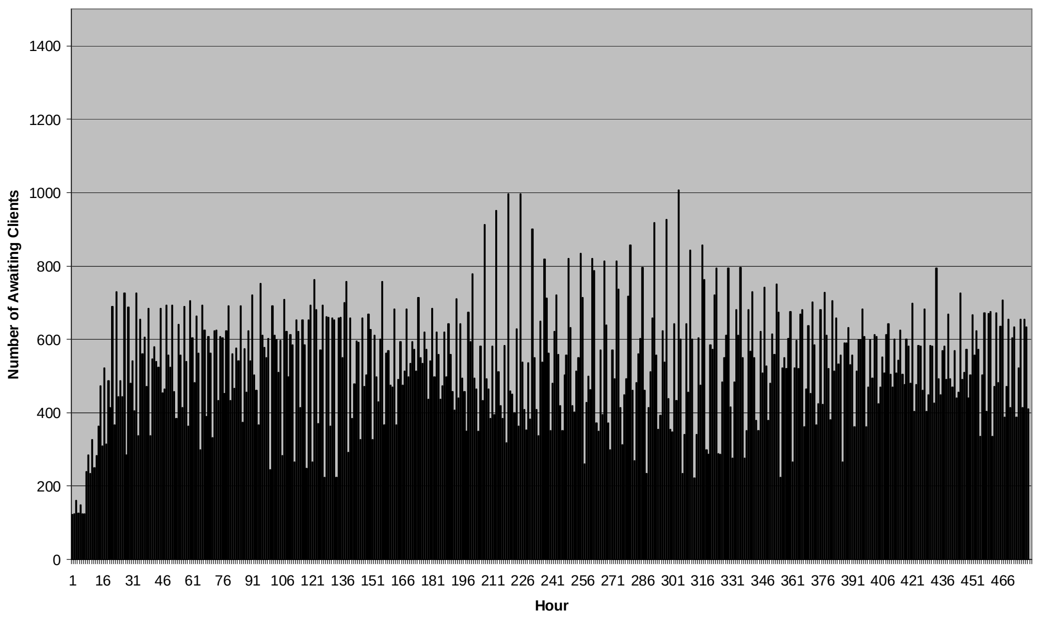


Figure 5.8. Tuning expiration interval to 6 hours (20-day period is plotted)

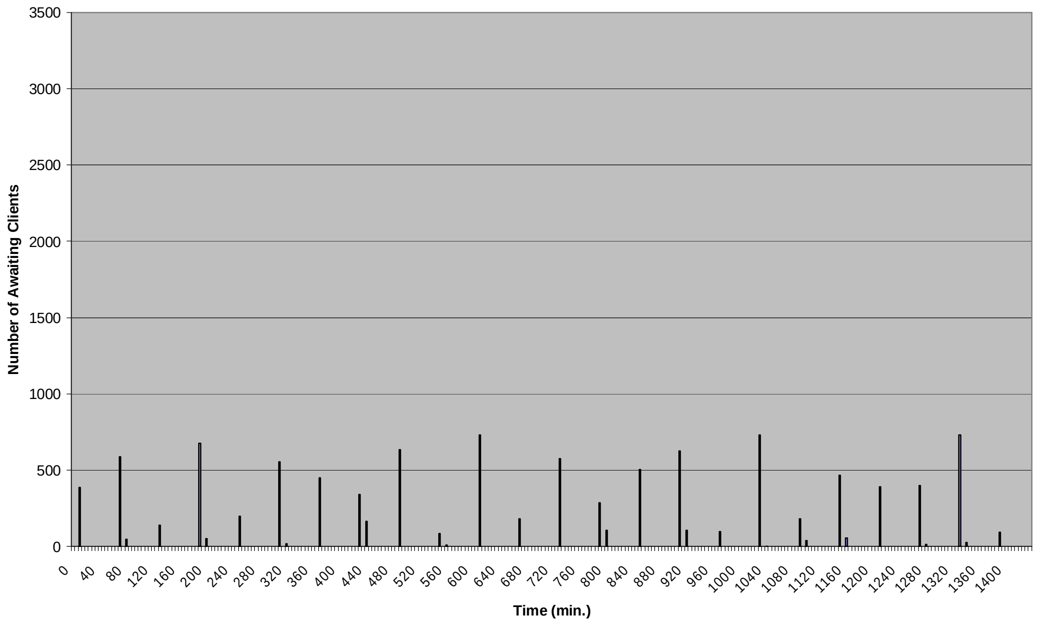


Figure 5.9. Tuning expiration interval to 6 hours (1-day period is plotted)

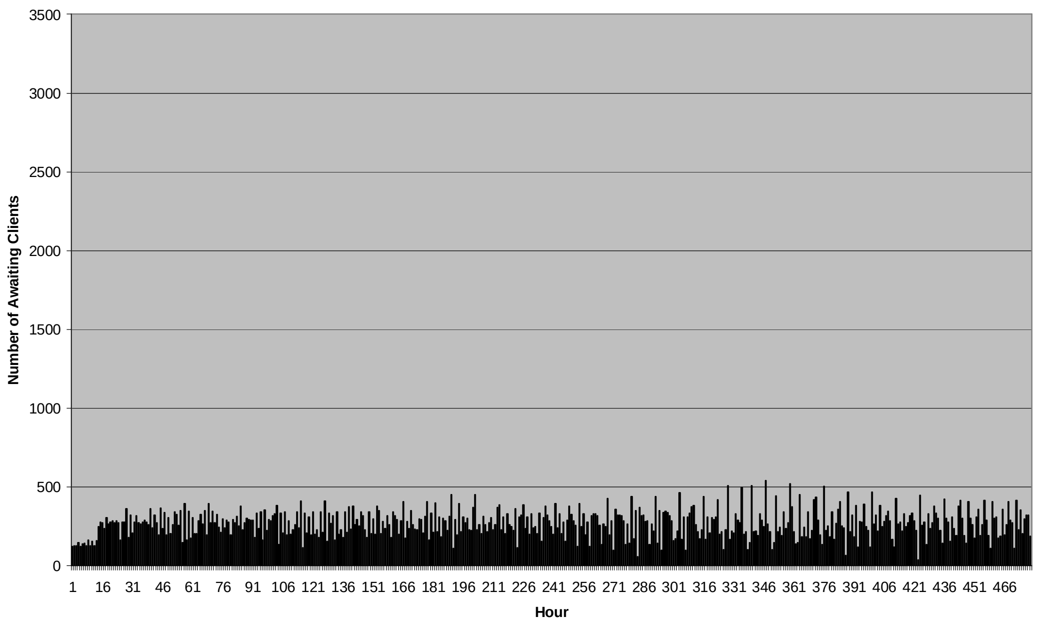


Figure 5.10. Tuning expiration interval to 12 hours (20-day period is plotted)

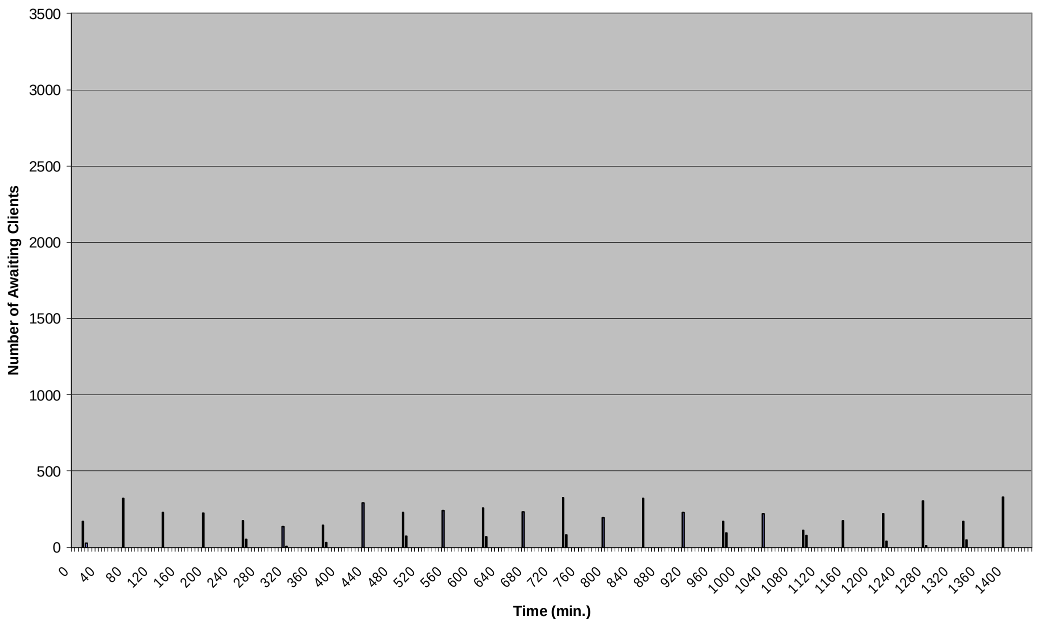


Figure 5.11. Tuning expiration interval to 12 hours (1-day period is plotted)

