ON THE EFFICIENCY OF AUTHENTICATION PROTOCOLS, DIGITAL SIGNATURES AND THEIR APPLICATIONS IN E-HEALTH: A TOP-DOWN APPROACH

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

BY

KEMAL BIÇAKCI

IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2003

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

Kemal Bıçakcı
Author

Approval of the Graduate School of Informatics.

Prof. Dr. Neşe Yalabık
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Onur Demirörs
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Nazife Baykal
Supervisor

Examining Committee Members

Prof. Dr. Semih Bilgen

Assoc. Prof. Dr. Nazife Baykal

Assist. Prof. Dr. Y. Murat Erten

Dr. Kıvanç Dinçer

Dr. Altan Koçyiğit

ABSTRACT

ON THE EFFICIENCY OF AUTHENTICATION PROTOCOLS, DIGITAL SIGNATURES AND THEIR APPLICATIONS IN E-HEALTH: A TOP-DOWN APPROACH

Bıçakcı, Kemal

Ph.D., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Nazife Baykal

September 2003, 164 pages

Choosing an authentication protocol or a digital signature algorithm becomes more challenging when performance constraints are of concern. In this thesis, we discuss the possible options in a top-down approach and propose viable alternatives for the efficiency criteria.

Before all the technical discussions, we argue that identifying prerequisites, threats and risks on an organizational context has utmost importance so that effective solutions can be delivered at a reasonable cost. For instance, one approach to solve the performance problem is to relax the security requirements if it is allowable and use one-time passwords as the more efficient entity authentication protocol. SCOTP is the first protocol proposed in this study which improves the security and flexibility of one-time passwords.

After requirements are set up, another high-efficiency solution is based on new designs of improved protocols. These new protocols might utilize the trade-offs between efficiency of distinct system parameters such as communication versus computational load. SAOTS is our new protocol designed to improve the performance and increase the round efficiency of server-assisted signature protocols.

With an example in e-health, we also demonstrate that efficiency can be provided on the implementation level as well, the last step in the chain. EVEREST is the third proposal in this thesis which improves the real-time efficiency of digital signatures concerning the fact that the medical images are huge in size and to verify the signature a considerable amount of time is spent to compute the hash of the image file.

Keywords: Cryptography, Network Security, Digital signature, Authentication, Server assisted signature, One-time password, Teleradiology

ÖZ

TANIMA PROTOKOLLERİ, SAYISAL İMZALAR, VE BUNLARIN E-SAĞLIK UYGULAMALARININ VERİMİ ÜZERİNE: YUKARDAN AŞAĞIYA BİR YAKLAŞIM

Bıçakcı, Kemal

Doktora, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Doç. Dr. Nazife Baykal

Eylül 2003, 164 sayfa

Bir tanıma protokolünün veya sayısal imza algoritmasının seçimi performans kısıtlamaları söz konusu olduğunda daha da zorlaşmaktadır. Bu tezde, olası tercihler yukardan-aşağıya yaklaşımı ile tartışılıp verim ölçütü için uygulanabilir alternatifler önerilmektedir.

Teknik tartışmalardan önce, gereksinim, tehdit ve risklerin organizasyonel bağlamda tespit edilmesinin çok önemli olduğu ve ancak bu şart ile etkin çözümlerin makul maliyetlerle sağlanabileceği belirtilmiştir. Mesela, performans problemini çözmek için yaklaşımlardan bir tanesi eğer mümkünse güvenlik gereksinimini biraz gevşetmek ve tek-zamanlı parolaları daha verimli kişi tanıma protokolü olarak kullanmaktır. Tek-zamanlı parolaların güvenlik ve esnekliğini geliştiren SCOTP bu çalışmada önerilen ilk protokoldür.

Gereksinimler saptandıktan sonra, verim için bir diğer uygulanabilir çözüm yeni ve gelişmiş protokol tasarımlarına dayanmaktadır. Bu yeni protokoller sistemdeki hesaba dayalı ve iletişimsel yük gibi farklı parametreler arasındaki verim değiş-tokuşlarından yararlanabilir. SAOTS adını verdiğimiz yeni protokol sunucu destekli imza protokollerinde gecikme ve basamak verimini arttırmak için tasarlanmıştır.

E-sağlık'da verilen bir örnek ile verimin zincirdeki en son halka, gerçekleştirme aşamasında da sağlanabileceği gösterilmektedir. Bu tezdeki üçüncü ve son öneri olan EVEREST yine sayısal imzalarda gerçek-zaman verimini geliştirmektedir. Fakat bu defa tıbbi imgelerin ebatça çok büyük olması ve imzanın doğrulanması için hatırı sayılır bir sürenin imge dosyasının özetinin hesaplanması için harcandığı gerçeğinden hareket edilmiştir.

Anahtar Kelimeler: Kriptografi, Ağ güvenliği, Sayısal imza, Tanıma, Sunucu yardımlı imza, Tek-zamanlı parola, Teleradyoloji

ACKNOWLEDGMENTS

"O God, through Your mercy, appoint us among those who give thanks, O Most Merciful of the Merciful! Glory be unto You! (We have no knowledge but that which You have taught us; indeed, You are All-Knowing, All-Wise [1])."[2]

It is a pleasure for me to express my sincere gratitude to my supervisor Dr. Nazife Baykal for her patience, encouragement and guidance throughout the study. I greatly appreciate her share in the every step taken in the development of the thesis.

Also, I owe much to the committee members Dr. Semih Bilgen, Dr. Kıvanç Dinçer, Dr. Altan Koçyiğit and Dr. Y. Murat Erten for helpful comments and discussions.

I also want to thank Enver Çavuş for his help on conducting the experiments. I am grateful to Dr. Albert Levi and Dr. Yongdae Kim for the reviewing of our papers.

I would further like to thank to Turan Demirci, Umut Orguner and my office mates Nart Bedin Atalay, Umut Özge, Kaan Bıyıkoğlu and Sevgi Özkan for their encouragement and support.

[1] Qur'an, 2:32.
[2] Letters, Said Nursi.

I appreciate Sibel Gülnar and Mukaddes Yenilmez in the institute for their kindness since the beginning of my Ph.D. study.

I am also indebted to my previous advisors Dr. Levent Ertaul, Dr. Brian Tung and Dr. Gene Tsudik.

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

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

ACSU	Attribute Certificate Service Unit.
AES	Advanced Encryption Standard.
BAN	Burrows Abadi Needham.
BW	Bandwidth.
CA	Certificate Authority.
CCITT	Comite Consultatif Internationale de Telegraphie et Telephonie.
CGI	Common Gateway Interface.
CRL	Certificate Revocation List.
CT	Computed Tomography.
DES	Data Encryption Standard.
DICOM	Digital Imaging and Communications in Medicine.
DSA	Digital Signature Algorithm.
EVEREST	Efficient Verification of Electronic (Digital) Signatures in Real-time Teleradiology.
HIPAA	Health Insurance Portability and Accountability Act.
ITU	International Telecommunication Union.
JCE	Java Cryptography Extension.
KDC	Key Distribution Center.
LAN	Local Area Network.
LDAP	Lightweight Directory Access Protocol.
MAC	Message Authentication Code.
MD5	Message Digest Five.
MRI	Magnetic Resonance Imaging.
NBS	National Bureau of Standards.
NIST	National Institute of Standards and Technology.
NSA	National Security Agency.

OCSP	Online Certificate Status Protocol.
OTP	One-Time Password.
OTS	One-Time Signature.
OWF	One-Way Function.
PACS	Picture Archiving and Communications System.
PGP	Pretty Good Privacy.
PKI	Public Key Infrastructure.
PMI	Privilege Management Infrastructure.
PRIV	Private Key.
PUB	Public Key.
RA	Registration Authority.
RSA	Rivest Shamir Adleman.
SAOTS	Server Assisted One-Time Signature.
SAS	Server Assisted Signature.
SC	Signature Chain.
SCOTP	Signature Chain Based One-Time Password.
SDK	Software Development Kit.
SHS	Secure Hash Standard.
SSH	Secure Shell.
TTP	Trusted Third Party.
VS	Verifiable Server.
WAN	Wide Area Network.

CHAPTER I

INTRODUCTION

Open, Sesame!

- Ali Baba and Forty Thieves

Privacy and anonymity might be important for our social and business well-being, but authentication is essential for survival. Authentication is about the continuity of relationships, knowing who to trust and who not to trust, making sense of a complex world.

- Bruce Schneier

"Security" means to protect certain assets in spite of certain threats and attacks. In security engineering, the focus is on threats caused by intelligent adversary whereas accidental behaviors or statistical mistakes are more related to scientific disciplines of safety or fault tolerance.

For centuries, we have protected our assets by traditional means such as locks, fences, seals, spoken passwords etc. To a first approximation, online society is similar to offline society, digital threats mirror the threats in the physical world and the analogs of traditional protection methods are applicable in this new world. However three important unique characteristics of the cyberspace make the security problem more horrifying and these are basically the reasons the terms “computer security” and “network security” gain so much popularity in recent years [1]:

- Automation: With computers, attacks that would not have been possible or were too marginal to notice become a major threat.
- Action at a Distance: Attackers do not have to be near their targets.
- Technique Propagation: Attack tools can easily be propagated through Internet.

Attacks are changing in nature and become much more dangerous, meanwhile many of us are today critically dependant on the security of the surrounding systems. The security measures implemented in these systems can be categorized into three groups:

- Prevention
- Detection
- Reaction

Usually, reactions follow the detection. A simple example for reaction is calling the police after you detect that something has been stolen.

I.1 Information Security Targets

Before choosing the specific countermeasure, we should decide on our security goal/target by answering the question, "what kind of security do we need?"

By examining how information assets can be compromised, another classic list in the security world distinguishes three main information security targets [2]:

- Confidentiality: prevention of unauthorised disclosure of information.
- Integrity: prevention of unauthorised modification of information.
- Availability: prevention of unauthorised witholding of information.

I.2 Authentication Protocols

One of the security goals is "authentication", to establish the identity of a communication partner.

You can ask "what happened to the authentication target? I could not see in the list."

- It is in the list, but I have made it invisible for security reasons...:)

Switching back to serious-mode, authentication is not in the list not because it is unimportant but because it has relationships with all the security goals listed. By a closer look to the list we see that there is a common word "unauthorized" in all the items. Only if we can securely authenticate someone, we can decide on "authorization", whether he is allowed or not. In other words "authorization" follows "authentication". That is why "authentication" is the key for information security and if the authentication mechanism is compromised, the rest of the security measures are bypassed as well. In spite of military domain where sometimes security and secrecy might even be used as synonyms, authentication becomes far more important than secrecy in modern business in the information age we are living in.

There are two well-known types of authentication:

- Entity authentication: Gaining assurances that the identity of the claimant is as declared, thereby preventing impersonation.
- Message authentication: While entity authentication typically involves no meaningful message other than the claim of being a particular entity, message authentication does [3].

Another major difference between these two is that message authentication itself does not provide any timeliness guarantees whereas entity authentication confirms the identity in real-time (while the verifying entity awaits).

Lastly, as with any protocol, a security protocol compromises a series of predetermined steps designed to complete a task. Just like a communication protocol designed to establish a communication between participating entities, the task in an "authentication protocol" is to authenticate the user(s).

I.3 Digital Signatures

Handwritten signatures have long been used to authenticate the messages signed but the means to provide digital signatures for computer communication that is roughly equivalent to handwritten signatures on paper documents became available with the advances in modern cryptography. Whether we use a handwritten or a digital signature, other than authenticating the message signed, the signature also ensures the message integrity and solves the non-repudiation problem. As we will see in the next chapter, in cryptography world, while there are other tools like message authentication codes (MACs) to ensure data integrity and authentication, digital signatures are better in one important respect. They are the only ones addressing the issue of non-repudiation.

"Nonrepudiation" means blocking a sender's false denial that he or she signed a particular document, thus enabling the recipient to easily prove that the sender actually did sign the document. Authentication is a prevention-type countermeasure whereas nonrepudiation is an issue more related to reaction e.g., in case of a dispute a digital signature can be submitted to the arbiter as a proof.

The concepts of message authentication and message integrity have close meanings and are sometimes confused. Integrity is not concerned with the origin of the data but whether it has been modified or not [1] (It is different than "accuracy" which conveys the notion of precision). On the other hand, it makes no sense to provide the message authentication without also guaranteeing integrity of the message. Our discussion about the relationships between information security targets is summarized in Figure I.1.

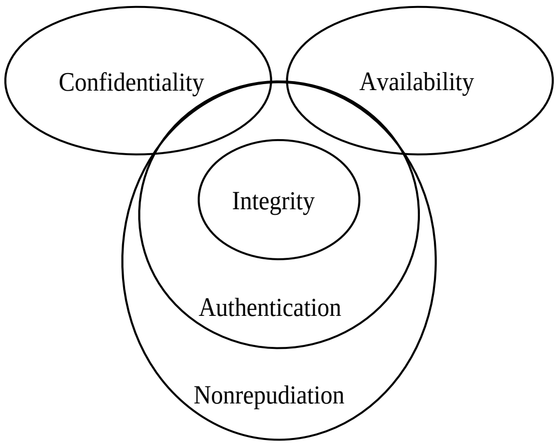

Figure I.1: Information Security Targets

I.4 Efficiency Considerations

Broadly speaking, in a security protocol, we can define three dimensions that have to be considered:

- **Security**: The quality or state of being protected from unauthorized actions. Absolute security may in practice be impossible to reach; thus the security 'quality' could be relative.
- **Efficiency**: Skillfulness in avoiding wasted time, effort and other resources.
- **Others**: In a security protocol one might desire other features such as flexibility, scalability, survivability, user-friendliness etc.

We define some of the aforementioned other features which are relevant to our future discussion as follows:

Survivability: Ability to continue to function in different conditions.

Flexibility: Ability to adjust readily to different conditions in terms of security and efficiency (In our definition a survivable inflexible protocol is the one that becomes less secure and/or less efficient in a different condition).

User-friendliness: Ability to be easily used by non-specialists.

After the desired level of security is specified and the desired features are determined, the goal in designing a security protocol is to provide all these in the most "efficient" way.

In the early days of computers where saving a few clock cycles has a meaning and available cryptographic tools are unoptimized and very slow, being inefficient might mean being unusable. Today we see that in spite of high-performance computing and really fast cryptographic algorithms, efficiency still remains a remarkable concern in designing a security protocol. This is because with more efficient security protocols three important targets can be attained:

- Cost cuts are possible e.g., by buying a less expensive and less powerful web server.
- "Pervasive computing" vision can be realized where computer applications are hosted on a wide range of platforms, including many that are small, mobile and regarded today as devices having only limited computational capabilities.
- Security level can be increased e.g., by using a longer key length.

Traditionally, the word "efficient" without an affix is used to mean efficient in terms of computational delay. For instance if one says Protocol A is more efficient than Protocol B, he means that executing Protocol A takes less time than Protocol B. However considering only the delay-efficiency (sometimes called real-time efficiency) might mislead us in some situations because there are other constraints (e.g., battery power of mobile devices) we should take into consideration when designing a security protocol. Moreover, most of the time when a new efficient authentication protocol or a digital signature scheme is to be designed we face trade-offs between efficiency of distinct system parameters such as communication versus computational load. Hence throughout this thesis when we refer to efficiency if we do not use it in its general meaning we will try to make it clear what kind of efficiency we are talking about.

I.5 E-Health

Before explaining you what this thesis is all about, we would like to shortly introduce one remaining term used in the thesis title; “e-health”.

Barely in use before 1999, the term e-health now seems to serve as a general ”buzzword,” in line with other “e-words” used to characterize not only ”Internet medicine”, but also virtually everything related to computers and medicine [4]. The term was apparently first used by industry leaders and marketing people rather than academics. Some feels that the term should remain in the realm of the business and marketing sector and should be avoided in scientific medical literature and discourse. However, the term has already entered the scientific literature. As of 22 June 2003, 156 Medline-indexed articles contain the term ”e-health” in the title or abstract (two years ago this number was only 76 [4]).

Consequently, the unstoppable rise of "e-health" stimulates us to prefer it over other alternatives like telemedicine.

According to a recent study [1], if health organizations in United States had spent \$50 billion each year on information technology, they might have saved \$270 billion each year (Up to our knowledge, a similar study has not been done for Turkey yet). Only these figures are sufficient to demonstrate why healthcare industry is labeled as being desperately in need of the efficiencies provided by information systems. However, security issues in e-health systems have taken center stage in recent years and due to the importance and the complexity of the problem most experts see it as one of the significant challenges to successful e-health projects.

I.6 Scope of This Thesis

In this thesis, our aim is to discuss the "efficiency" dimension of authentication protocols, digital signatures and their applications in e-health with some originally developed/designed examples in a top-down approach. More specifically, as illustrated in Figure I.2 in a big picture we first identify three main options in order to provide the efficiency required at any kind [2]:

- **Requirement Analysis**: Before all the technical discussions, we argue that identifying requirements (prerequisites, threats, risks etc.) on an

[1] R. Wagner, "US Healthcare", white paper, March 2001, Available from http://www.pkiforum.org/pdfs/healthcarenote.pdf, Last access: September 17, 2003.

[2] This classification is quite natural and very well-known in the software engineering discipline however incorporating security engineering in software engineering process is a relatively new concept. Refer to [5] for more information.

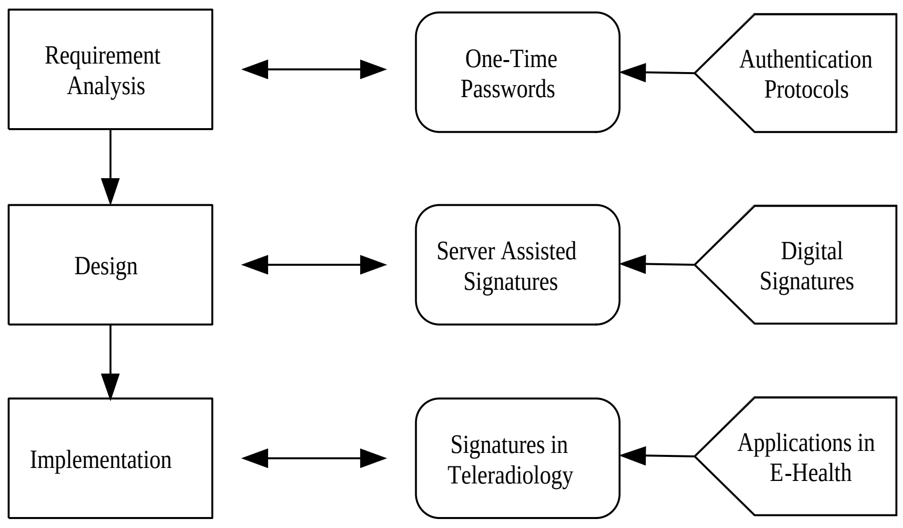

Figure I.2: Alternatives for Efficiency

organizational context has utmost importance so that effective solutions can be delivered at a reasonable cost. For instance, one obvious approach to solve the performance problem is to relax the security requirements if it is allowable.

- **Design**: After requirements are set up, another viable solution for the efficiency is based on new designs of improved protocols.

- **Implementation**: By taking into account the unique characteristics of the application and with some implementation tricks, efficiency can be provided on the the last step in the chain as well.

The objective of this dissertation is to illustrate the viability of each of the aforementioned alternatives with some originally developed examples as shown on the righthand side of Figure I.2.

First of all, we argue that "one-time passwords" can be used as an efficient entity authentication protocol when in the requirement analysis phase we decide that some sophisticated attacks are not of our concern. Then after proposing a new construction called Signature Chain (Infinite Length Hash Chain) we design the SCOTP (**S**ignature **C**hain based **O**ne-**T**ime **P**assword) protocol to improve the security and flexibility of one-time passwords [3].

Secondly, we design a new efficient server assisted signature protocol called SAOTS (**S**erver **A**ssisted **O**ne-**T**ime **S**ignatures). We believe that SAOTS protocol is very promising since it is more computation, storage and power efficient than previous verifiable server assisted signature protocols.

Thirdly, we have investigated that in real-time teleradiology (a branch of e-health which involves digital radiographic images), a considerable amount of time is spent to verify a digital signature since the image size is huge (tens of megabytes). Motivated by this fact, we propose, implement and evaluate the performance of a communication and delay-efficient methodology for verification called EVEREST (**E**fficient **VER**ification of **E**lectronic (digital) **S**ignatures in real-time **T**eleradiology)

EVEREST is proposed primarily for teleradiology services whereas SAOTS and SCOTP are for general use. Last but not the least, in this thesis we discuss the applicability and benefits of SCOTP and SAOTS in e-health applications.

[3] One-time passwords as a concept can provide efficiency, however improving the efficiency is not the objective of SCOTP protocol. Don't let that confuse you.

I.7 Organization of This Thesis

In this introductory chapter, the basic concepts in security are presented, authentication protocols, digital signatures and e-health systems are introduced and the importance of efficiency is emphasized. The scope and the organization of the thesis is provided.

Chapter 2 is for background information. However we prefer to provide the background material in this chapter only when this material is a prerequisite to understand two or more subsequent chapters. Otherwise the background is given in the beginning of subsequent chapters. In addition, the related work and literature survey is also included in the first sections of following three chapters.

Chapter 3 is about entity authentication and one-time passwords. We propose the construction of Signature Chain and introduce the SCOTP protocol in this chapter. At the end possible solutions to establish a survivable authentication framework in e-health systems including the one based on one-time passwords are discussed.

To reduce the costs of generating digital signatures one viable method is to employ a third party; the server. Chapter 4 introduces server assisted signatures first and then presents SAOTS, our new design. Similar to Chapter 3, Chapter 4 also covers the benefits of SAOTS in e-health applications.

Chapter 5 is concerned with efficient verification of signatures in real-time teleradiology applications. Our third proposal, EVEREST is proposed here.

In Chapter 6, we conclude by summing up our work and discussing future possibilities for research. The conclusions and future works discussed in this

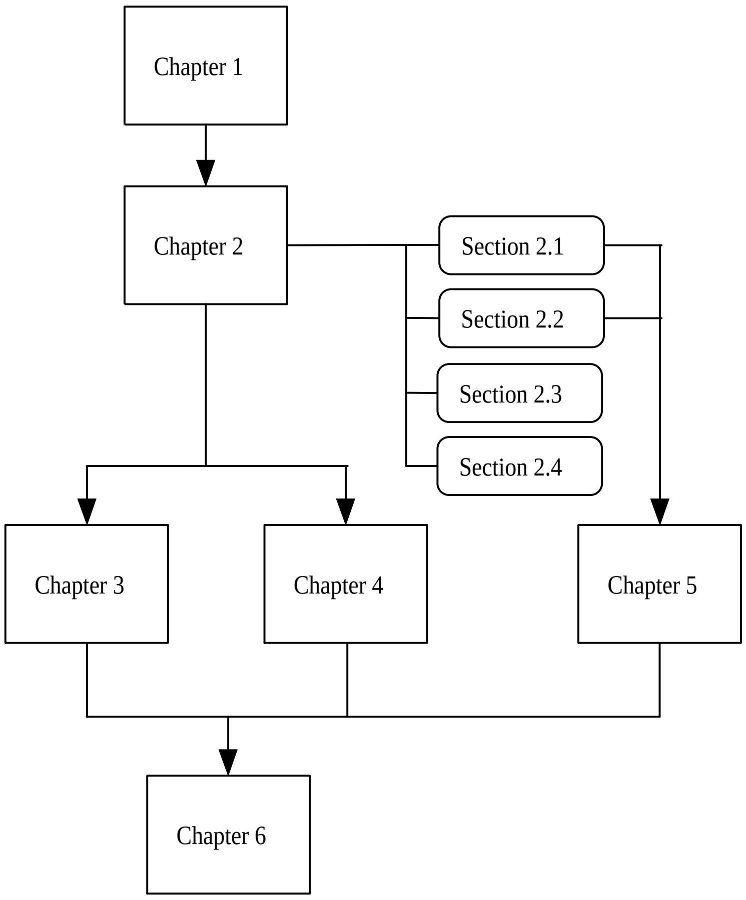

Figure I.3: Organization of The Thesis

chapter restate the important points given in the summary sections of previous three chapters and also present some other broader issues and open problems.

Figure I.3 illustrates the structure of the thesis and the dependency of its chapters.

CHAPTER II

BACKGROUND

In theory there is no difference between theory and practice. In practice there is.

- Yogi Berra

Just as a doctor needs to understand physiology as well as surgery, so a security engineer needs to be familiar with cryptology as well as computer security (and much else).

- Ross J. Anderson

Cryptography is where security meets mathematics [6]. The field is an incredibly broad one, with a history spanning hundreds, even thousands of years. Traditionally its main objective is to design ciphers to protect sensitive data and make it utterly unintelligible to anyone but the intended recipient. More or less this has continued until Diffie and Hellman invented the concept of public key cryptography in 1976 [7].

During the last 25-30 years, public academic research in cryptography has exploded. Consequently security protocols that could perform the most fantastical interactions become possible (e.g., digital cash, e-voting, online auctions etc.)

In our opinion today the real problem is not what cryptography can do but whether it is understood and applied correctly or not. For instance most people are not aware of the danger of using “home-made crypto”. The reason is that no matter how unskilled one can design a cryptographic primitive that he himself cannot break. The question to ask them is that “if you do not know much of the theory behind crypto, would you expect to do any better at designing than the best cryptographic minds of World War II?”

In the implementation of protocols proposed in this dissertation, we use published, well-used algorithms that has been well scrutinized by respected cryptographers over a period of years e.g., DSA [8], SHS [9].

Since there are a bunch of very good introductory textbooks on cryptography providing very detailed information especially about the succeeding two sections, for the sake of brevity we provide only a short summary background information here. This information is for describing how cryptographic tools can be used rather than on the mathematical details of the algorithms. We defer to [1, 3, 6, 10, 11, 12, 13] for a comprehensive treatment of the subject.

II.1 Basic Cryptographic Tools

Security protocols are built on lower level cryptographic tools. Six tools - symmetric encryption, public-key encryption, message authentication codes, random number generators, one-way hash functions, and digital signature schemes- comprise the toolbox of cryptography consumers [1]. Anyone of these six tools may be used in an authentication protocol and we will now introduce briefly each of them:

II.1.1 Symmetric Encryption

Historically, cryptography has been used for one thing: to keep secrets. Suppose there is a message, sometimes called the plaintext, that someone wants to keep secure. Maybe the someone (we will call her Alice) wants to send it to someone else (we will call him Bob). What she does not want is for anyone other than Bob to be able to read the message. Let's look at what must happen for Alice to send a message to Bob securely:

1. Alice and Bob agree on a cipher (cryptosystem) e.g., DES [14], AES [15].
2. Alice and Bob agree on a key.
3. Alice takes her plaintext message and encrypts it using the encryption algorithm and the key to form the ciphertext.
4. Alice sends the ciphertext message to Bob.
5. Bob decrypts the ciphertext message with the same algorithm and the same key and reads it.

One notable point here is that the security depends on the secrecy of the key rather than the cipher itself. This is also the reason symmetric encryption is also known as “secret key cryptography”. Keys relieve us of the need to worry about the algorithm used. Which is more trustworthy? a secret algorithm or an algorithm that would do its job even if everybody in the world knows exactly how it works.

The concept of “key” also brings one of the fundamental problems in cryptography known as “key distribution”. Once keys are distributed securely, the rest is much more straight-forward. Suppose n users would like to securely communicate with each other using secret key cryptography then each user would need to know $n-1$ keys, one for each other user. If a new user joins then n keys need to be generated for that new user to have a shared secret with each of the other users.

At this point, the concept of “key length” should be explained. To break a cipher, one obvious thing that an attacker can do is to try every possible key. Not using any intelligence at all, this attack is named as “brute force attack”. The most secure (secret-key) cipher is the one which can not be broken by a less effort than the brute force attack. The susceptibility of a cipher to a brute force attack is directly related to the key length e.g., an algorithm with a 40-bit key length can be broken by about a trillion trials. Of course the longer the key, the harder to break the system by a brute-force attack, but on the other hand using a key longer than what we need slows down the encryption/decryption and is inconvenient to store and to transmit.

The length of the key is a major design issue today. Computers are doubling their speed every year and the ciphers, which were secure at the time of their designs, are not secure anymore. For instance IBM's original submission to NBS (National Bureau of Standards) had a 112-bit key. By the time this became a standard as Data Encryption Standard (DES) [14], that was reduced to a 56-bit key. Many cryptographers argued for the longer key. Their arguments centered on the possibility of a brute-force attack. In 1977, it was argued that a special-purpose DES-cracking parallel computer could recover the key in a day at a cost of $20 million [12], which was out of reach for everybody except organizations like the NSA (National Security Agency). Today, it is possible to build such machines in a much cheaper way. The cost will drop by a factor of 5 every 10 years. DES will only become less secure as time goes on. As a result NIST, National Institute of Standards and Technology, formerly the NBS has recently developed a new standard, Advanced Encryption Standard (AES) [15], which superseded DES as the government standard for secret-key encryption of unclassified data.