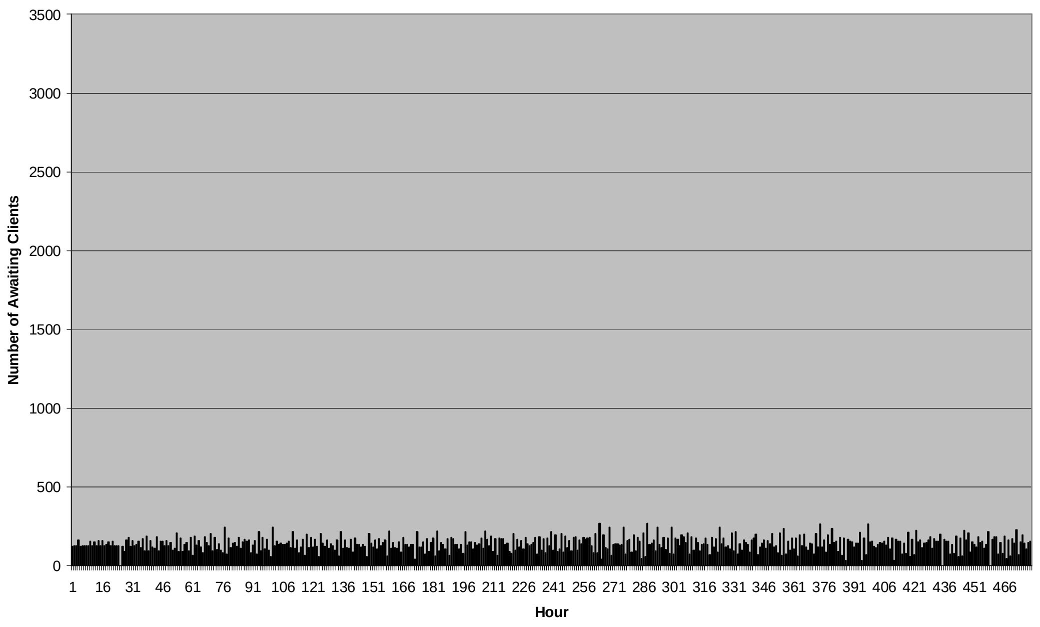


Figure 5.12. Tuning expiration interval to 24 hours (20-day period is plotted)

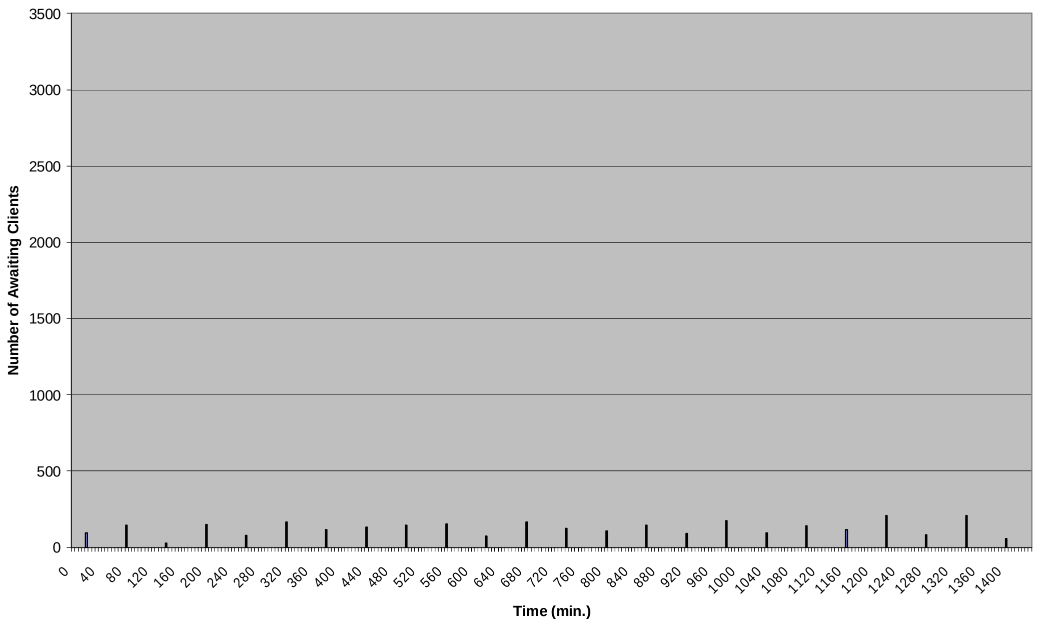


Figure 5.13. Tuning expiration interval to 24 hours (1-day period is plotted)

After the state of a particular client becomes Awaiting, this client becomes ready to be synchronized. Moreover, to increase the expiration interval causes a client that is synchronized to be in the Aware state for a long time. Therefore, setting the expiration interval to 24 hours decreases the number of clients that are in the Awaiting state per hour as shown in Figure 5.12 and Figure 5.13. If a synchronization attempts fails, it is repeated but earlier because the lost client seeding interval is smaller than the seeding interval. Therefore, the number of clients in the awaiting state decreases in some hours as shown in Figure 5.4, Figure 5.6, Figure 5.8, Figure 5.10, and Figure 5.12, meaning that these clients will be contacted again because of the previous unsuccessful synchronization attempt .

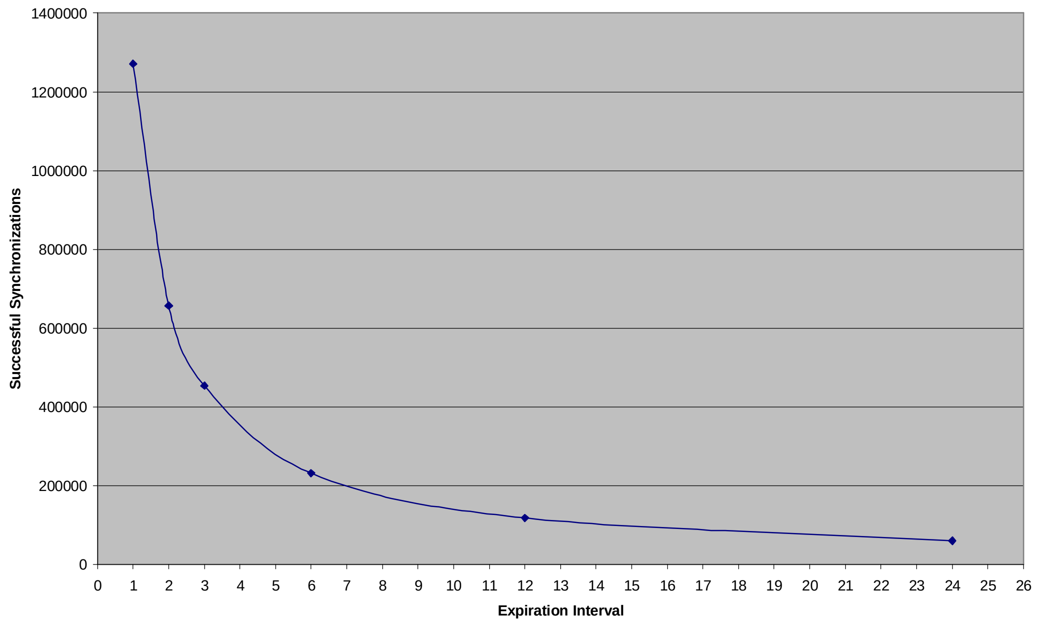


Figure 5.14. The average of successful synchronizations per client

In an enterprise network, there may be more than 3,000 computers. We have simulated our system for networks consisting of 3,000, 10,000, and 20,000 computers and obtained the results in Table 5.4.

t_{exp}	**Avg. Successful Synchronizations**		
	n=3,000	n=10,000	n=20,000
1 h	423.8	423.2	423.6
6 h	77.13	77.06	77.14
12 h	39.14	39.14	39.12
24 h	19.91	19.90	19.90

Table 5.4. The Average of successful synchronizations per client in different-sized Networks

Decreasing the length of the expiration interval increases the number of synchronization attempts made by the server as expected. The average of the number of successful synchronizations per client stays constant in different-sized networks as shown in Table 5.4. Scalability of the system can be defined as the direct proportion between the number of clients and the average of successful synchronizations per client. Therefore, the system can be scaled up to a higher number of clients if the parameters are configured correctly.