The issue of key length for public key encryption algorithms and digital signature schemes are more complicated. Since the security of these algorithms depend on the difficulty of a certain mathematical problem (e.g., integer factorization for RSA [16], discrete logarithms for DSA [8]) the efficiency of the algorithms solving these mathematical problems directly affects the borderline of the key length to have the same level of security as secret key equivalents. For instance for RSA and DSA, 1024-bit key length is recommended today to have the same security level of a 80-bit keyed secret-key algorithm [1].

Lastly, to have the same level of security in a MAC algorithm one needs to choose the same key length of secret-key algorithms. However one-way hash functions should have a length equal to twice the key length.

II.1.2 Public-key Encryption

To be able to send secret messages to people you have not met yet, and with whom you have not agreed on a secret key, public-key (asymmetric) encryption schemes (e.g., RSA [16]) have been introduced where there are two keys instead of a single key that Alice and Bob share. One key is for encryption, the other is for decryption and it is not possible to compute one key from the other. Now Bob can generate a pair of these keys and publishes the one for encryption. Alice can take the encryption (public) key and encrypt a message to Bob. Bob can use his (private) decryption key to decrypt and read Alice's message.

The key point here is that key distribution is much easier with public key cryptography. Each user is responsible for knowing his own private key, and all the public keys can be accessible in one place. If a new user joins, it is sufficient to generate a private key and make the corresponding public key accessible (See subsection II.2.2).

II.1.3 Message Authentication Codes

One of the biggest fallacies in applying cryptography is the thought that encryption provides data integrity. This belief is not correct since for instance for some ciphers in some modes of their operations even some attacks that can modify the data without turning it into garbage are practically possible (e.g., cut and paste attacks).

Message authentication codes (MACs) do not provide secrecy but ensure message authentication and integrity. MACs use a shared secret key, just like symmetric encryption algorithms. When Alice wants to send a message to Bob, she computes the MAC of the message using the secret key and appends it to the message. Every message has a unique MAC for each possible key. When Bob receives the message, he computes its MAC again using the same shared secret key and compares it with the MAC he received from Alice. If they match, then he knows two things: The message really does come from Alice and it is complete and unaltered.

II.1.4 Random Number Generators

"Random number generators are least-talked-about cryptographic tool, but are no less important than the others" [1]. Almost every security protocol that uses cryptography requires random numbers (for keys, seeds, unique values etc.) and the security of these protocols depends on the randomness of those random numbers.

If the random number generator is insecure (i.e., does not generate good-enough random numbers), it is not difficult to break the protocol no matter how carefully it is designed.

II.1.5 One-way hash functions

“One-way functions” are functions that are relatively easy to compute but significantly harder to reverse. That is, given x it is easy to compute $f(x)$, but given $f(x)$ it is hard to find $x^{'} \neq x$ such that $f(x) = f(x^{'})$. Having been used for computer science for a long time, “hash functions” take a variable-length input and convert it to a fixed-length generally smaller output. And finally, one-way hash functions are hash functions that works in one direction or in another view, they are like digital fingerprints: small pieces of data that can serve to identify much larger digital objects (e.g., MD5 [17], SHS [9]). They are public functions; no secret keys are involved. The security is in their one-wayness. These functions have an enormous range of applications in security as one of them is in digital signature schemes as seen in Figure II.1 and we will see some one other in section II.3.

II.1.6 Digital Signature Schemes

Like public-key encryption, digital signatures use a pair of keys but this time we are going to reverse the order of operations applied as seen in Figure II.1 (On the lefthand side the signing is performed by Bob and the private key used is Bob’s whereas in public key encryption when Bob wants to send an encrypted message to Alice, he uses Alice’s public key).

In practical implementations, algorithms are often too inefficient to sign messages. To save time, digital signatures are usually implemented with hashing algorithms, which convert the message into a fixed length smaller output. This

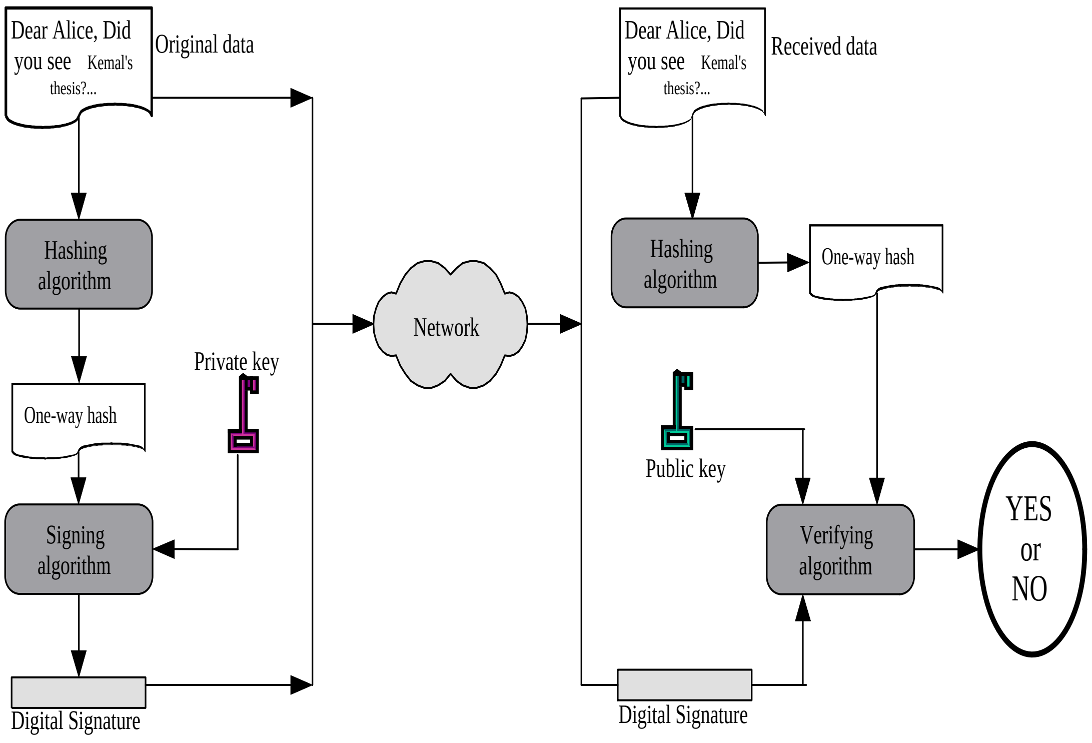

Figure II.1: Digital Signatures with Public Key Cryptography

one-way hash value not the message itself is signed and verified.

Digital signature algorithms have nothing to do with encryption and their goal is not confidentiality as explained in section I.3. A common mistake is to call the public key and private key operations in Figure II.1 as encryption and/or decryption. As a matter of fact some people get confused and can not distinguish signing and encrypting (or verifying and decrypting) simply because one of the popular digital signature algorithm (RSA [16]) can be both used for digital signatures and public key encryption. However in general this is not true for other algorithms.

II.2 Public Key Certificates and Public Key Infrastructure

II.2.1 The Need for Public Key Certificates

With respect to digital signatures and signature based authentication protocols, the use of public key cryptosystems raises the following crucial problem.

The public key of the signer is needed for the verification process. By the use of the public key it can be justified whether the signature was generated by the corresponding private key or not. However, it is not provable whether or not a certain entity owns the public key.

Obviously, the authentic link between the public key and its owner is needed [1]. Such a link can be provided by "public key certificates" which are signed mes-

[1] This link is necessary for public key encryption as well in order to use the correct encryption (public) key.

sages specifying an identity and the corresponding public key. Since each public key corresponds to a particular private key, a binding of the private key to its owner is given indirectly. If the public key is not bound to its legal owner, attacks like the following become possible:

An adversary X generates his own key pair $(PUB_x, PRIV_x)$ and publishes the public key PUB_x by claiming a wrong identity A. Afterwards, X is able to forge signatures of an entity A. This is due to the fact that the verifier assumes that the public key used for the verification process, belongs to its legal owner, namely entity A.

Public Key Infrastructure (PKI) provides protocols, services, and standards for public-key cryptography in order to employ the public key certificates to securely and effectively use digital signatures and public key encryption.

The fundamental problem PKI tries to solve is to distribute public key certificates securely. More liberally, there are various ways for secure distribution either employing PKI or not; we will summarize the most popular ones as follows:

1. **Out-of-band Distribution**: Before sending someone a digitally signed document, we can pass on the public key offline (by out-of-band channels). More specifically, we can distribute the public key over an insecure channel and only the hash of the public key over a secure channel, such as a telephone.

   It is evident that this approach has some practical limitations i.e., out-of-band channels are not always available. Other than that, the third feature of digital signatures, ability to solve the nonrepudiation problem requires the public key to be certified by a third party. One also might think that in this way there is no advantage of public key cryptography over secret key cryptography where the secret key needs to be distributed in a similar way. However with out-of-band distribution using public keys has still some advantages over using a secret key because for instance you no longer worry about the risk of eavesdropped (public) key which you distribute to other parties.

2. **Public Key Infrastructure**: In a PKI, a trusted third party usually called the certificate authority (CA) generates public key certificates which are signed messages specifying a name and the corresponding public key. Now all users are required to get securely the CA's public key so that they can verify the signature on certificates. Then, you might ask: "what is the deal? If I can get securely a public key why am I not get the public key of the user himself in the first place?" The answer is that obtaining CA's public key has an advantage since it is the only one required to verify signatures generated by a number of users.

   In the most basic form of this model, all users in the world choose a single CA. The public key of that CA is embedded in all software and hardware [10].

   In a more practical PKI trust model, instead of preconfigured with a single key, the products (e.g., web browsers) come configured with many CAs and a certificate issued by anyone of them is accepted.

There is also the delegated CA model where a top level CA issues certificates to other lower level CAs. Users can then obtain certificates from one of these delegated CAs. To verify a certificate, Alice processes a chain of certificates from the top-most level (root) CA to Bob's name.

3. **Web of Trust Model**: This model does not require a CA and trust is established at a user level as opposed to higher-level CAs. This model has been popularized by the encryption software PGP (pretty good privacy) [18]. In the PGP model, the user initially trusts some set of other users. Therefore, he will trust the public keys associated with these users. From the initial set of public keys that the user stores in a key file, "transitive trust" may be used to trust other public keys. For example, if you trust Alice in your key file and Alice trusts Bob, you can choose to trust Bob and add Bob's public key into your key file.

II.2.2 The Components of a Public Key Infrastructure

As we have mentioned, CAs are the well-recognized entities to serve as trusted third parties to produce an authentic link between an individual's identity and his public key. However on the behind scene, a PKI involves a collaborative process between several entities: the CA, a registration authority, a certificate directory and key recovery server [13]. In this subsection we discuss each of these components in short.

- **Certificate Authority**: If we think of a certificate as being similar to a driver's licence, CA operates as a kind of licencing bureau analogous to Directorate of Security. The CA is ultimately responsible for the authenticity of its end users. CA's fall into two categories: private and public. Private CAs are usually for closed networks on the other hand public CAs operate via Internet, providing services to the general public.

- **Registration Authority**: A registration authority (RA) can serve as an intermediate entity between a CA and its end users, assisting the CA in its day-to-day functions such as accepting and verifying registration information about new registers, generating keys on behalf of end users, distribute hardware devices such as smartcards etc. In general CAs can delegate their authority to accept registration information to a local RA (not a local research assistant!) when it becomes geographically dispersed. However RAs are different than delegated CAs discussed previously because the end user sees a single certificate instead of a chain of certificates when RAs are employed

- **Certificate Directory**: Certificates must be stored for later use after they are generated. End users can store the certificates on local machines but CAs often utilize a certificate directory as a central storage location. X.500 standard and LDAP (lightweight directory access protocol) becomes widely accepted in implementations [13]. Note that because of self-verifying nature of certificates, these directories themselves do not necessarily have to be trusted.

- **Key Recovery Server**: As the name implies, the key recovery server gives the end users the oppurtunity to recover their lost private keys. The loss might result from hardware failure, forgotten password etc.

II.3 Hash Chains

The idea of "hash chain" was first proposed by Lamport [19] in 1981 and suggested to be used for safeguarding against password eavesdropping. However being an elegant and versatile low-cost technique, the hash chain construction finds alot of other applications ranging from micropayments [20] to server assisted signature protocols [21]. Despite the fact that the concept is not difficult to grasp and can be explained in a few sentences, it is better to present it in a separate section due to its importance and widespread use.

A hash chain of length N is constructed by applying a one-way hash function $h()$ recursively to an initial seed value (s).

$$K^N = h^N(s) = \underbrace{h(h(h(...h(s)...)))}_{N\,times}$$

The last element K^N resembles the public key in public key cryptography (i.e., by knowing K^N, K^{N-1} can not be generated by those who does not know the value s). This property of hash chains has been directly evolved from the property of one-way hash functions.

In most of the hash-chain applications, first K^N is securely distributed and then the elements of the hash chain is spent one by one by starting from K^{N-1} and continuing until the value of s is reached (at this point the hash chain has been exhausted and a new hash chain needs to be generated to proceed). For instance in the micropayment scheme, K^{N-i} corresponds to the i^{th} payment (e.g., worth of one cent) from the user to the vendor.

II.4 Formal Methods

So far, we have discussed the basic cryptographic tools and some supporting technologies. At the end of this chapter we would like to introduce another important tool that is widely used in designing/analyzing authentication protocols.

Authentication is not an easy problem. Needless to say, using secure cryptographic tools is not sufficient to design a secure authentication protocol. In the security literature many protocols have been published and at a later time weaknesses and flaws were discovered in them. This leads security scientists to establish better guidelines for protocol design and analysis.

The most popular approach to analyze authentication protocols is through the use of logic. The first and the most popular one is the BAN logic, named after its inventors Mike Burrows, Martin Abadi, and Roger Needham [22]. The BAN logic provides a formal method of reasoning about the beliefs of participants in a security protocol. It reasons about what is reasonable for a participant to believe, given sight of certain messages.

BAN logic has been used succesfully to find the flaws in some security protocols e.g., CCITT X.509 protocol [23]. On the other hand it has also its own limitations. According to Meadows [24], "it is unlikely that any formal method will be able to model all aspects of a protocol, and thus it is unlikely that any formal method will be able to detect or prevent all types of protocol flaws". We do not want to go into the details but refer to some references for these limitations [6] [24].

What we would like to do here instead is to give a short background information on the BAN logic. For a more comprehensive treatment we recommend the original papers [22, 25].

The key idea in BAN logic is that we will believe that the identity of the claimant is as declared when a received message is fresh and encrypted with a relevant key. Notice that this belief is only true when the underlying cryptographic algorithms are secure. It also demonstrates the difference between the motivations of protocol analyzer and crypto-analyst.

The constructs of the logic are expressed using the following notations:

$A| \equiv X$: A is entitled to believe X.

$A| \sim X$: A once said X.

$A| \Rightarrow X$: A has jurisdiction over X (A is authority on X and is to be trusted).

$A \triangleleft X$: A sees X.

$\sharp X$: X is fresh.

X_K: X is encrypted using the key K.

$\rightarrow^K A$: K is A's public key (K^{-1} is only known by A).

These constructs can be manipulated by using the following postulates:

Message-meaning rule (for public key cryptography): States that if

A sees a message encrypted under K^{-1} and K is the public key of B, then A will believe that the message was once said by B.

$$\frac{A \triangleleft X_{K^{-1}}, A| \equiv \rightarrow^{K} B}{A| \equiv B| \sim X}$$

Nonce-verification rule: States that if a participant once said a message and the message is fresh, then that participant still believes it.

$$\frac{A| \equiv \sharp X, A| \equiv B| \sim X}{A| \equiv B| \equiv X}$$

Jurisdiction rule: States that if a participant believes something and is an authority on that matter, then he or she should be believed.

$$\frac{A| \equiv B| \Rightarrow X, A| \equiv B| \equiv X}{A| \equiv X}$$

Note that the statements on the top are the conditions and the one on the bottom is the result. There are a number of further postulates but the most important ones given here will be sufficient to make the analysis of the protocols proposed in the following two chapters.

Finally, using the BAN logic, the analysis of protocols will be performed in three steps:

- The idealized protocol is derived from the original one.
- The goals of the authentication protocol are formalized and assumptions about the initial state are written.
- The postulates of the logic are applied to discover the beliefs held by the participants.

CHAPTER III

ENTITY AUTHENTICATION AND ONE-TIME PASSWORDS

A donkey with a golden saddle still remains a donkey.

- Turkish Proverb

Why do you security people always speak of compromise as if it's a bad thing. Good engineering is all about compromise.

- overheard at a project review [10]

Just after the introductory and background information provided in the first two chapters, our detailed discussion on the "efficiency" issue starts with this chapter where we argue that one-time passwords can be an efficient alternative for entity authentication when looser security requirements are allowable. In addition, our new protocol called SCOTP which improves the security and flexibility (not the efficiency) of previous one-time password protocols is proposed here.

As we have discussed in section I.2, entity authentication [1] is the process by which a system can determine whether or not a given user is the one who he claims to be. When the authentication mechanism is compromised, the rest of the security measures are bypassed as well therefore one can hardly imagine any security without having a secure means of establishing authentication.

It is widely accepted that authentication uses either one of the following (or a combination).

- Something you have (smart cards, smart tokens etc.)
- Something you are (biometrics)
- Something you know (passwords)

The last alternative, the passwords are the most widely used method for authentication. This is due to its low cost and convenient usage. While the others depend on specialized hardware devices, this method can be fully utilized by using only software techniques. This is why sometimes it is called "software authentication". Other than inconvenience and cost, the first two alternatives have their own problems. We do not explain these but refer instead to some references [1, 6, 26].

Authentication by using only passwords is a well-known problem in security area and a large number of papers have been published trying to solve the weaknesses of the previous ones. Due to the importance of the problem, section III.1 is reserved for the introductory information and literature survey about

[1] In the rest of this chapter we will use "authentication" in short to refer to entity authentication.

software authentication techniques and their weaknesses. Next we introduce one-time password (OTP) schemes, which offer a viable alternative but have some deficiencies with respect to flexibility and security they provide. In section III.2 we also explain how and when one-time password protocols can provide the efficiency we are looking for.

Before going into the details of our proposal, we briefly explain our contribution in section III.3. We will answer the question of how not to store a secret on the server side in one-time password (OTP) schemes in section III.4. In Section III.5, to overcome the limitations of hash chains, we will propose the concept of Signature Chain. In Section III.6, by designing a new protocol we will show how this new idea can be employed in OTP operation. Section III.7 will analyze SCOTP, our new authentication protocol that uses Signature Chain. In Section III.8 possible solutions to establish a survivable authentication framework in e-health systems including the one based on one-time passwords are discussed. Section III.9 provides the security analysis of one-time password protocols (including SCOTP). We will look at some practical issues in Section III.10. Section III.11 concludes the chapter with the summary and possible future works.

III.1 Overview of Software Authentication Techniques and Their Vulnerabilities

The classical way of authentication via passwords in early protocols like telnet is composed of 4 steps:

1. User enters the name and the password.

2. The client machine sends the name and the password across the network.

3. Server uses the password to authenticate user's identity.

4. Server authorizes access for authenticated identity.

Since in these primitive protocols, the password is transmitted across the network in plaintext (without any encryption), anybody that can intercept the password can use it later for impersonation. This simple attack was the initial motivation behind the design of one-time passwords (OTPs). Nowadays, for the sake of security, system administrators are switching their computer systems from telnet to secure shell (SSH) [27]. In SSH where the password is transmitted in an encrypted channel across the network, "eavesdrop and replay" kinds of attack are impossible and therefore at first glance one might think that having the elegant solution of SSH, using OTPs becomes obsolete and is not more than an inconvenience for the user. We will show why this is not the case but first let us finish our discussion of authentication methods in general.

We first try to categorize the hacking attacks trying to break into the authentication method in use. There are various ways to hack an authentication system. Figure III.1 shows the classification of attacks depending on the place that is susceptible of:

- **Network Attacks**: We have already mentioned one simple example of passive network attacks in protocols like telnet. In systems where the password is transmitted in an encrypted channel, this attack is still possible if the encrypted channel has not been designed carefully. Active network

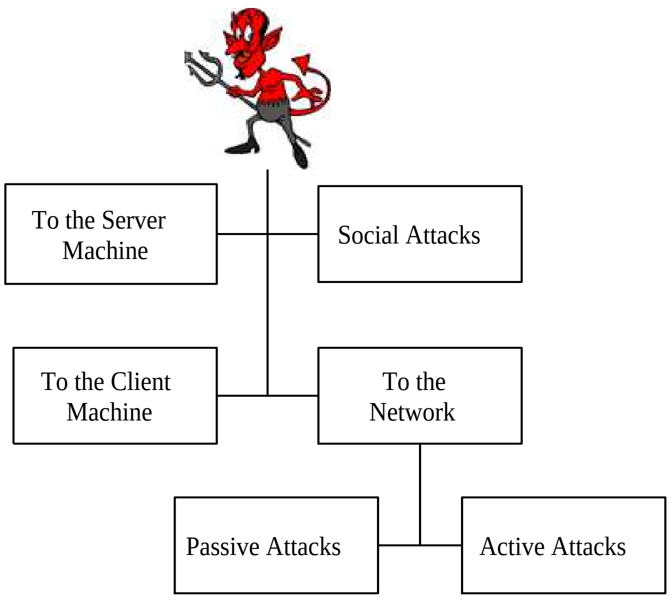

Figure III.1: Classification of Hacking Attacks

attacks are more sophisticated attacks in which the attacker does not only listen to the network but also he has the capability to delete, change or insert the authentication packets in real time. Examples are "man-in-the-middle" attacks, "hijacking connection" attacks etc. (Active attacks are mostly beyond the capabilities of most attackers and one-time passwords are not sufficient for these active attacks, one needs to use more advanced methods like SSH to safeguard against.)

- **Social attacks**: Maybe the easiest of all four types is the social attack if you can somehow persuade the untrained user to disseminate his password e.g., by introducing yourself as the system admin over the phone. This type of attacks does not only consist of this simple case only but includes more intelligent and sophisticated techniques e.g., Lally and Hardy conducted a survey to get an idea of password reuse and the results show that passwords are reused often [2]. In this survey, a fair amount of people reused a ”real” password as the ”survey” password. Not being difficult to set up a malicious web site, it is very easy to learn one’s password if he reuses it.

The applicability of the remaining two attacks depends on the authentication method in use therefore we would like to first partition software authentication methods into two major groups:

- Authentication methods based on shared secret between the server and the user.
- Authentication methods, which do not need shared secrets.

In the first group, the server and the user shares a secret and that secret is mostly called the password. The secret is unknown to anybody else and the user needs to prove that he knows the secret in order to be authenticated. The procedure how the user proves that he knows the secret differentiates with respect to method in use. In telnet and SSH’s current password authentication method [3] for instance, the user proves by simply sending it (across the open or encrypted channel). Today, more sophisticated methods are available in which the user does not need to send the secret but in a number of rounds both sides exchange some messages serving the purpose of proving that both sides know the

[2] M. Lally and S. Hardy, “Pitfalls of Password Reusage”, Available at http://www.ece.wpi.edu/∼sunar/ee579r/password.ps, Last access: September 17, 2003.

[3] In SSH after the encrypted tunnel is set up, it is possible to use either the mostly preferred method; password authentication or another widely-used one; public-key authentication that will be explained shortly [27].

secret without revealing it. These methods are usually called ”zero-knowledge proofs” [3, 11]. We do not go into the details of these more advanced methods but as stated next, in all of these methods there are two properties in common and two corresponding attacks are possible:

- The user should enter the same password in each authentication.
- The server requires to store the secret information in a protected file to authenticate its users.

Let’s turn back to our discussion on the types of attack:

- **Attacks to the Client Machine**: There are several ways in which your password may be snooped directly on the client machine e.g., someone with root access may maliciously have installed a ”wiretap” device driver in the kernel, or a trojan horse version of an application program. If a system administrator installing the software is not malicious but careless enough not to check that he has an unmodified version of software distribution, a “keyboard-trapping” routine inside the modified version of authentication software can again capture the password when you are typing and forward it to the attacker’s machine. Since the password used in each authentication is the same one, the password stolen can easily be used later for impersonation.
- **Attacks to the Server Machine**: The second property above, storing the password in a file causes another serious vulnerability. The server can

leak that secret to third parties accidentally or maliciously. For instance if the server is also a web server, the attacker can utilize the CGI vulnerabilities to steal the password file. The passwords are not listed in plaintext in the password file where password is hashed and the hash value is stored but since it is not practical for a user to choose a difficult-to-guess (high-entropy) password, (off-line) dictionary attacks are generally powerful to compute the password from its hash value. So this method is practically same as storing the password.

In the authentication methods, that do not need to share a secret with the server, the user has two options:

- In protocols like Kerberos [28], he shares a secret with a trusted third party (TTP).
- By using public key algorithms, which is based on two keys named public key and private key, the user does not need to share a secret with any other party. The server does only know the public key of the user, which is available to anybody. The private key should be kept secret and known only to its owner. An algorithm should be in operation to allow the user to prove that he knows the private key corresponding to the public key.

In both of these, the server does not need to store any secret information eliminating any attacks on the server side. For the sake of brevity, we will not explain here how the protocol Kerberos works, but for our purposes note that offline dictionary attacks are still possible, now the TTP is vulnerable instead of the server. At this point to keep our word we will explain in a simplified fashion how traditional public-key authentication process works in protocols like SSH. The simplified protocol involves three steps:

1. The server sends a randomly generated number called "challenge" to the user.

2. The user digitally signs the challenge by his private key and sends back the digital signature.

3. The server uses the user's public key in order to verify the signature of the challenge. If the signature is good, then authentication is done. Otherwise, authentication fails.

This method has another advantage that is the password is never transmitted over the network thereby eliminates attacks on the encrypted channel to intercept the password.

In software authentication, generally user's private key is stored on the client machine protected with a password (this is why this method can also be studied under the umbrella term "authentication via passwords" as far as software authentication is concerned). The user again needs to enter his password to the client machine to sign the challenge. To safeguard against attacks in stealing the private key from the client machine one can argue that the user would type his private key instead of the password to sign the challenge however the attack we have mentioned on the client machine with a wiretap device or a trojan horse program is still possible in this case. Only the difference is that the attacker now steals the private key instead of the password.

III.2 One-Time Password Schemes

In summary, so far we have shown that none of the software authentication methods we have seen safeguards against the attacks on insecure client machines where it is always possible someone to steal the password when the user types in. There are two possible countermeasures for this attack:

- Making sure that the client machine is secure so that it does not allow someone to snoop the secret when the user types in.
- We can employ one-time passwords for authentication purposes. Since each password we use is valid for only one time, the password that is snooped is not useful afterwards.

The first option is acceptable when the user is an expert with the full administrative rights of the client machine used. This is not realistic however in most cases where for instance the user travels frequently and should use insecure, untrusted and even hostile machines for reaching his home machine.

One-time password (OTP) schemes, where each password is used only once offers a viable alternative or a supplement to traditional password schemes. OTP schemes' advantages depend on which mode of operation is used. Two modes of operation for OTPs are possible and their corresponding advantages are as follows [10]:

- **Workstation environment**: In this first mode of operation, to facilitate user-friendliness, each user has only (and need to memorize) one password just like traditional password schemes. This password is used to authenticate the user to the client machine and this machine generates the one-time password to be authenticated by the server. On the network between the client and server machine only the one-time password is transmitted and hence safeguards only from "eavesdrop and replay" kinds of attack.
- **Human and paper environment**: For the applications that require stronger security, it is possible to have the user enter the one-time passwords without getting any help from the system. In this case not only "eavesdrop and replay" attacks are impossible but also the other problems of traditional passwords are avoided.

If one-time passwords are used in the first mode of operation, just after user is authenticated by entering his password to the client machine, the client machine transmits already prepared/calculated one-time password through the network hence no computation needs to be carried out in the client machine. For the sake of reducing number of rounds, one-time passwords can be implemented in a way where the user's name and one-time password together are transmitted to the server. This results a protocol running in one-round only. As a result, one-time passwords are the most real-time efficient (for the client) and communication-efficient protocol that can ever be designed (you can not do better than a one round protocol that requires no computation on the client side). In addition it is computational efficient with respect to server's computation as well i.e., only one hash operation or public key operation (if our new protocol is preferred) is required.

On the other hand SSH protocol works at least in three rounds:

- Client encrypts a random number (session key) with Server's public key.
- Server decrypts session key, sends an encrypted confirmation to Client.
- Client is authenticated to Server using the encrypted tunnel e.g., by password authentication (safeguard against both active and passive attacks).

Notice that in the first round above the client should perform public key operation (a heavy operation especially when constraint devices are of concern).