5.3 Tuning Seeding Interval

The duration between the synchronization phases initiated by the server is specified by a system parameter, Seeding Interval. In the simulations, we have examined this parameter by setting it to 1, 6 and 12 hours while the other parameters have been set as shown in Table 5.5.

Parameter	**Value**
n	1,000
$\mathbf{t_{exp}}$	12 hours
$\mathbf{t_{lcs}}$	15 min
m	5
$\mathbf{P_s}$	0.9

Table 5.5. Simulation parameters used in tuning seeding interval

The simulation has yielded the results in Table 5.6. The number of successful synchronizations between the clients and the server decreases as the length of the seeding interval increases.

$\mathbf{t_s}$	**Number of Synchronization Attempts**		
	Successful	**Failed**	**Total**
1 hour	391,428	41,901	433,329
6 hours	374,885	40,325	415,210
12 hours	356,545	38,457	395,002

Table 5.6. Number of synchronization attempts due to t_s (seeding interval)

In Figure 5.15, Figure 5.16, and Figure 5.17, the number of awaiting clients per hour in the beginning of the timeline is nearly constant because the initial synchronization times of the clients are assigned by the simulation program. As shown in Figure 5.15, the synchronization attempts can be evenly distributed over the timeline by configuring the seeding interval to 1 hour instead of 6 or 12 hours. As shown in Figure 5.16 and Figure 5.17, to enlarge the interval causes the synchronization attempts to be split over a few number of points in the timeline, which may result in performance bottlenecks in the server. Therefore, distributing synchronization attempts smoothly in the time and avoiding performance bottlenecks can be achieved just by shortening the seeding intervals.

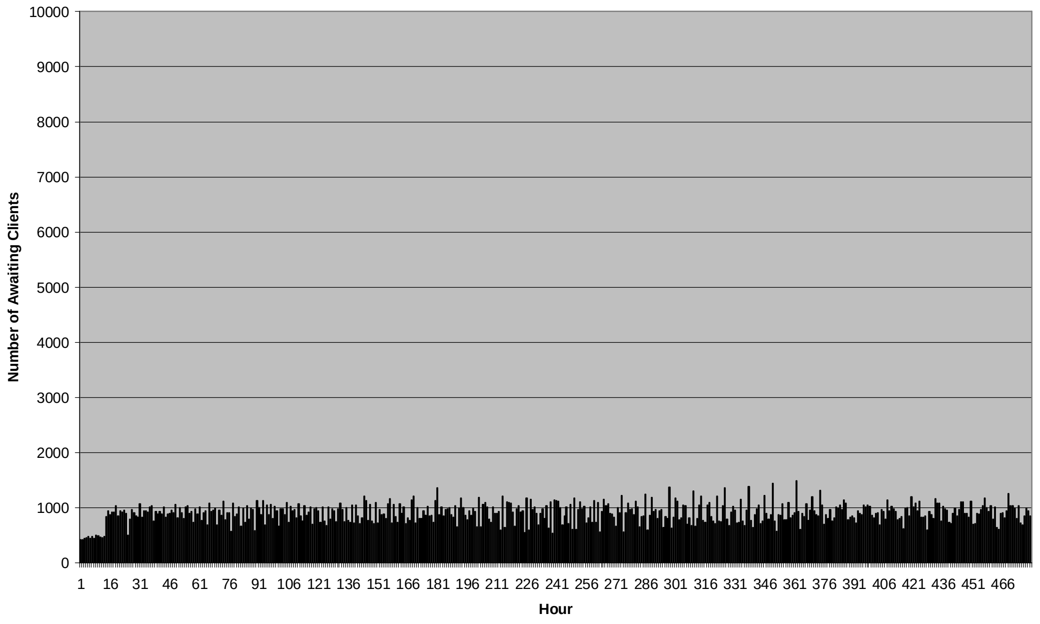


Figure 5.15. Tuning seeding interval to 1 hour

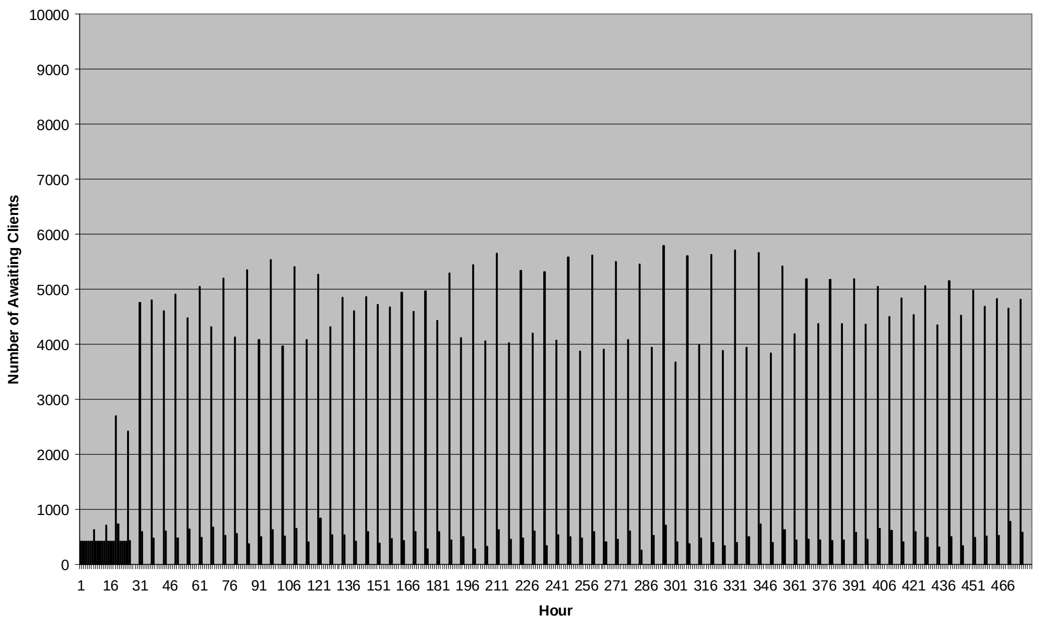


Figure 5.16. Tuning seeding interval to 6 hours

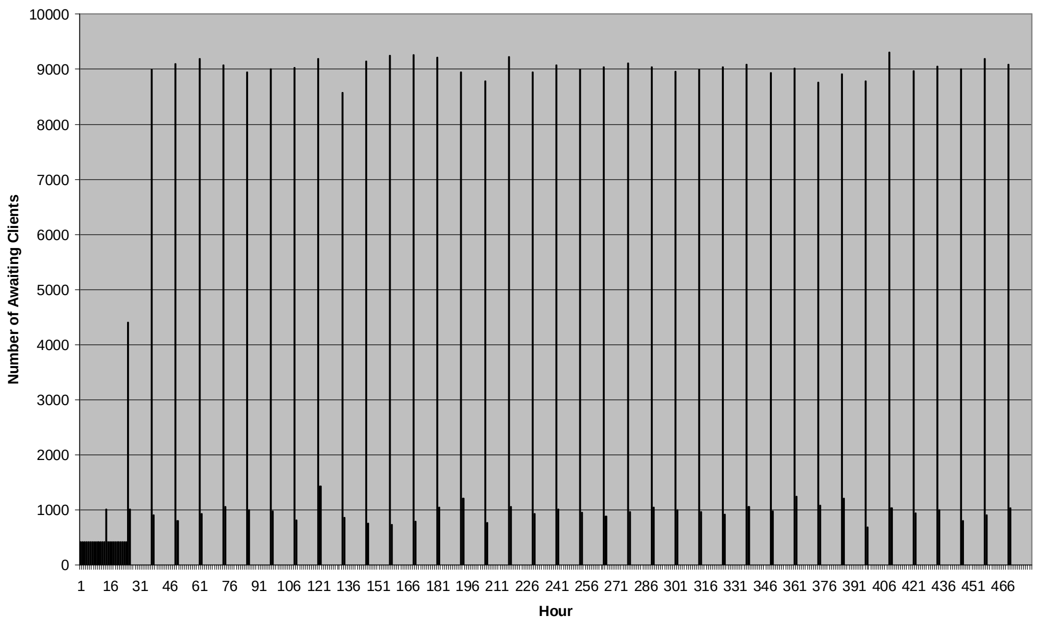


Figure 5.17. Tuning seeding interval to 12 hours

In the previous simulations mentioned above, the seeding interval is configured as 1, 6, and 12 hours that are relatively big values. In order to analyze the effect of smaller seeding intervals, we have run the simulations for 2 days (48 hours) and set the seeding interval to 10, 15, 30, and 60 seconds while keeping the expiration interval as 1 hour.

As shown in Table 5.7, the total numbers of synchronization attempts in each run are very close to each other. Therefore, the seeding interval can be configured as a very small value. In addition, the distribution of the synchronization attempts over time (one hour period) can be seen in Figure 5.18, Figure 5.19, Figure 5.20, and Figure 5.21.

t_s	**Number of Synchronization Attempts**		
	Successful	**Failed**	**Total**
10 seconds	36,399	3,661	40,060
15 seconds	36,467	3,931	40,398
30 seconds	36,186	4,307	40,493
60 seconds	35,825	3,239	39,064

Table 5.7. Number of synchronization attempts due to smaller t_s (seeding interval)

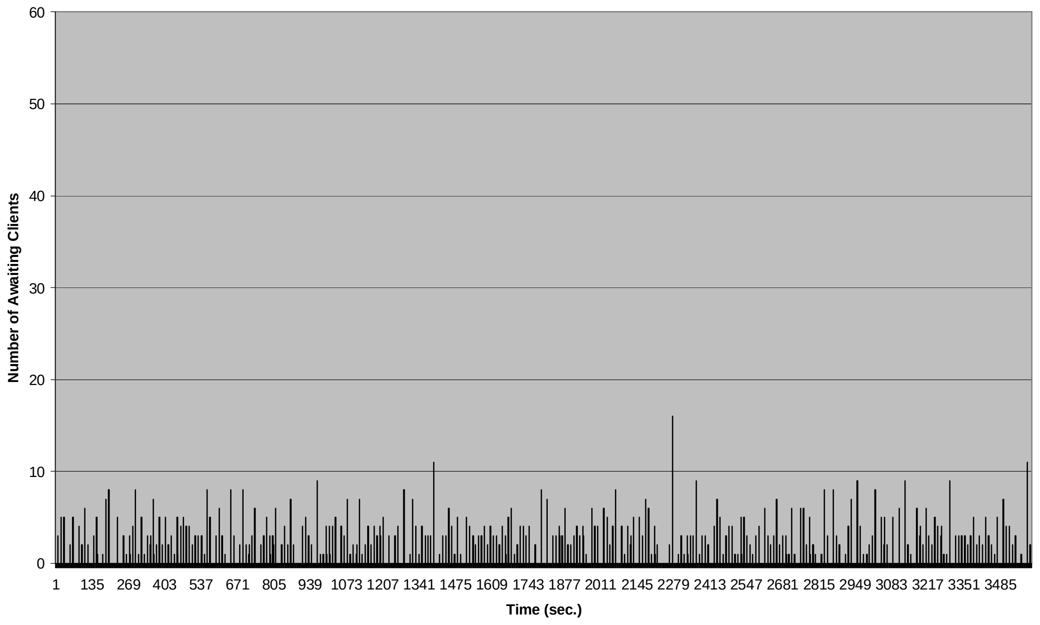


Figure 5.18. Tuning seeding interval to 10 seconds

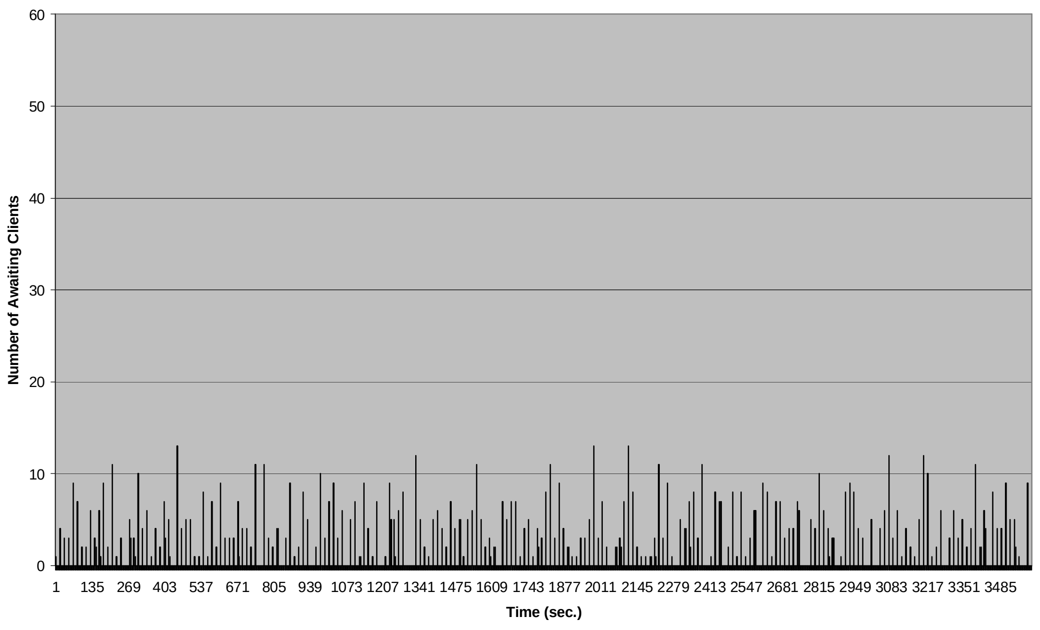


Figure 5.19. Tuning seeding interval to 15 seconds

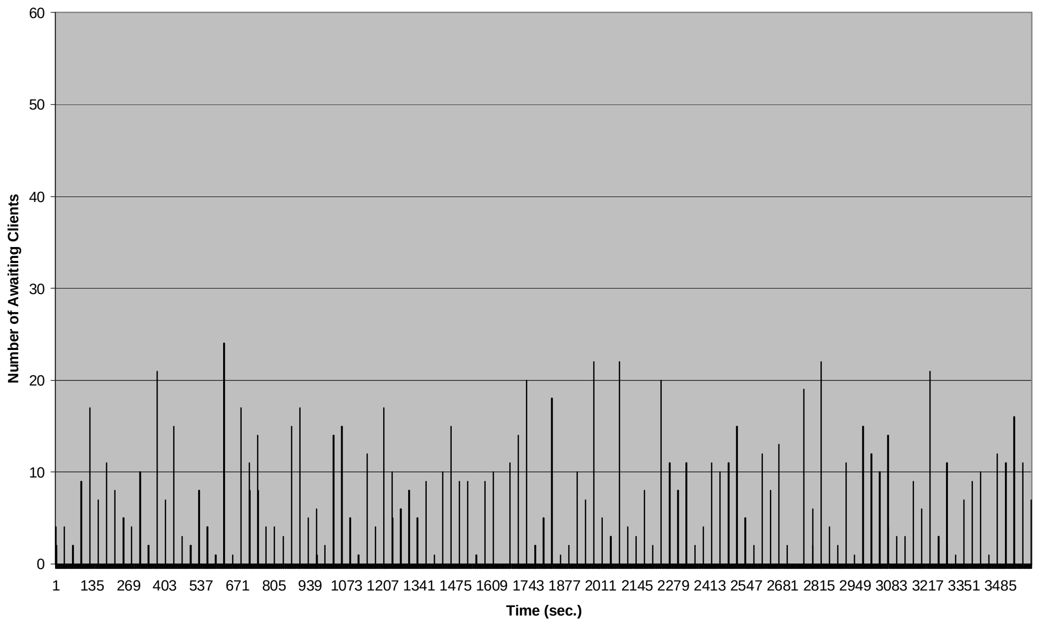


Figure 5.20. Tuning seeding interval to 30 seconds

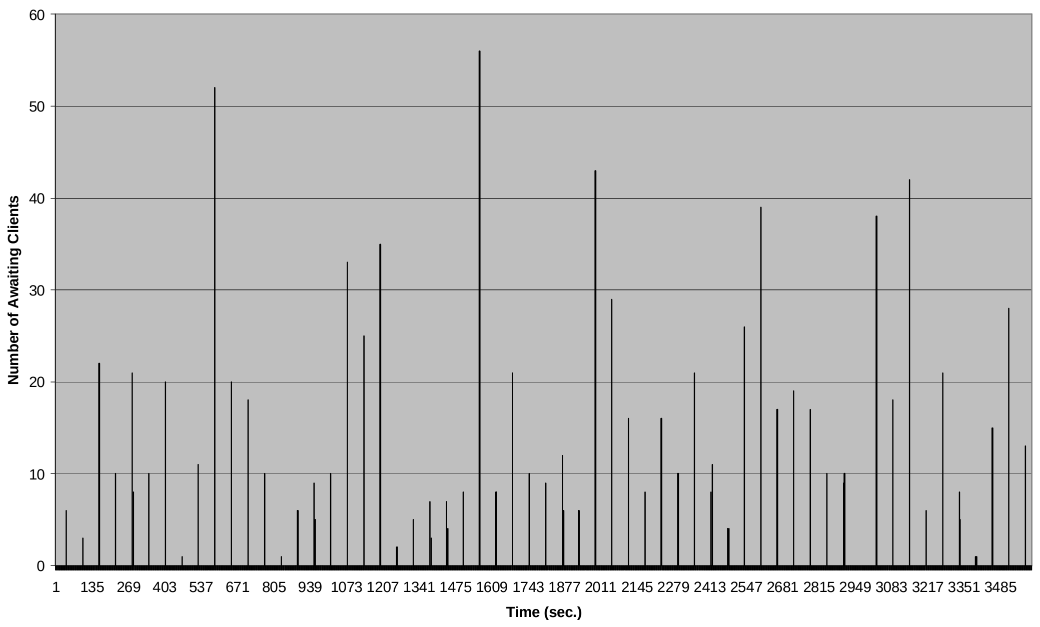


Figure 5.21. Tuning seeding interval to 60 seconds

The simulation results show that the seeding interval is not a very critical parameter in tuning the system performance because it does not affect the total number of synchronizations when it is changed. However, the system administrator can control the distribution of the synchronization attempts over time. This feature leads to a more controlled system.

5.4 Tuning Maximum Number of Seeding Attempts

The server stops attempting to initiate the ESSP with a client after a certain amount of unsuccessful attempts. This number is specified by a system parameter, Maximum Number of Seeding Attempts. Unsuccessful seeding may occur because of the general network errors. It is vital to configure this parameter correctly. This notion is integrated into the model by using a parameter that defines the probability of success of a single run of the synchronization protocol, P_S.

The probability of considering a client in the Dead state, P_{dead}, is calculated as follows:

$$P_{dead} = (1-P_s)^m$$

In the equation above, m is the maximum number of seeding attempts. Increasing the value of m increases the probability of success, but also causes the server to initiate the synchronization protocol much more times. Therefore, configuring this parameter to a higher value makes the system more robust but creates much more network traffic. On the other hand, decreasing its value results in dead clients that are in fact running smoothly. The simulation parameters as shown in Table 5.8 describes this relation very clearly as it is shown in Figure 5.22.

Parameter	**Value**
n	10,000
t_s	1 hour
t_{exp}	12 hours
m	1, 3, and 7
P_s	0.5

Table 5.8. Simulation parameters used in tuning maximum number of seeding intervals

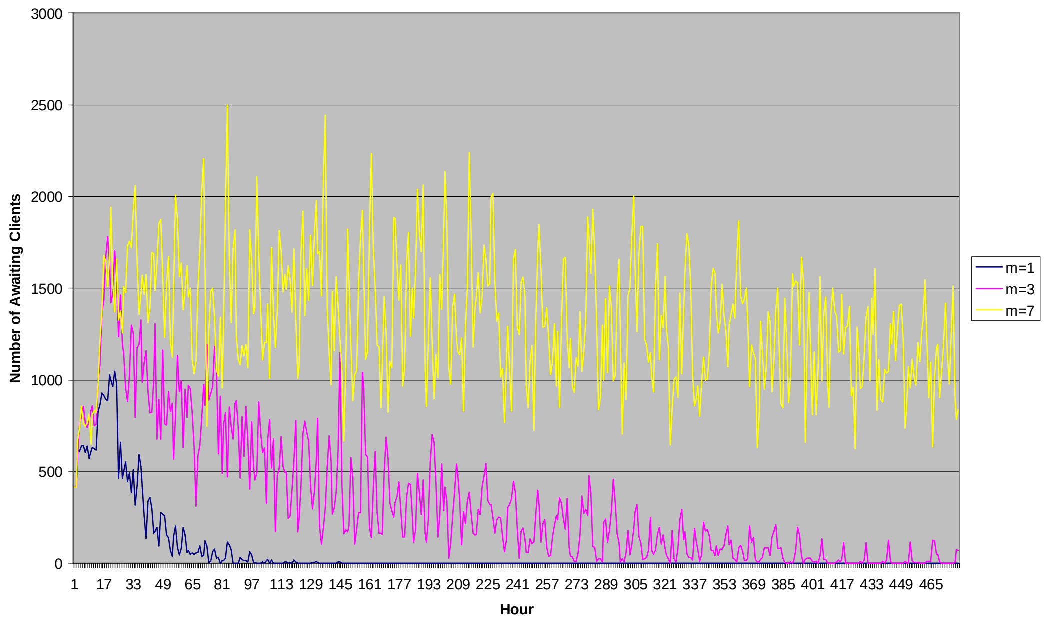


Figure 5.22. Tuning maximum number of seeding intervals

When m is set to 1, the number of Awaiting clients decreases to zero at the 145th hour. In case of two failing synchronization attempts with a client in a row, which has the probability of 0.5, the state of the client is set to Dead. Therefore, in each synchronization phase, the server will stop synchronization with the 50% of the clients as expected. As a result, it is shown that the system cannot survive when m is set to 1 and stops functioning after a while. However, setting m to 3, in our case, doesn't prevent the system from losing synchronization with the clients, but it makes the system to stand for 480 hours. Ultimately, m is set to 7, which results in a more tolerant system and at the end of the simulation for twenty days, the synchronization is still being provided. However, increasing the maximum number of synchronization attempts increases the network traffic dramatically as shown in Table 5.9. Since the server attempts to reach a lost client more times, as the maximum number of synchronization attempts is increased, the increase in network traffic is an expected result.