After this discussion it becomes clear that one-time passwords operating in workstation environment are much more computational and communication efficient with respect to SSH (and other public-key protocols including more advanced ones based on zero-knowledge proofs). However they are not good for active network attacks. Our conclusion is that if active network attacks are not of concern but "eavesdrop and replay" kinds of attacks should be avoided then OTP protocols has a tremendous efficiency advantage over other alternatives.

Let's turn back to the human and paper environment. In this mode, of course we cannot expect someone memorizes all these one-time passwords. In a practical situation, mostly these passwords are written down on a piece of paper and carried in the pocket (alternatively, mobile devices such as PDA and cellular phone can be used for storing the OTP list). This is somehow an inconvenience for the user but in most situations demanding high level of security, that inconvenience is worth. Note also that for secure authentication, the user should keep the OTP list secure.

Variations of OTP schemes [3] include “sequentially updated OTPs” where there is initially only a single secret password that is shared and the user creates and transmits the new password while he is being authenticated with the previous one and “shared lists of OTPs” where the user and the system use a set of secret passwords (each valid for a single authentication and distributed as a pre-shared list).

As we discussed in section II.3, the idea of hash chain was first proposed by Lamport in 1981 [19]. "OTP sequences based on hash chain” is a more elegant design and has more attractive properties than the other two variations. First of all, this method is more efficient with respect to bandwidth than sequentially updated OTPs. Sequentially updating also becomes difficult when communication failures occur. Second, shared lists have the drawback of maintaining and distribution of the list.

OTP schemes based on hash chain (Lamport’s idea) operates in three steps [29]:

1. **Preparatory Step:**

   The server and the user agree on a shared secret (the password). The server sends a seed in clear text to the user. The user concatenates the password with the seed so that the user can use the password more than once by changing the seed. The result of the concatenation is called s on the generation and verification steps.

2. **Generation Step:**

   By applying the one-way hash function sequentially to s, a sequence of one-time passwords (p_i) is produced. The initial one-time password is produced by applying the one-way hash function $h()$ N times:

$$P_0 = h^N(s) \tag{III.1}$$

   The next one-time password is generated by applying the one-way hash function $N-1$ times.

$$P_1 = h^{N-1}(s) \tag{III.2}$$

   And, the general formula is:

$$P_i = h^{N-i}(s) \tag{III.3}$$

   By knowing P_i, P_{i+1} cannot be generated because hash function is a one-way function.

3. **Verification Step:**

   First of all the initial one-time password P_0 is calculated by the host. Before a user tries to be authenticated, the current value of i and the *seed* are passed to him, so that the user enters the next one-time password (the first password to be used for authentication would be P_1). The server applies the one-way hash function to it:

$$P_i = h(h^{N-i-1}(s)) = h(P_{i+1}) \tag{III.4}$$

If it is the same as the one stored on the server side, the user is authenticated and the one-time password is saved for the next authentication.

III.3 Our Contribution in a Nutshell

Lamport's idea of hash chains has been widely employed in popular OTP software packages [4] [29, 30]. But we are not aware of any further study to improve his idea. One exception is [31] where the authors extend Lamport's idea to more general access mechanisms by combining it with zero-knowledge techniques.

More recent studies [5] [32] recommend other OTP schemes (not based on hash chains) to overcome the limitations of Lamport's idea but as discussed in the previous section, they have their own limitations. In [32], the author proposes a new OTP scheme with a shared list of passwords predistributed. Their contribution is to allow the server decreases the storage requirements for this list by cryptographic techniques. However, secure distribution of this long list is problematic in most environments and the encryption of the shared secret by a master key does little on protecting the secret in the server where the master key is the new secret and can be compromised. The proposition in [**?**] is also a shared list approach and will be discussed more in the next section. This

[4] "OPIE, One-time Password In Everything", Available at http://inner.net/opie/, Last access: September 17, 2003.

[5] M. Kuhn, "OTPW - A One Time Password Login Capability", Available at http://www.cl.cam.ac.uk/~mgk25/otpw.html, Last access: September 17, 2003, Last update: September 1, 2003.

study [33, 34] is the first one trying to improve the security and flexibility of Lamport's idea.

Currently in all of the available one-time password software systems, the server stores a secret. This makes the server side vulnerable to attacks i.e., steal the password file and make a dictionary attack (there are various tools available in the Internet to easily perform these attacks [6] [7]). Our first contribution is the design of a one-time password based authentication method, which does not need to store a secret on the server side while still has the capability of defending against client-side attacks.

Our second contribution is a chain-based scheme in which from any incorrectly revealed one-time password, unspent passwords cannot be calculated therefore for secure operation our scheme does not assume the user is careful enough to enter the correct password in each authentication. (it is obvious that without chaining, with a cost of increased storage, this is easily supported in a shared list of independent passwords). Our last contribution at this subject is to allow using one-time passwords without a need to reinitialize the system after a certain number of authentication. As having an infinite length, in contrast to current method in use, the chain in our proposition is more flexible and facilitates using the protocol without the complexity and communication overhead of restarting.

[6] John the ripper - password cracker homepage, Available at http://www.openwall.com/john/, Last access: September 17, 2003.

[7] S/KEY patch for the "John the ripper" program, Available at http://www.monkey.org/ dugsong/john-1.6.skey.patch-1/, Last access: September 17, 2003, Last update: January 26, 2000.

III.4 How not to Store a Secret in OTP Schemes

At a first glance, one might thing that there is no stored secret on the server's side in Lamport's OTP scheme in [30] i.e., the server needs to store only the last element in the hash chain at start and the previous one-time password while in operation. In the reality this is not the case; first of all, hash chains are constructed from the concatenation of a password and a seed (transmitted to the user in cleartext). The password is the shared secret and is stored on the server side in current implementations. Second, even if the password is destroyed after the hash chain is generated (any further chains can not be generated), the server is susceptible to attacks since the password might probably be a guessable one and an off-line dictionary attack to the hash chain is very powerful just like in the traditional password scheme [10, 3]. As a result, in current implementations of Lamport's OTP the system is vulnerable since practically we store a secret on the server side.

There are a couple of ways not to store a secret on the server in OTP schemes and therefore not having the problem of server side attacks like the off-line dictionary attack:

1. The user does not start the construction of hash chain from a guessable password but from a random number using a secure random number generator. He then passes securely this random number to the server. This can be accomplished offline or require the user go somewhere in person. Other than the inapplicability in some environments of this off-line process we encounter in traditional password methods as well, we have one

more problem specific to OTP methods. In traditional password methods, it is sufficient to store the hash of the secret whereas since we need to construct other hash chains when the first hash chain is totally spent, in OTP schemes the server cannot delete and need to store the secret without hashing in order to generate succeeding hash chains. The alternative way, separate random numbers passed to server offline for each hash chain will be an increased inconvenience for the user.

2. Again, the user starts the construction of hash chain from a random number using a secure random number generator and in one of the following ways he passes the last element of hash chain to the server:

   - This process can be off-line just like in the previous method. The deficiencies in this case are similar to the ones explained above.
   - The user might have already been authorized to have an access to his account (he was authenticated previously by any other means). He can store the last element of the chain in his account so that the server will be able to look it up for the later authentications by OTP. The deficiency in this case is that the assumption of having already other authentication methods available will result in storing secrets not for OTP operation but for the other methods e.g., traditional password authentication.
   - The user cannot send to the server this last element in plain (transmitted in the network). Remember our discussion in subsection II.2.1.

Similarly there would be a security problem when an authentic link is not established between the last element of the hash chain and the user himself. The solution to this problem is the well-known digital signature that is the user signs this last element so that nobody can forge it.

3. The user needs to use public key cryptography to sign. Public-key cryptography is not a silver bullet and one might be very careful for a secure implementation (The most important issue that needs great care has been discussed in section II.2 and II.3). But if public key cryptography is to be used anyhow, the third and the last alternative the "signature chain" approach which will be discussed next has much more attractive properties.

III.5 Signature Chains

In this section we propose the Signature Chain as an alternative to Lamport's hash chain. Next section will show how to employ this new idea in OTP schemes.

Our proposition uses public-key cryptosystems very similar to digital signature schemes. The only difference is that in digital signatures generally the algorithm is performed on the hashed version of the message, this is because most of the times the message is too long to be processed by the public-key algorithm. In our case we do not need to bother with this kind of pre-processing if we are careful enough in choosing the digital signature scheme to be used.

We have previously claimed that in our proposition unlike hash chains, the previous passwords cannot be generated from the (incorrectly entered) later passwords. In Lamport's hash chain, think for instance instead of the first OTP, the last OTP is entered by the careless user. Then all the previous passwords can be generated from it by applying the (public) hash function recursively. For instance the first OTP is generated by applying the hash function n times (n is the length of the chain) to the last OTP. If an appropriate digital signature scheme is used in our "signature chain" construction, then this attack will not work in our proposal. The property the digital signature scheme should not have is called "message recovery" property [3] which will be explored in the next subsection but first let's define the "signature chain".

III.5.1 Definition

We define "Signature Chain (SC)" as follows:

Definition: Signature chain is a chain where the elements in the chain are the signatures obtained by applying the signing algorithm recursively starting with an initial input message.

The notation $S^i(x)$ is used for i^{th} element in the signature chain and it is the result of applying the signing algorithm (S) i times starting with the input message x. The length of the chain can be legitimately increased infinitely thereby facilitate using it without restarting or bootstrapping.

We explain below how we can construct such a signature chain:

Construction: Let algorithm S be a signing algorithm in a signature scheme (e.g., RSA [16], DSA [8]) where d is the private key and V is the corresponding verification algorithm with the corresponding public key e. Let x and y

constitute a pair such that

$$S_d(x) = y$$

$$V_d(x) = \begin{cases} true & \text{if } y = S_d(x) \\ false & \text{if } y \neq S_d(x) \end{cases}$$

$S_d^N(x)$ denotes that we apply the signing algorithm S recursively N times to the initial input message (seed) x using the private key d. As seen below, recursive applications result in an (infinite length) signature chain originated from the initial input message x:

$$x, S_d(x), S_d^2(x), S_d^3(x), \ldots S_d^{N-1}(x), S_d^N(x), S_d^{N+1}(x) \ldots \tag{III.5}$$

III.5.2 The Message Recovery Property

In the "signature chain" construction defined above, the attacker cannot generate OTPs from incorrectly entered later OTPs only if the verification algorithm in the chosen digital signature scheme does not have the message recovery property. We can briefly summarize this property as follows:

Definition: If the message itself is not required as input to verify the digital signature or in other words if the original message can be recovered from the signature itself, then the digital signature scheme has the property called "message recovery".

Assuming that the signature is a good one, then an attacker can generate the message (previous OTP) from the signature (later OTP) by employing this property. Now, we will demonstrate how this attack can be performed on RSA, which has the property of "message recovery".

In RSA signature scheme where n and b constitute the public key and a is the private key, a signature for a message x is composed by

$$y = S_a(x) = x^a \; mod \; n \tag{III.6}$$

and the corresponding verification works as follows

$$V_b(x, y) = true \; if \; x = y^b \; mod \; n \tag{III.7}$$

As seen above, the verification computation does not involve the message x and the output of this computation is compared with the message x in the end.

Suppose we choose RSA in constructing a signature chain, then the first three OTPs can be constructed as follows by using s as the initial seed:

$$P_1 = s^a \; mod \; n, \; P_2 = P_1^a \; mod \; n, \; P_3 = P_2^a \; mod \; n$$

Suppose also the user entered P_3 incorrectly instead of P_1 for the first authentication, then an attacker can compute P_2 and P_1 by using P_3 and the public key as follows:

$$P_2 = P_3^b \; mod \; n, \; P_1 = P_2^b \; mod \; n$$

In order to safeguard against this attack, hashing the password before signing it should be performed:

$$P_1 = [h(s)]^a \; mod \; n, \; P_2 = [h(P_1)]^a \; mod \; n, \; P_3 = [h(P_2)]^a \; mod \; n$$

Then P_2 can not be generated from P_3 (The attacker can generate $h(P_2)$ but not P_2).

On the other hand, Digital Signature Standard [8] scheme, which does not have the "message recovery" can safeguard against the concerned attack without a need to bother with hashing. In Appendix A, we demonstrate how DSA can be used to construct a signature chain.

III.6 SCOTP Protocol Proposed

Having defined the signature chain above, let's see how we utilize this idea in generating one-time passwords. Suppose the public key e is transmitted to the server securely and s is the seed to be used. Then the first one-time password can be constructed by

$$S_d(s) = P_0 \tag{III.8}$$

When this password is received by the server, it can be verified by applying

$$V_e(s, P_0) = true \tag{III.9}$$

The general formula for the i^{th} one-time password is

$$P_i = S_d^i(s) \tag{III.10}$$

or

$$P_i = S_d(P_{i-1}) \tag{III.11}$$

Note that just like ordinary one-time passwords, by knowing P_{i-1}, P_i cannot be generated because d is unknown to the server.

And the verification of the i^{th} one-time password is done by applying

$$V_e(P_{i-1}, P_i) = true \tag{III.12}$$

since P_{i-1} is already received.

In this scheme, unlike Lamport's hash chain, the value s does not need to be a secret value. It should be generated randomly (for security reasons) and agreed on before the first authentication takes place. For instance the user computes the hash of his public-key or public-key certificate to generate the seed and the server repeats the same computation to have the same seed to start with. Another method is that server signs the value of s and sends it to the user initially (e.g., when the user registers to the system). We will summarize the initialization and operation of SC based OTP (SCOTP) protocol below.

SCOTP Authentication Protocol:

Initialization:

1. The user generates a public key-private key pair, registers himself to the server and sends securely his public key to the server.
2. The server and the user agree on the seed value to be used.
3. The server keeps a table for each user which stores
    - user's ID
    - user's public key

- OTP sequence number (initially zero)
- Previous OTP (initially the seed value)

4. The user generates any number of OTPs he wishes from the seed value by constructing a SC using his private key and prints out them in a piece of paper. The user can update his OTP list anytime he wishes by computing more elements in the infinite length SC e.g., when the OTPs are about to finish.

Operation:

1. The user sends his ID to the server.
2. The server finds the entry for the ID the user has entered in the authentication file and sends the current value of OTP sequence number to the user.
3. The user enters the OTP with the sequence number the server has requested.
4. The server verifies the OTP by the user's public key, if it is correct then authentication succeeds, it updates the previous OTP with the one the user has entered and also increment the OTP sequence number. If the OTP is not correct then the authentication fails, it sends a warning message to the user (possible precautions the user needs to take if he has carelessly revealed the later OTPs will be discussed in subsection III.10.2).