m	**Number of Synchronization Attempts**		
	Successful	**Failed**	**Total**
1	10,597	10,000	20,597
3	80,230	76,173	156,403
7	319,747	313,491	633,238

Table 5.9. Number of synchronization attempts due to m (maximum number of synchronization attempts)

6 CONCLUSION

In this thesis, we have designed and developed a secure synchronization and querying system, called ESIS. We have also developed a set of application prototypes in order to implement the solutions we propose.

Synchronized password generation prevents local administrator passwords from being compromised because of their limited validities. The secure protocols that constitute the base of the ESIS System depend on the powerful cryptographic algorithms. However, just a secure system lacking performance, scalability, and robustness has no room in the real world. The ESIS System provides scalability and robustness as well as the optimum performance under varying circumstances.

The system provides a secure querying facility to be used by technicians. The multi-factor authentication (token and username and password combination) employed in the secure querying protocol brings the following benefits:

- A technician cannot give his token to somebody in order to let him/her compromise the system because then, he/she cannot authenticate himself/herself.
- If a token is lost, it cannot be exploited and used without the knowledge of the corresponding username and password.
- If a technician loses his/her token, he/she can notice this situation immediately and take some precautions.

In order to improve the performance of the system, the system parameters should be configured very carefully. We have focused on tuning the expiration interval, the seeding interval and the maximum number of seeding attempts and obtained the following results:

- Shortening the expiration interval increases the number of synchronization attempts made by the server as expected. Therefore, configuring this parameter to a relatively small value should be avoided in order to keep the network traffic low.
- The aim is to keep the average number of the successful synchronizations per client at a constant number. This can be achieved in the ESIS System.
- The synchronization attempts can be distributed evenly or in pieces over the time by configuring the seeding interval.
- The probability of failure of a synchronization attempt increases as the quality of the network infrastructure decreases, which results in the increasing number of unreachable clients. Configuring maximum number of synchronization attempts to a relatively big value prevents the clients from shifting to the dead state, which in fact are in the awaiting mode and cannot be reached because of the general network errors. However, setting this parameter to a big value increases the network traffic. Therefore, there is a trade-off here and an extra care should be given in tuning this parameter. In the simulations, setting this parameter to 7 has given a satisfactory output under certain circumstances where the probability of success of a synchronization attempt is 0.5.

The proposed system provides significant benefits. However, there are some open issues that should be resolved. One of these issues is the deployment of the client application. Since $seed_{obf}$ is used as the symmetric key in the initialization of a client, it should be different on each client. To obfuscate different seed values, the application code needs to be re-compiled for each seed, which results in different binaries. Therefore, a different application should be installed to each client, which is very hard to handle in an enterprise network. Management and deployment of client applications should be make practical in order to make the system acceptable for the organizations.

Another issue is the problem of generating tokens. A technician’s token stores the encrypted form of his/her password digest. The digest is encrypted with the RSA private key of the server. If the private key or the technician’s passwords is changed, the token must be updated. It is easy to update or to generate tokens in a central place. However, an enterprise network may be formed of subnetworks that may be geographically separated, so that a central place for managing the tokens is not very practical. Therefore, a flexible solution for management of tokens is needed.

Yet another issue is the availability of the ESIS Server. If the server crashes or becomes disconnected, the password synchronization stops between the clients and the server. To overcome this problem, one or more servers can be employed additionally as ESIS Servers.

To sum up, we have designed and developed a prototype for a secure password synchronization and querying system. As the problems discussed above are resolved, the system will be a superior candidate to be implemented in a real enterprise network.

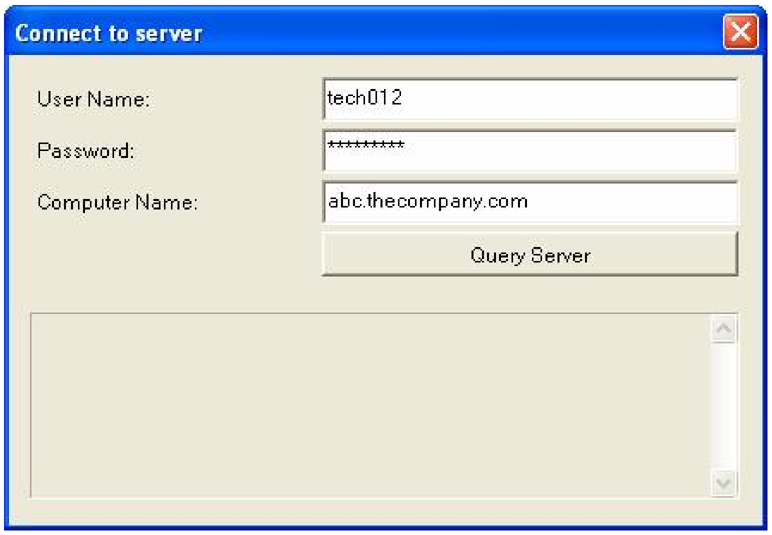
7 REFERENCES

1. National Institute of Standards and Technology, “Data Encryption Standard (DES)”, FIPS Pub 46-2, October 1999
2. National Institute of Standards and Technology, “Advanced Encryption Standard (AES)”, FIPS Pub 197, November 2001
3. William Stallings, “Cryptography and Network Security Principles and Practices”, Third Edition, Prentice Hall, 2003
4. W. Diffie and M. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, November 1976
5. R. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978
6. The Sans Institute, “SANS Glossary of Terms Used in Security and Intrusion Detection”, http://www.sans.org/resources/glossary.php
7. P. Jones, “US Secure Hash Algorithm 1 (SHA1)”, RFC 3174, IETF Network Working Group, http://www.ietf.org/rfc/rfc3174.txt
8. H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD", Fast Software Encryption, LNCS vol. 1039, pp. 71-82
9. National Institute of Standards and Technology, “Digital Signature Standard”, FIPS Pub 186-2, October 2001
10. American National Standards Institute, “Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm”, ANSI X9.62-1998, January 1999
11. P. Jones, “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, IETF Network Working Group, http://www.ietf.org/rfc/rfc2104.txt
12. R. Hamming, “The Art of Probability for Scientists and Engineers”, MA: Addison – Wesley, 1991

13. Larry O'Gorman, “Comparing passwords, tokens, and biometrics for user authentication”, Proceedings of the IEEE, vol. 91, no. 12, pp. 2020- 2030, December 2003
14. R. Morris and K.L. Thompson, “Password security: A case history.”, Commun. ACM, vol. 22, no. 11, pp. 594-597, November 1979
15. Larry O'Gorman, “Comparing passwords, tokens, and biometrics for user authentication”, Proceedings of the IEEE, vol. 91, no. 12, pp. 2019- 2020, December 2003
16. D. L. Jobusch, Oldehoeft, A. E., "A Survey of Password Mechanisms:Weaknesses and Potential Improvements. Part 1", Computers & Security, vol. 8, pp. 587-604, 1989
17. D. L. Jobusch, Oldehoeft, A. E., "A Survey of Password Mechanisms:Weaknesses and Potential Improvements. Part 2", Computers & Security, vol. 8, pp. 675-689, 1989
18. M. Peyravian, Zunic, N., "Methods for Protecting Password Transmission", Computers & Security, vol. 19, pp. 466-469, 2000
19. National Institute of Standards and Technology, “Password Usage”, FIPS Pub 112, May 1985
20. Virgil L. Hovar, “Personal Are Networks, How personal are they?”, SANS Security Essentials GSEC Practical Assignment Version 1.2e, May 2001
21. Art Conklin, Glenn Dietrich, Diane Walz, “Password-Based Authentication: A System Perspective”, Proceedings of the 37th Hawaii International Conference on System Sciences, January 2004
22. Simson Garfinkel, “Anti-Social Engineering – Lessons from reading Mitnick”, CSO Magazine, http://www.csoonline.com/read/100702/machine.html, October 2002