III.7 Features of SCOTP

In section III.2, we have shown why one-time passwords are preferable when we should use insecure client machines to be authenticated. Our proposition, OTP with SC has some other nice features which traditional one-time passwords lack. These are:

1. Since our chain is constructed by using public-key techniques only, no shared secret is needed for the correct operation just like the other authentication methods based on public keys. This important property eliminates the possibilities of attacks on the server side (it was already mentioned that the system also protects the user from the client side attacks in contrast to other software authentication methods based on public-keys).

2. By utilizing a signature scheme, which does not have the "message recovery" property, it was shown that from any incorrectly revealed one-time password, unspent one-time passwords could not be calculated therefore for secure operation our scheme does not assume the user is careful enough to enter the correct one-time password in each authentication.

3. Figure III.2 shows the comparison between OTP schemes based on hash chain and SC. In this figure, each rectangle demonstrates an OTP and the line connecting these rectangles shows that hash operation for the hash chain and signing operation for SC are applied respectively each started from the seed value. As seen, we have much more flexibility in using the SC because infinite number of authentications can take place by using a

single SC [8]. Contrarily, in Lamport's one-time password scheme, the user will be able to be authenticated by the system at most N times. For the $(N+1)^{th}$ authentication, the system should be restarted or in other words the preparatory step should be repeated (i.e., in S/Key system a special command "keyinit" should be executed [29]). If the computer used to generate the passwords was not connected to the network for security reasons, to be able to get the new seed value from the server, the connection needs to be re-established. By using SC, the user can generate OTPs in his secure (home) machine in any number he wishes and prints out in a piece of paper to be used later on to be authenticated from insecure client machines. As being paranoid, to protect the private key used in generating the OTPs, the user can disconnect the computer to be used from the network.

There is one attack left which is both effective on OTP with or without SC. This attack is called "hijacking connection" attack, which we have mentioned in section III.1. All one-time password schemes are vulnerable to connection hijacking. Once the user/service has authenticated itself, their connection can be taken over. We will say more on this issue in subsection III.10.3.

[8] For secure operation every element of the SC should have a distinct value and we assume that the length of elements of the SC is chosen to be long enough to avoid the SC to repeat itself.

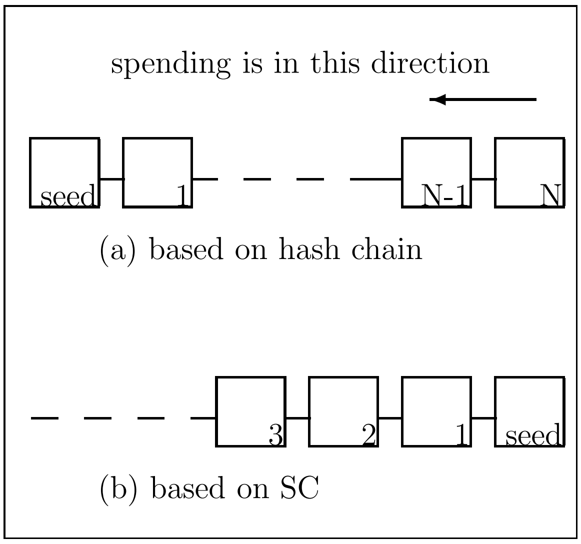

Figure III.2: Comparison of One-Time Password Schemes

III.8 Survivable Authentication for E-health Systems

Since passwords by itself does not provide the security level most systems require, NIST defines a strong authentication to be characterized by the use of at least two kinds of evidence at least one of which is resistant to replay [35]. This motivates us to use special hardware for the authentication framework for e-health systems since the only method not relying on special hardware resistant to replay is OTPs that is not so convenient to use.

At last year's WETICE workshop, Ahn and Shin [36] presented a token-based architecture for e-health systems. Their main contribution was the design of framework to make the authentication service scalable for adopting smart tokens using different technologies. However, we claim that "survivability" is more important than "scalability" in a e-health systems where there is a significant risk in locations where there is no smartcard reader available such as on an airplane or in a foreign country. For instance, suppose that a minister gets sick while he is in a formal visit to Angola, his doctor traveling with him needs to reach his patient record but there is not any smart card reader to use for authentication.

This problem is precisely the topic of this section. We have first investigated that there are several ways to overcome this limitation of smartcard-based authentication and make the system more survivable:

1. **Off-line override**: The doctor calls the system admin of patient record database and asks him for information about the minister. The main limitation of this method is its susceptibility of attacks known as social engineering (attacks). This attack on medical record privacy usually comes from a private detective who phones with a plausible tale asking about a person's record claiming that there is an emergency situation. This kind of attack is usually so successful that in both United States and Britain there are people who earn their living doing it [37].

2. **Password-only method**: Using only password authentication when there is no smart card reader has serious drawbacks. First, the system needs to distinguish when there is no smart card reader available in order to safeguard against password replay attacks. Second, there is also a risk of making smartcard-based authentication disused since ordinary users will find using only their passwords more convenient.

3. **Cryptographic calculators**: A cryptographic calculator is like a smart token that it performs cryptographic calculations using a key that it will

not disclose [10]. It is unlike a smart token in that it requires no electrical connection to the terminal. It has a display and a keyboard, and all interaction is through the user. One way it could work is by simulating a smart card; The user enters a PIN to unlock the device; the computer wishing to authenticate the user generates a random challenge and displays it to the user; the user types it into the calculator; the calculator encrypts the value and displays the result; the user enters the result on the terminal; the computer does the same calculation and compares the result.

The disadvantages of using cryptographic calculator is two-folds. First, it is economically infeasible to spend money on two separate hardware devices (smart token and calculator) for the authentication service while people find expensive only one of them. Second, carrying a second device and interact with it manually is not much more convenient than using one-time passwords.

4. **One-time passwords**: The last alternative for the supplement to have a survivable authentication system is one-time passwords (OTP) already introduced. The system then would work as follows:

   OTPs are written down on a piece of paper and carried in the pocket. In order to be authenticated using devices without a proper smart card reader, users enter the correct OTP manually. This is somehow an inconvenience for the user but that inconvenience is worth in order to be authenticated and reach critical information in most medical emergency scenarios. The inconvenience of OTPs in fact is useful to facilitate secure authentication using smart tokens in usual procedure.

In e-health systems, there is one more advantage other than the general advantages in using SCOTP protocol instead of Lamport's traditional OTP as the supplement of smart-token based authentication. That is, integrating SCOTP to the smart-card based authentication is simple and seamless. Moreover it does not require the user to communicate with the authentication server beforehand in his preparation of using OTPs.

III.9 Security Analysis of OTP Protocols

Yet one other external assumption and a limitation of BAN logic explained in section II.4 is that we assume the password is not available to anyone who might use it in an unauthorized manner. However as we have discussed, there are several ways in which your password may be snooped directly on the client machine while you are typing. Remember that no matter how sophisticated they are, none of the software authentication methods based on traditional passwords can safeguard against the attacks on insecure client machines where the only way to be authenticated securely is through the use of one-time passwords.

Our goal in this section is to analyze one-time password (OTP) protocols using BAN logic. The most popular one-time password protocol, S/KEY system [29] built on top of Lamport's hash chain idea [19] will be analyzed shortly after a required extension to the logic is proposed.

The second viable OTP approach, the signature chain based one-time password (SCOTP) protocol [34] is analyzed first.

III.9.1 Security Analysis of SCOTP

The analysis of SCOTP protocol will be performed in three steps:

- The idealized protocol is derived from the original one.
- The goals of the authentication protocol are formalized and assumptions about the initial state are written.
- The postulates of the logic are applied to discover the beliefs held by the participants.

We can express the SCOTP protocol in standard notation as follows:

Message 1: $A \rightarrow S : ID_A$

Message 2: $S \rightarrow A : SEQ_A$

Message 3: $A \rightarrow S : (R_{SEQ_A})_{K_A^{-1}}$

In BAN logic, the cleartext communication is omitted. So the protocol in idealized form can be written as:

$A \rightarrow S : (N_A)_{K_A^{-1}}$

The goal of SCOTP protocol is $S| \equiv A| \equiv N_A$ (one-way authentication only, not a mutual authentication). It is important to realize that this is a weaker goal than most other authentication protocols in which the protocol is completed when two parties agreed on the value of a secret key K.

As the assumptions, first it makes sense to accept that the server (S) has received securely (and therefore believes) the public key of A. In BAN notation, this is written as $S| \equiv \rightarrow^{K_A} A$.

Secondly we assume that the user A did not previously sign the value N_A and therefore N_A can be accepted as fresh. SCOTP protocol ensures that in the previous runs of the protocol user A did not sign the current value of N_A but it is also vital to ensure this in a situation where the same public key is reused for other purposes. We will say more on this issue in subsection III.9.3.

The final step is very straigtforward. By applying the message-meaning rule we can show $S| \equiv A| \sim N_A$. Then we can apply the nonce-verification rule and prove that $S| \equiv A| \equiv N_A$.

As stated previously, SCOTP protocol does not protect against active attacks such as "man in the middle" and "hijacking connection" attack. We observe that this is due to the weaker goal the protocol accomplishes. To safeguard against these more sophisticated attacks we need stronger achievements: e.g., S and A should believe in sharing the same secret key K so that all the communication between them can be performed in an encrypted tunnel thereby eliminates the risk of hijacked connection.

As the final note of this subsection, we agree with [34] in the recommendation of complementing SCOTP (or any other one-time password protocol) with protocols like SSH [27] to be protected against active network attacks.

III.9.2 Security Analysis of S/KEY

In S/KEY [29], a hash chain of length N is constructed by applying recursively a hash function $h()$ to a initial seed value (s).

$$K^N = h^N(s) = \underbrace{h(h(h(...h(s)...)))}_{Ntimes}$$

The last element K^N resembles the public key in public key cryptography so the message-meaning rule and the nonce-verification rule of the BAN logic can be restated as follows (note: the parameter i can take value between 1 and $N-1$):

$$\frac{A \triangleleft K^i, A| \equiv \rightarrow^{K^N} B}{A| \equiv B| \sim K^i}$$

$$\frac{A| \equiv \sharp K^i, A| \equiv B| \sim K^i}{A| \equiv B| \equiv K^i}$$

Now similar to SCOTP protocol, the authentication goal ($S| \equiv A| \equiv K^i$) can be proved by applying message-meaning rule and nonce-verification rule successively.

III.9.3 Secure Integration of SCOTP with Smartcard Authentication

Once the necessity to integrate smartcard authentication with one-time passwords is brought up (in section III.8), we should handle the issue of chosing the best method to accomplish it.

One-time password alternatives for the integration are as follows:

- use S/KEY in its original form.
- sign the last element of the hash chain in S/KEY.
- use SCOTP protocol.

The first alternative is based on a shared secret between the user and the server and has some security wekanesses such as off-line dictionary attacks. (the hash chain is derived by using the password (shared secret) as the seed value.)

Most smartcard based authentication protocols use public key cryptography where the greatest advantage is to avoid shared secrets between participants. The server uses only the user's "public" key for verification therefore the risk of compromised or hostile servers is eliminated. Since we have a public key infrastructure available used for smartcart based authentication, it is a reasonable idea to use the same infrastructure for one-time passwords as well.

While both of the last two alternatives provides a secure solution, SCOTP protocol is a more convenient one since it can generate infinite number of OTPs without requiring the user to interact with the server. As previously mentioned another advantage of SCOTP is; from any incorrectly revealed OTP, unspent OTPs cannot be calculated therefore for secure operation the protocol does not assume the user is careful enough to enter the correct OTP in each authentication.

In [38], one of the principles for public key protocols was: "avoid using the same key for two different purposes". The reason for this principle is best illustrated by an example in [6]. Let us adapt the attack in that example to our case [39].

Suppose we use the same key for smartcard authentication and for generating the signature chain. In smartcard authentication, usually the server asks the user to sign a random challenge to identify him. If an attacker asks the user to sign the OTP that was just used, then he can get a valid OTP to be useful for the next authentication (if the Lamport's hash chain is used instead and the attacker gets the user to sign the last element of the hash chain, then the situation is worse; the attacker gets a list of valid OTPs).

The easiest solution to this attack is the one which Anderson and Needham recommended [38]; use different keys for these two different tasks.

If we need to use the same key for these two tasks for some reasons (e.g., to save users having to carry two different cards or to save the server to store two public keys for the same user), then the following guidelines will help for secure operation:

1. Choose a signature scheme which does not have the message recovery property and do not apply hash algorithm before signing in constructing the signature chain. In smartcard authentication the random challenge is hashed before signing so any signature the attacker can get is not useful in the SCOTP protocol.

2. If the public key (or the certificate) is to be used as the seed value (then of course we need hashing before signing), the first valid OTP should be the second element in the signature chain because the first element can be

obtained by the attacker by getting the user sign the hash of the public key. However, the second element can not be obtained if the first guideline is conformed.

III.10 Practical Considerations

In this section, we will look at different issues that are considered to be important for putting our new authentication method in practice.

III.10.1 Signature Chain in Workstation Environment

As stated in section III.2, there are two modes of operation in OTP schemes. As opposed to human and paper environment, in workstation environment the workstation does calculate the OTP on behalf of the user after the user enters his traditional password to the system. This is a more convenient approach since the user does not need to carry OTPs and enter them manually. In this mode, other than attacks on insecure client machine, there is an attack called "small n attack" [10], where an intruder after impersonates himself as the server asks the workstation for a future OTP (with a smaller value of n) so that he can generate a list of valid OTPs by using the hash chain's one-way property. Signature chains are also effective to safeguard against this kind of attack since it was already shown that from any incorrectly revealed one-time password, unspent one-time passwords could not be calculated.

In workstation environment (as well as in human and paper environment), it is also possible to design the protocol so that the server asks for a one-time password with a randomly determined sequence number of n. The goal here is to make the attacker not guess the time correctly to use an incorrectly revealed password for impersonation. However, if this design is preferred in a workstation environment, we have now a more serious performance problem than the one that will be mentioned in subsection III.10.5. For instance if the server asks for the 100^{th} OTP and currently the 10^{th} OTP is stored on the workstation, in the previously proposed signature chain construction, the workstation needs to perform 90 public key operations in real time, which is of course not so practical. For this problem to generate OTPs, we propose the following alternative method what we call "counter method":

1. The server and the user agrees on the seed value (let's call s) just like in our original proposal.

2. However now if the server asks for the hundredth OTP, the workstation computes the signature on the value of s+100 and returns this signature as the OTP.

Notice that in this method, always only one public key operation is required. There is an interesting analogy between the two alternatives we have proposed to generate OTPs using public-key cryptography and the modes of operation in block ciphers (secret key cryptography), which specifies how a block cipher can be extended to process messages of arbitrary length [6]. While the signature chain construction we proposed is the analogue of "output feedback mode", the method we proposed in this subsection is very similar to the "counter encryption mode" in block ciphers. We defer to [6] for more information on modes of operation in block ciphers.