APPENDIX A

Screenshots of the applications

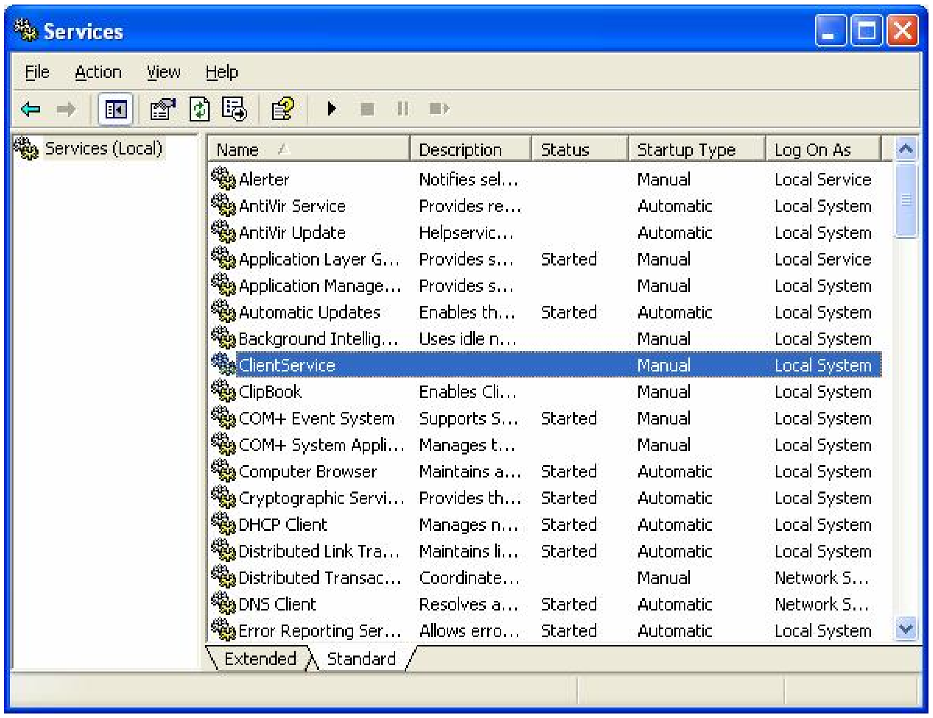

Connect to server

User Name: tech012

Password: **********

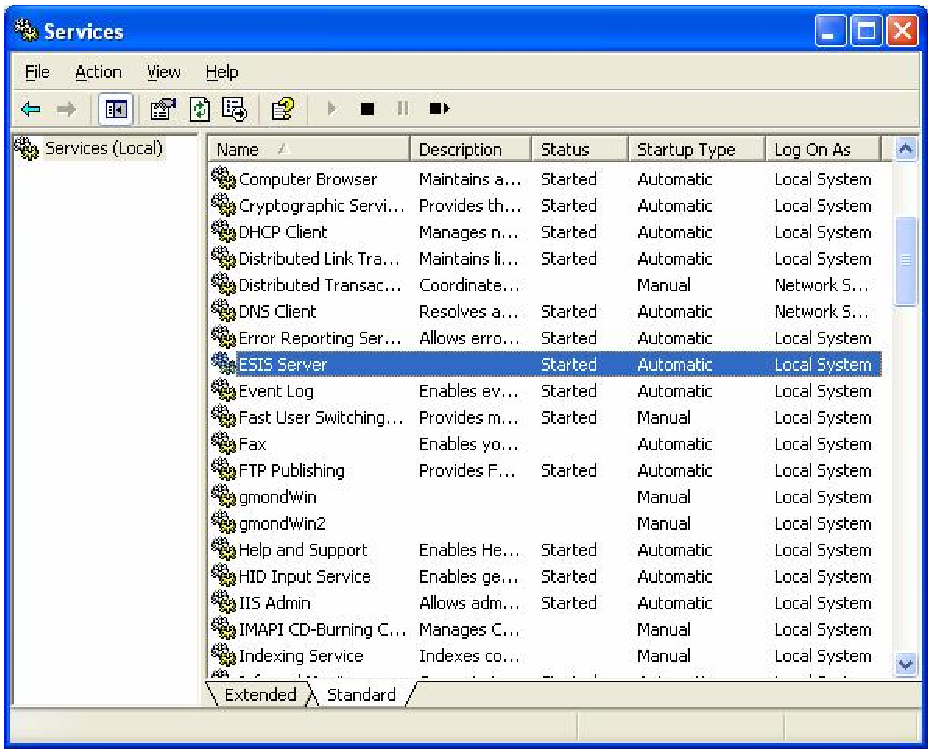
Computer Name: abc.thecompany.com

Technician’s querying application



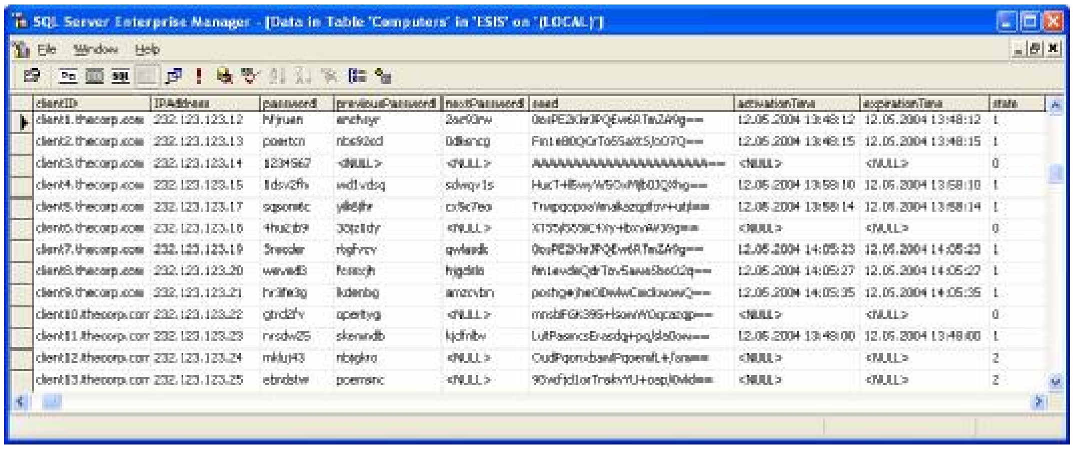
Name	Description	Status	Startup Type	Log On As
Alerter	Notifies sel...		Manual	Local Service
AntiVir Service	Provides re...		Automatic	Local System
AntiVir Update	Helpservic...		Automatic	Local System
Application Layer G...	Provides s...	Started	Manual	Local Service
Application Manage...	Provides s...		Manual	Local System
Automatic Updates	Enables th...	Started	Automatic	Local System
Background Intellig...	Uses idle n...		Manual	Local System
ClientService			Manual	Local System
ClipBook	Enables Cli...		Manual	Local System
COM+ Event System	Supports S...	Started	Manual	Local System
COM+ System Appli...	Manages t...		Manual	Local System
Computer Browser	Maintains a...	Started	Automatic	Local System
Cryptographic Servi...	Provides th...	Started	Automatic	Local System
DHCP Client	Manages n...	Started	Automatic	Local System
Distributed Link Tra...	Maintains li...	Started	Automatic	Local System
Distributed Transac...	Coordinate...		Manual	Network S...
DNS Client	Resolves a...	Started	Automatic	Network S...
Error Reporting Ser...	Allows erro...	Started	Automatic	Local System

Client Service Application in the Windows Services Pane



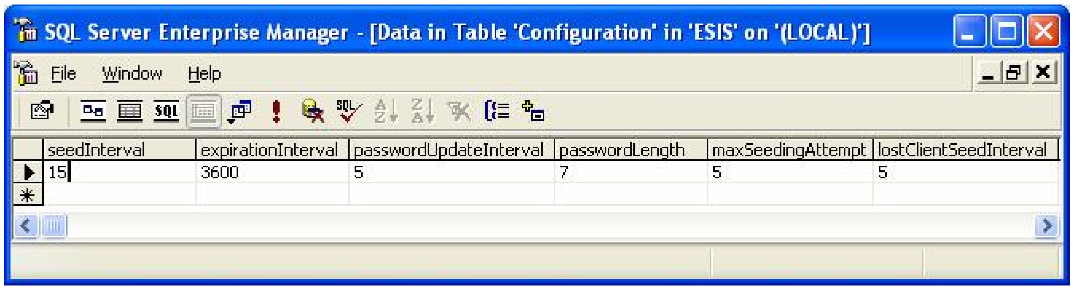
Name	Description	Status	Startup Type	Log On As
Computer Browser	Maintains a...	Started	Automatic	Local System
Cryptographic Servi...	Provides th...	Started	Automatic	Local System
DHCP Client	Manages n...	Started	Automatic	Local System
Distributed Link Tra...	Maintains li...	Started	Automatic	Local System
Distributed Transac...	Coordinate...		Manual	Network S...
DNS Client	Resolves a...	Started	Automatic	Network S...
Error Reporting Ser...	Allows erro...	Started	Automatic	Local System
ESIS Server		Started	Automatic	Local System
Event Log	Enables ev...	Started	Automatic	Local System
Fast User Switching...	Provides m...	Started	Manual	Local System
Fax	Enables yo...		Automatic	Local System
FTP Publishing	Provides F...	Started	Automatic	Local System
gmondWin			Manual	Local System
gmondWin2			Manual	Local System
Help and Support	Enables He...	Started	Automatic	Local System
HID Input Service	Enables ge...	Started	Automatic	Local System
IIS Admin	Allows adm...	Started	Automatic	Local System
IMAPI CD-Burning C...	Manages C...		Manual	Local System
Indexing Service	Indexes co...		Manual	Local System

ESIS Server Application in the Windows Services Pane


SQL Server Enterprise Manager - [Data in Table 'Computers' in 'ESIS' on '(LOCAL)']