III.10.2 Incorrectly Revealed Passwords in SCOTP Protocol

As we have already stated, in SCOTP protocol an attacker cannot generate unspent OTPs from the incorrectly revealed one. However there is still the risk to use the revealed OTPs for impersonation if the attacker can somehow guess when the OTP at hand is good for. If the proposed signature chain construction is used, it is easy to see that an attacker has no choice other than blindly trying the OTP from time to time however in counter method we observe that the attacker can find out the sequence number of the OTP at hand. As a result it is evident that signature chain construction is more secure than counter method when incorrectly revealed passwords are concerned. To eliminate the risk of revealed OTPs totally, the SCOTP protocol can be extended so that the user can choose to enter more than one password at any time. Now the user who is aware of revealing can enter the incorrectly revealed password just after the previous OTP in a single authentication process so that the OTP at attacker's hand is useless forever (the next OTP after the revealed one in the list is required for the next authentication).

III.10.3 Hybrid Usage

It is possible to combine two authentication methods, SSH [27] and OTP to benefit the strengths of both. Session hijacking is one of the active network attacks in which the one-time passwords are not sufficient for. So to protect the user from these kinds of attack, SSH can be used with OTP. Another reason why one chooses to use such a combination might be the need of confidentiality of data exchanged. Since it is possible to secure the machine we have the root privileges and routine checks for malicious software are performed, then we may choose not to use OTPs in this machine (only use SSH). So the inconvenience in using the OTPs is left to other machines which we do not trust.

III.10.4 Length of One-Time Passwords in SCOTP

With the present state of the art in cryptanalysis, a security level of 2^{80} is recommended (an attacker should perform 2^{80} operations to break the cryptosystem). There is a well-known attack called "birthday attack" on hash algorithms which finds two inputs producing the same output in $n^{1/2}$ operations where n is the length of output. While this attack is of concern when we use the one-way hash function before signing, it is of no value in hash chain constructions. So to have a security level of 2^{80}, one can employ a secure hash algorithm like SHS [9] producing 160 bits output and folding the output of SHS with exclusive-or to produce a 80 bit output (like in [30] where they use MD4 [40] as the hash function which is considered to be insecure today). Therefore the OTPs using the hash chain construction has a length of 80 bits.

As illustrated in Appendix A, in SC construction the signatures in DSA scheme (therefore the OTPs) are 320 bits in length. We cannot decrease this length similarly by folding since the server requires the signature in plain in order to verify it. Of course, this increase in the size of OTPs is an inconvenience for the users when they need to manually enter the OTPs. Fortunately we can utilize the new results of research on decreasing the length of signatures while offering the same level of security. In [41], the authors proposed a new digital signature QUARTZ, which has a length of 128 bits offering a security level of 2^{80}. The progress in cryptography research is fascinating, recently, the shortest signature ever known was provided by [42] where the signatures are only 81 bits in length and they offer a security level of 2^{80}. So only a small increase in the size of OTPs is possible if we use this new short signature scheme in constructing the SC. However, the difficulty to employ these new short signature schemes in SC is the increased verification time and increased public-key length.

III.10.5 Performance Evaluation

To compare the performances of signature chain and hash chain, we have conducted an experiment on a PC with an 800 MHz Pentium III and a 128 MB memory. The library used was MIRACL version 4.7 [9] and the compiler was Microsoft Visual C++ Version 6.0. SHS and DSA with a key length of 1024 bits were chosen to generate the hash chain and signature chain, respectively. Our implementation results show that using public-key algorithms instead of hash functions does result in a degradation of the performance in verification of one-time passwords (hash functions are designed to be very fast, only 0.028 msecs is sufficient to perform one hash operation). However we claim that verification

[9] Shamus Software Ltd, "MIRACL Library", Available at http://indigo.ie/~mscott/, Last access: September 17, 2003.

is sufficiently quick (around 6 ms) and does not produce a significant problem considering that the server that verifies OTPs is generally a powerful device.

III.11 Summary

In this chapter, we first show that one-time passwords as a concept can be used as an efficient entity authentication protocol when active network attacks are not of our concern.

Secondly, using public-key techniques we have provided a method called Signature Chain (SC) as an alternative to hash chain to improve the security and flexibility of one-time passwords. The main disadvantage of signature chain is the larger verification time with respect to hash chain based approaches [10].

The use of wide-ranging authentication services based on public key cryptography including the one we have presented in this chapter becomes practical if it is complemented by a trustworthy mean to manage and distribute the public keys. We believe that our method will be more useful when public key infrastructure deployment reaches its true potential. As a future work it is promising to implement a fully functional OTP scheme using the SC idea and integrate it with SSH for hybrid operation. Another future work is experimenting with different signature schemes to make the operation of our protocol more efficient.

[10] For the sake of clearness, we restate that improving the efficiency is not the objective of SCOTP protocol.

CHAPTER IV

IMPROVED SERVER ASSISTED SIGNATURES

Forged in USA.

- engraved on a screwdriver claiming to be of brand Craftsman [10]

Hand is superior to hand.

- Turkish Proverb

This is the second chapter in a series of three where we propose viable alternatives for the efficiency criteria. As a part of our discussion in top-down approach for efficiency alternatives, this chapter demonstrates how efficiency can be provided in the design phase of security engineering. More precisely, in this chapter we present our new server assisted signature protocol design which improves the delay and round efficiency of previous approaches.

In recent years, one major trend in computing has been towards an environment where computer applications will be hosted on a wide range of platforms, including many that are small, mobile and regarded today as devices having only limited computational capabilities. This "pervasive computing" vision could bring a great deal of convenience but also great deal of security risks. For instance think about the possible consequences when the integrity of the data collected from a remote health monitoring sensor is not protected.

On the other hand, in subsection II.1.6 we have seen that most current techniques for generating digital signatures are based on public key cryptography (based on complex mathematical problems such as factoring or discrete logarithms e.g., RSA [16] or DSA [8]). These traditional methods are simply untenable from a performance perspective when constrained devices are of concern (On a Pentium I - 200 MHz machine [1], using 1024-bit keys RSA takes around 59 ms to sign and 14 ms to verify). Due to this fact, research in the security community so far was mostly focused on the computational incapabilities (the real-time inefficiency) of these devices.

One traditional way of increasing the real-time efficiency of generating digital signatures is to do most of the work as background computations. These methods are useful when the signer has a very limited response time once the message is presented but he can carry out costly computations between consecutive signings. On the other hand, they suffer from two fundamental problems:

- Some mobile devices may have 8-bit microcontrollers running at very low CPU speeds, so public key cryptography at any kind may not even be an option for them.

[1] Today's high-end PDA's and palmtops have a processor speed of 200 MHz.

- Low CPU speed is not the only constraint for pervasive devices. Their batteries hold only a small, finite amount of energy [43]. It is well known that public key algorithms consume much more energy than symmetric algorithms and one-way functions [44]. Precomputations does not help when the most relevant performance figure is no longer bits per second, but bits per joule [45].

One other way to reduce the computation cost on mobile/constrained devices is to employ a verifiable and a powerful server. Getting help from a verifiable-server has an advantage over proxy-based solutions (using a fully trusted server) in open networks since as opposed to proxy-server, verifiable-server's cheating can be proven.

Asokan et al. [21] proposed an efficient verifiable server assisted signature protocol called SAS but it does not totally eliminate public key operations for the signer (the signer does not need to generate but instead verify a public key signature). In this chapter, we propose a new alternative called SAOTS (**S**erver **A**ssisted **O**ne-**T**ime **S**ignatures) protocol where, just like proxy signatures, generating a public key signature is possible without performing any public key operations at all. This feature results in both power efficiency and real-time efficiency of the proposed protocol.

Furthermore, while we are designing our new protocol, we take into account the storage constraints of pervasive devices as well i.e., no signature storing is required for the signer to prove the server's cheating. The last but not the least, executing in less number of rounds, SAOTS protocol is more communication-efficient.

The rest of this chapter is organized as follows. In the next section, related work on efficient digital signature constructions is given. We propose our new signature protocol called SAOTS in section IV.2. Section IV.3 is reserved for the security analysis of the proposed protocol. In section IV.4, we give the results of our performance evaluation study. In section IV.5, we introduce a variant of SAOTS where the length of messages exchanged are shortened. In section IV.6, we discuss the other important issue in server assisted signatures, the issue of revocation of public key certificates. The concept of attribute certificate is introduced and integration of this new type of certificates with SAOTS is discussed in section IV.7. In the last section of this chapter we conclude by summing up our work and discussing future possibilities for research.

IV.1 Related Work

IV.1.1 Efficient Public Key Signatures without Server

One possible way to construct efficient signatures would be to relax the security requirements (Remember our discussion in section I.6). If a certain amount of risk is acceptable, then one could use less-studied signature algorithms. This method might provide efficiency but if the underlying cryptographic assumptions later turn out to be invalid, these signatures become completely open to compromise.

As we have already mentioned in short, a more secure solution to speed up the operation of public key signatures is to do the most of the computations on background as precomputations. The second (on-line) phase is performed once the message to be signed is known and is supposed to be very fast so that the response time of signing using a mobile device would be in acceptable range. While some signature schemes can be naturally partitioned into these two phases e.g., DSA [8], on-line/off-line signature scheme was first introduced by Even et al. [46] to convert any signature scheme into the aforementioned two phases.

While in the original proposal [46], the length of signatures are longer since it uses one-time signatures (will be explained in the next subsection), the authors in [47] introduce an alternative scheme where the signature size does not increase that much with a trade of heavier on-line computation requirement.

The notion of on-line/off-line signatures is nice, however we now face a new problem; the batteries of mobile devices hold only a small, finite amount of energy; this fact places a bound not on the response time but on the total amount of computation the device can perform. That is why on-line/off-line signatures does not help when the more relevant performance figure is not signatures per second but signatures per joule [45].

IV.1.2 One-time Signatures

One-time signatures (OTS) provide an attractive alternative to public key based signatures. Unlike signatures based on public key cryptography, OTS is based on nothing more than a one-way function (OWF). Examples of conjectured OWFs are SHS [9] and MD5 [17] [2]. OTSs are computationally more efficient since no

[2] SHS and MD5 were originally designed as one-way hash functions but they can easily be used as one-way functions when the input message length is set to be equal to the length of output.

complex arithmetic is involved. Additionally, since OTS is based only on OWF whereas public-key signatures are based on complex mathematical problem as well as OWF, using OTSs allow us to elliminate one more point of vulnerability.

The OTS concept is very easy to grasp [48]. Broadly speaking, a message sender prepares for a digital signature by generating a random number r, which is retained as the private value. He then securely distributes the hash of r, $h(r)$, where h is a one-way function; this represents the public value and is used by receivers as the signature certificate to verify the signature. The signature is sent by distributing the value r itself. Receivers verify that this message could only be sent by the sender by applying h to r to get $h(r)$. If this matches the value for $h(r)$ in the signature certificate, then the OTS is considered to be verified, since only the sender can know r. This, in effect, allows the signing of a predictable 1-bit value. In order to sign any 1-bit value, two random numbers $(r1, r2)$ are needed; this way, both $h(r1)$ and $h(r2)$ are pre-distributed but at most one of (r1,r2) is revealed as a signature. In the original proposal [48], 160 random numbers out of 320 are revealed as the signature for 160-bit hash value of any given message.

Despite the performance advantages provided by OTSs, they have not gain much attention in security world. Other than non-technical reasons, we believe two disadvantages of OTSs were in effect.

First of all, one-time signatures are longer than traditional signatures results in more serious storage and bandwidth constraints. Recent studies succeeded in decreasing the length of one-time signatures in some extent. The authors in

[49] realized that p out of n random numbers are sufficient to sign a b-bit length message if the following inequality holds for a given n and p.

$$2^b \geq C(n,p) = \frac{n!}{p! * (n-p)!} \tag{IV.1}$$

To sign an arbitrary length message by OTS, just like the public-key based signatures, we can reduce the length of the message m by computing the hash value of the message, $h(m)$ and then sign $h(m)$. This means for instance for b = 160 (e.g., SHS), n must be at least 165 with subsets of size 75 ($p = 75$).

Having determined the values of n and p, there is only one issue left to complete the signing with OTS, that is how to map a specific message to an OTS or in more concrete terms how to choose p out of n random numbers for the message in hand. Please refer to Appendix B for the discussion of how to obtain a valid mapping for a message [49]. For our purposes it is sufficient to know that this mapping is not computationally heavy. It was also shown in the Appendix B that this costs less than one hash operation.

The extra length of one-time signatures (which was an important concern two decades ago) is negligible today owing to the high speed of modern networks hence we think the second disadvantage is a more serious one, that is one-time signatures can be used to sign only one message per one public key in its simple form. Since the public key requires to be distributed in a secure fashion which is done most typically using a public key signature, the benefit of using quick and efficient hash function is apparently lost. There is also a bunch of clever approaches to overcome this limitation. One of which we have already mentioned is on-line/off-line signatures where the public key of one-time signatures is signed by using public key techniques off-line before the message is known. When the message to be signed is in hand, there will not be any necessity to perform public key operation so that the response time (real-time efficiency) is improved.

Due to power constraints, we might want to minimize the number of public key operations no matter it is off-line or not. Then Merkle's proposal [50] can be preferred where one-time signatures can be embedded in a tree structure, allowing the cost of a single public key signature to be amortized over a multitude of OTS. The problem in this formulation is the longer lengths of signatures. Now we face a more severe storage and bandwidth requirement than one-time signatures in its simple form since the length of signatures increases as the number of signatures generated using the tree structure increase.

IV.1.3 Signatures Employing a Server

The third and the last approach to use one-time signatures more than once by using a single public key is the SAOTS protocol we propose in this thesis which is based on a third party (server). But before introducing it, in a more general view we would like to summarize the work on employing a powerful server to decrease the computation requirements for a digital signature.

Server assisted signatures can be explained in three subgroups depending on the trust relationship between the user and the server. More specifically the server employed may be

- fully trusted (proxy)
- untrusted
- verifiable

In the first category, after receiving an authenticated message from a user (A MAC algorithm which can be implemented very efficiently may be used for authentication), a more powerful proxy server on behalf of the user generates a public key digital signature for the message [51]. Notice that the user himself does not need to perform any public key operation, he just computes a MAC using secret key cryptography. The drawback here is that this simple design is only applicable when the user fully trusts the proxy server i.e., the server can generate forged signatures and that cheating cannot be proven.