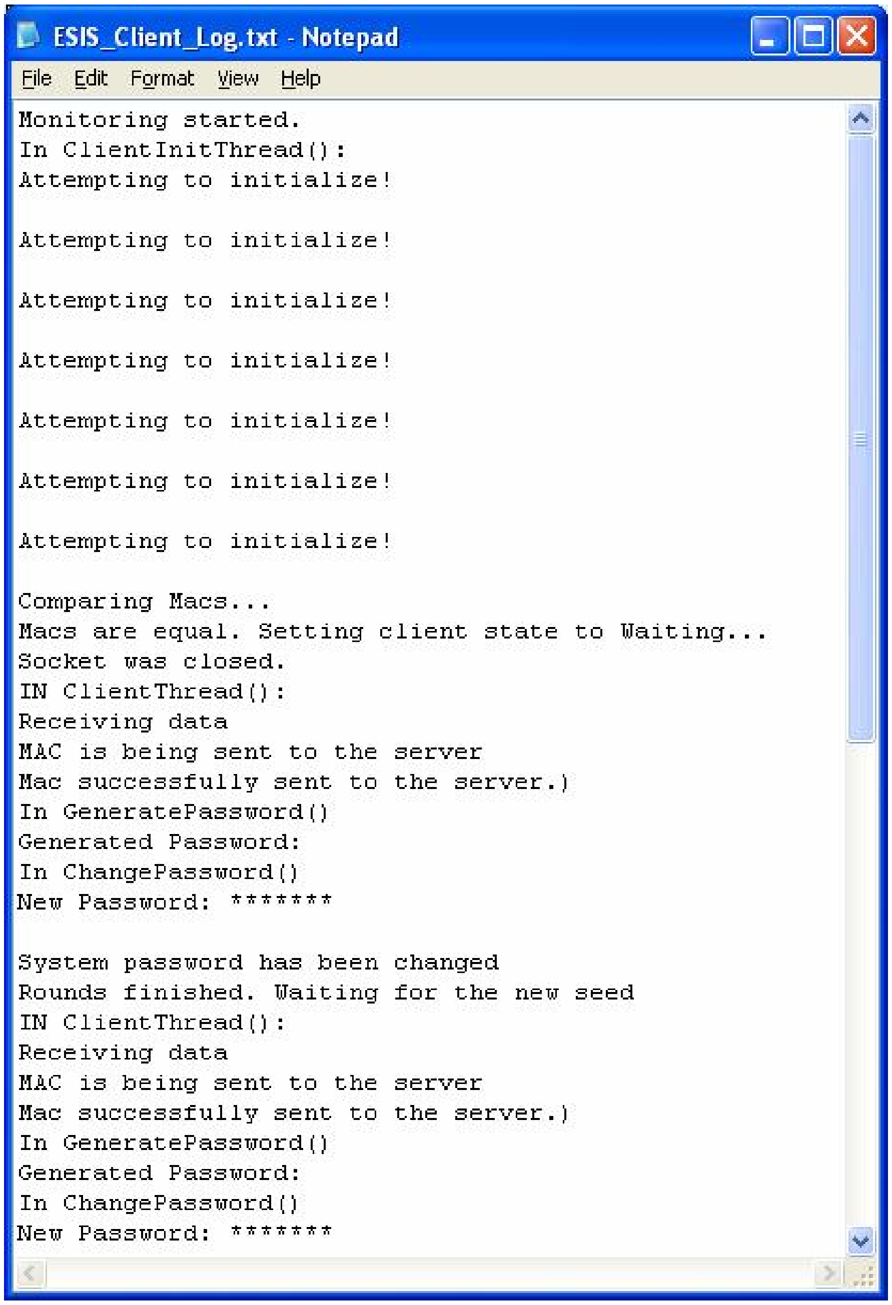
clientID	IPAddress	password	previousPassword	nextPassword	seed	activationTime	expirationTime	state
client1.thecorp.com	232.123.123.12	hfjruen	anchvyr	2ar93rw	0esPE2KIr3PQEw6ATmZAYg==	12.05.2004 13:48:12	12.05.2004 13:48:12	1
client2.thecorp.com	232.123.123.13	poertcn	nbe92cd	0dlkncg	FIn1eB0QGrTo65axtSJoO7Q==	12.05.2004 13:48:15	12.05.2004 13:48:15	1
client3.thecorp.com	232.123.123.14	1234567	<NULL>	<NULL>	AAAAAAAAAAAAAAAAAAAAAA==	<NULL>	<NULL>	0
client4.thecorp.com	232.123.123.15	lldsv2fh	wd1vdsq	sdwqv1s	HucT+lEwyWSOvMjb0JQXhg==	12.05.2004 13:58:10	12.05.2004 13:58:10	1
client5.thecorp.com	232.123.123.17	sqsoretc	ylk6jhr	cx5c7eo	TruqocpoaWnalkszqpfov+utj==	12.05.2004 13:58:14	12.05.2004 13:58:14	1
client6.thecorp.com	232.123.123.18	4huzb9	30tzlldy	<NULL>	XT55f6S59C4Xy+lbcvAW39g==	<NULL>	<NULL>	0
client7.thecorp.com	232.123.123.19	3reoder	rfgfvcv	qwlasdk	0esPE2KIr3PQEw6ATmZAYg==	12.05.2004 14:05:23	12.05.2004 14:05:23	1
client8.thecorp.com	232.123.123.20	wevad3	fcsscjh	hjgdsb	fm1evdaQdrTevSaaa5beO2q==	12.05.2004 14:05:27	12.05.2004 14:05:27	1
client9.thecorp.com	232.123.123.21	hr3fe3g	lkdenbg	amzcvbn	poshg#jheODwlwCwdiowowQ==	12.05.2004 14:05:35	12.05.2004 14:05:35	1
client10.thecorp.com	232.123.123.22	gtrd2fv	opertyg	<NULL>	mnsbFGK39S+lsowWOqcazqp==	<NULL>	<NULL>	0
client11.thecorp.com	232.123.123.23	rxsdw25	skenwdb	kjdfnbv	LulPasncsErasdq+pq/ds0ow==	12.05.2004 13:48:00	12.05.2004 13:48:00	1
client12.thecorp.com	232.123.123.24	mkluj43	nbgkro	<NULL>	OudPqonxbawPqoenfL+/ora==	<NULL>	<NULL>	2
client13.thecorp.com	232.123.123.25	ebndstw	poemanc	<NULL>	90wfjd1orTrakvYU+osp/0wdw=	<NULL>	<NULL>	2

The ESIS Client Data on the MS SQL Server Database


SQL Server Enterprise Manager - [Data in Table 'Configuration' in 'ESIS' on '(LOCAL)']

seedInterval	expirationInterval	passwordUpdateInterval	passwordLength	maxSeedingAttempt	lostClientSeedInterval
15	3600	5	7	5	5

The ESIS Configuration Data on the MS SQL Server Database



```
Monitoring started.
In ClientInitThread():
Attempting to initialize!

Attempting to initialize!

Attempting to initialize!

Attempting to initialize!

Attempting to initialize!

Attempting to initialize!

Attempting to initialize!

Comparing Macs...
Macs are equal. Setting client state to Waiting...
Socket was closed.
IN ClientThread():
Receiving data
MAC is being sent to the server
Mac successfully sent to the server.)
In GeneratePassword()
Generated Password:
In ChangePassword()
New Password: *******

System password has been changed
Rounds finished. Waiting for the new seed
IN ClientThread():
Receiving data
MAC is being sent to the server
Mac successfully sent to the server.)
In GeneratePassword()
Generated Password:
In ChangePassword()
New Password: *******
```

The Client Service Application Log File

7 REFERENCES

1. National Institute of Standards and Technology, “Data Encryption Standard (DES)”, FIPS Pub 46-2, October 1999
2. National Institute of Standards and Technology, “Advanced Encryption Standard (AES)”, FIPS Pub 197, November 2001
3. William Stallings, “Cryptography and Network Security Principles and Practices”, Third Edition, Prentice Hall, 2003
4. W. Diffie and M. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol. 22, no. 6, pp. 644-654, November 1976
5. R. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems”, Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978
6. The Sans Institute, “SANS Glossary of Terms Used in Security and Intrusion Detection”, http://www.sans.org/resources/glossary.php
7. P. Jones, “US Secure Hash Algorithm 1 (SHA1)”, RFC 3174, IETF Network Working Group, http://www.ietf.org/rfc/rfc3174.txt
8. H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD", Fast Software Encryption, LNCS vol. 1039, pp. 71-82
9. National Institute of Standards and Technology, “Digital Signature Standard”, FIPS Pub 186-2, October 2001
10. American National Standards Institute, “Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm”, ANSI X9.62-1998, January 1999
11. P. Jones, “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, IETF Network Working Group, http://www.ietf.org/rfc/rfc2104.txt
12. R. Hamming, “The Art of Probability for Scientists and Engineers”, MA: Addison – Wesley, 1991

13. Larry O'Gorman, “Comparing passwords, tokens, and biometrics for user authentication”, Proceedings of the IEEE, vol. 91, no. 12, pp. 2020- 2030, December 2003
14. R. Morris and K.L. Thompson, “Password security: A case history.”, Commun. ACM, vol. 22, no. 11, pp. 594-597, November 1979
15. Larry O'Gorman, “Comparing passwords, tokens, and biometrics for user authentication”, Proceedings of the IEEE, vol. 91, no. 12, pp. 2019- 2020, December 2003
16. D. L. Jobusch, Oldehoeft, A. E., "A Survey of Password Mechanisms:Weaknesses and Potential Improvements. Part 1", Computers & Security, vol. 8, pp. 587-604, 1989
17. D. L. Jobusch, Oldehoeft, A. E., "A Survey of Password Mechanisms:Weaknesses and Potential Improvements. Part 2", Computers & Security, vol. 8, pp. 675-689, 1989
18. M. Peyravian, Zunic, N., "Methods for Protecting Password Transmission", Computers & Security, vol. 19, pp. 466-469, 2000
19. National Institute of Standards and Technology, “Password Usage”, FIPS Pub 112, May 1985
20. Virgil L. Hovar, “Personal Are Networks, How personal are they?”, SANS Security Essentials GSEC Practical Assignment Version 1.2e, May 2001
21. Art Conklin, Glenn Dietrich, Diane Walz, “Password-Based Authentication: A System Perspective”, Proceedings of the 37th Hawaii International Conference on System Sciences, January 2004
22. Simson Garfinkel, “Anti-Social Engineering – Lessons from reading Mitnick”, CSO Magazine, http://www.csoonline.com/read/100702/machine.html, October 2002