As the opposite, a totally untrusted server might be utilized i.e., the server only executes computations for the user. Now the goal of securely reducing computational costs on the sender's machine becomes more difficult to accomplish and in fact most of the schemes proposed so far have been found not to be secure. One exception is the interesting approach of Jakobson and Wetzel [52]. However we see that in their approach public key operations although in reduced amount are still needed to be performed on the constrained device.

IV.1.4 Verifiable-Server Assisted Signatures

The last alternative is to employ a verifiable server (VS). A VS is the one whose cheating can be proven. This approach can be considered in somewhere between the other two since the server in this case can cheat but subsequently the user would have the ability to prove this situation to other parties (e.g., an arbiter) [3]. We see that in literature, the verifiable server is sometimes named as semi-trusted server.

The first work that aims to reduce the computational costs to generate digital signatures for low-end devices by employing a powerful VS is SAS protocol [21]. In [53], the authors extend this work by providing implementation results as well as other details of the scheme. The scheme in [54] also utilizes a semi-trusted server to generate signatures but their goal is not to minimize the computation cost on low-end machines but to provide fast revocation without losing transparency for those who verify signatures. This work also has the advantage of supporting revocation not just for signatures but for public-key encryption as well. We will explore the issue of revocation in section IV.6 and IV.7.

We now would like to provide a brief summary of SAS protocol (For a comprehensive treatment, please refer to the original papers [21, 53]):

There is an initialization phase in SAS where each user gets a certificate from an offline certification authority for K^n (the last element of a **hash chain** of length n) where

[3] In traditional methods of digital signature generation, the signer usually obtains a public key certificate from a certification authority (CA). In order to trust the legitimacy of signatures, the receiver must trust the CA's certificate-issuance procedures. For instance the CA can issue a fake certificate for a particular user and then impersonate the user by generating a forged signature. However, if some kind of contract was signed in the certification process, in dispute the signer can prove the CA's cheating by asking this contract from the CA. Notice the similarities between the trust relationship between the signer and CA in traditional methods and the signer and the server in verifiable-server assisted signature protocols.

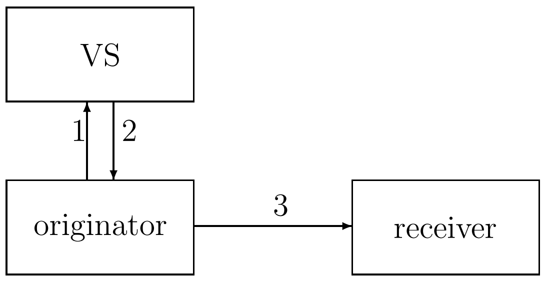

Figure IV.1: Asokan et al.'s SAS Protocol Operating in Three Rounds.

$$K^n = h^n(s) = h(K^{n-1}) \tag{IV.2}$$

In Equation IV.2, $h()$ is a one-way function like SHS [9] and $h^n(s)$ means we apply hash function $h()$ n times to an initial input s to generate a **hash chain** of length n. In addition, each user should register to a VS (which has the traditional public-key based signing capability) before operation. Then the SAS protocol works in three rounds as illustrated in Figure IV.1:

1. The originator (O) sends m and K^i to VS where
   - m is the message
   - K_i is the i^{th} element of the hash chain. The counter i is initially set to $n-1$ and decremented after each run.
2. Having received O's request, VS checks the followings:
   - Whether O's certificate is revoked or not.
   - Whether $h^{n-i}(K^i) = K^n$ or in a more efficient way $h(K^i) = K^{i+1}$ since K^{i+1} has already been received.

If these checks are OK, VS signs m concatenated with K^i and sends it back to O.

3. After receiving the signed message from VS, O verifies the VS's signature, attaches K^{i-1} to this message and sends it to the receiver R.

Upon receipt of the signed message, the receiver verifies VS's signature and checks whether $h(K^{i-1}) = K^i$.

IV.1.5 SAS Protocol Weaknesses

We have observed that SAS protocol has several drawbacks. These are:

1. **Verifying VS's signature**: In step 3 of the SAS protocol, before sending the signed message to R, O should verify the VS's signature otherwise a hostile attack cannot be noticed i.e., an attacker can change the message while in transit from O to VS and if VS signs this new message instead, O's revealing of K^{i-1} without verifying VS's signature will result a forged signature for the message the attacker has generated.

   Remember that for some pervasive devices restricted with CPU and/or battery constraints, public key cryptography is simply untenable no matter it is used for signing or verifying.

   For more powerful devices if the VS uses RSA [16] signature scheme, where verification is much more efficient with respect to signing, this might be affordable. However there are other popular digital signature schemes like DSA [8] where verification is at least as costly as signing so a protocol which offers lightweight signing without any restriction in the digital signature scheme used would be much more flexible and attractive.

2. **Network Overhead**: Remember that the main motivation to design the SAS protocol is to decrease the time required to generate a signature. One of the delay factors of the SAS protocol is the round-trip delay between O and VS. To decrease the network delay, one can try to decrease the number of rounds in SAS, however if O attaches the hash element K^{i-1} to the first message he has sent to VS, an attacker can forge a signed message easily by modifying the message while in transit.

   As a result SAS protocol cannot be a two-round protocol like the SAOTS protocol that will be introduced in the next section this is basically because the signature is not binded with the message itself in a two-round case. If the protocol is assumed to be running in a LAN environment, the network overhead is not significant and does not greatly affect the efficiency. However, today's anywhere anytime nature of pervasive computing invalidates that kind of assumptions. A protocol operating efficiently also in a WAN environment would be much more beneficial for our purposes.

3. **Storing VS's signatures**: In SAS protocol, the signer is required to store VS's signatures to prove its cheating [21]. For some pervasive devices which has a limited storage capacity, this also might put a burden on the operation.

IV.2 The Proposed SAOTS Protocol

In this section we propose the server assisted one-time signature protocol (SAOTS) which operates in two rounds as opposed to three. SAOTS is the first VS based approach where the user does not need to perform any public key operation at all [55]. Moreover in our proposed protocol unlike other alternatives the server not the user is required to save the signatures for dispute resolution. Thus SAOTS elliminates all the three aforementioned drawbacks of SAS protocol.

IV.2.1 Setup

Our protocol is built on top of one-time signature idea. As a setup, the user generates a one-time secret key (random numbers) and a one-time public key (hash value of these random numbers) and in a secure fashion he distributes the public key to the server. This can be accomplished by a public key signature if he has already a capability of traditional signing or he can directly get a certificate from a certification authority (CA) for the one-time public key he has. Similarly the server obtains a certificate from a CA for its public key.

In addition, just like the SAS protocol, each user should register to a VS before operation.

IV.2.2 Operation

The protocol works in two rounds as illustrated in Figure IV.2:

1. The user precomputes a second one-time secret key - public key pair. When the message to be signed is ready, he concatenates the message with the

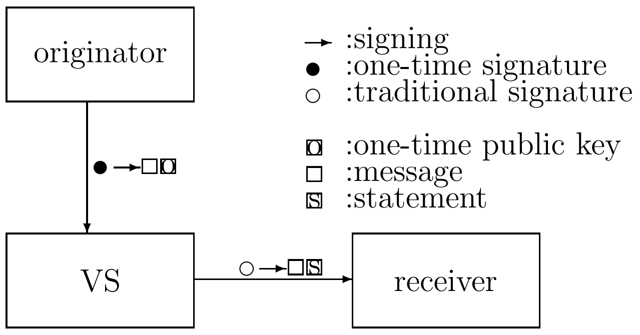

Figure IV.2: Operation of SAOTS Protocol

new public key and signs this by his previous one-time secret key. He then sends the message and the new public key as well as the one-time signature to the server.

2. Having already received securely the one-time public key of the user's signature on the message, the server verifies the one-time signature. He stores the new public key the user has signed for the verification of next message. It also signs the message with traditional public key techniques after appending a statement on the message saying that it has received it from the sender (if the sender's certificate is not revoked). The signed message is ready to be transmitted to the intended receiver(s).

The receivers can easily verify the signature by using server's public key. One can easily prove that this protocol provides all the three security services asked from a digital signature but only if the server does not cheat i.e., it does not sign any message on behalf of the user without user's approval. We will show in the next section how the user can prove the server's cheating. If he cannot prove, the other parties conclude that the user is the one who actually sends the message.

The user can sign any further messages easily by repeating the step 1. The server can always verify the one-time signature since it has securely received the public key in the previous run of the protocol. The server should store all the previous messages for the secure operation but the user does not need to store anything to prove the server's cheating. This becomes more clear when we make the security analysis in the next section.

We would like to point that the "chaining" technique we use that attaches the public key for the next message to the current message is first suggested by [56] for signing infinite length digital streams.

IV.3 Security Analysis

In this subsection, for the sake of a complete security proof, we will show

1. Underlying components (signature algorithms) are secure.
2. The authentication goal in SAOTS protocol is achieved.
3. How a dispute can be resolved.

At the end we say a few words on another issue in security; the strength of SAOTS to the denial of service attacks.

IV.3.1 Security of Underlying Components

For secure operation, we need to prove the security of signatures of both the sender and the server. Since the server's signature is a traditional one, we conclude that if the traditional signature algorithm used is a secure one, then the server's signature is also secure.

Secondly, we note that the security of the chaining technique used in the sender's signature has been studied previously. For the security proofs we defer interested readers to [56].

IV.3.2 Security Analysis of SAOTS Using BAN Logic

We can analyze the security of SAOTS protocol using the BAN logic. Let us first rewrite the messages of the protocol in idealized form:

Message 1: $O \rightarrow S : (m, Knext_O)K_O{}^{-}1$

Message 2: $S \rightarrow R : (S| \equiv O| \sim m)K_S{}^{-}1$

Note that the appended statement in the second message means that the server believes that the message is said by the originator and this has been expressed using BAN notation, above.

The authentication goal of SAOTS is $R| \equiv O| \sim m$ [4]. We can assume that the public key of sender is always believed by the server since it is received securely in the previous run of the protocol. Then by the message meaning rule we derive $S| \equiv O| \sim m$ from the first message. One other initial assumption was that the receiver has received the public key of the server securely so similarly

[4] No freshness is assured in SAOTS protocol but there is an implicit time stamping i.e., the receiver holding a signature is explicity assured that the signer's certificate was valid at the time the signature was generated. This is because of SAOTS's immediate revocation capability. For freshness assurance, it is also possible to build a time-stamping service based on VS.

we can derive $R| \equiv S| \equiv O| \sim m$.

In SAOTS protocol, the server is the authority on saying some message is said by someone. Then by the jurisdiction rule we finally derive the formula we are looking for:

$$\frac{R| \equiv S| \Rightarrow O| \sim m, R| \equiv S| \equiv O| \sim m}{R| \equiv O| \sim m}$$

IV.3.3 Dispute Resolution

Provided that the underlying signatures building the protocol are secure, we now want to show how a dispute can be resolved. In case of a dispute, the receiver can submit the message and its signature received from the server to an arbiter. The arbiter will verify the followings:

- the public key of the server is certified by the CA.
- the server's signature is valid.
- the message contains a statement saying that it is originated from the claimed sender.

If these checks are successful, then the sender is allowed to take the oppurtunity to repudiate the message. There will be two checks to decide whether the sender's claim is true or not:

- CA will be asked to prove that the sender's public key was registered by himself.

- The server will be asked to prove that the message was signed by the sender himself.

As a proof, the server shows all the signed messages received from the sender starts from the first one and continues until the message in question is reached. The arbiter verifies all these one-time signatures.

If both CA and server successfully shows that they did not cheat, the arbiter concludes that the sender is dishonest and claims falsely that he has not sent the message.

IV.3.4 Denial of Service Attacks

In previous server assisted signature protocols, unlike traditional signature schemes, denial of service (DoS) attacks aiming to deny the server's service to the users are of concern. The basic idea behind these attacks is as follows:

By sending legitimate (well-formed) requests, an adversary can force the server to perform alot of signing tasks so that it cannot response timely to the real request coming from users [53].

However if our proposal is preferred, these attacks are totally avoided because an adversary cannot forge users' one-time signatures and therefore cannot generate legitimate requests.

Table IV.1: Computational Comparison of SAS and SAOTS Protocols.

	SAS	SAOTS
Originator	$1H+1V$	$1H+1M$
Server	$2H+1S$	$(p+2)H+1M+1S$
Receiver	$1V+2H$	$1V+1H$

IV.4 Performance Evaluation

IV.4.1 Computation and Communication Comparisons of SAS and SAOTS Protocols

Rule#5: You can buy more bandwidth but not lower delays. **A.S.Tanenbaum, Computer Networks 3rd Edition, page 564.**

Table IV.1 shows the comparison of SAOTS and SAS protocols with respect to computation requirements on the participating entities.

H: hash computation

S: traditional signing by a public key

V: verification of public key signature

M: mapping computation (costs less than one hash)

p: number of hash computations to verify OTS

Note that in [49], the authors presented an efficient method which costs less than one hash operation for encoding a message for one-time signature. Encoding a message does refer to computation of which subset of random numbers should be revealed as the OTS of the message (See Appendix B).

In SAOTS protocol, the server needs to perform one hash to get the hash of the message and 75 hash operations ($p = 75$ if SHS is used) to verify the OTS if all of 165 hash values constitutes the public key. By a simple trick and with a cost of additional hash operation for the server we can reduce the length of public key to a single hash value. The idea is simple: as the public key, calculate the hash of concatenation of all the 165 hashes. Now to be able to verify the OTS the sender should send the chosen 75 random numbers and the other 90 random number's hash value. In each run of the protocol the user should send one signature and one public key so if the length of random number is equal to the length of hash value, in overall the signer should send $75 + 90 + 1 = 166$ hash values to the server.

Table IV.2 again makes a comparison between the two protocols but now in terms of communication efficiency.

m: length of message

h: length of random numbers and hash values (the server's statement is assumed to be equal to the length of hash)

s: length of signature

n: number of random numbers used in the OTS

As seen from this table SAOTS provides more efficiency with respect to number of rounds but with an increase in the length of the messages exchanged. It will be shown in the next two subsections that a decrease in the number of rounds of the protocol is generally much more important than an increase in the bandwidth usage as far as communication efficiency is concerned.

Table IV.2: Communication Comparison of SAS and SAOTS Protocols.

		SAS	SAOTS
Number of rounds		3	2
message	round 1	$m+h$	$m+(n+1)h$
length	round 2	$m+h+s$	$m+h+s$
(in byte)	round 3	$m+2h+s$	-

IV.4.2 Theoretical Comparison of Network Delays

In order to compare the network delays of SAOTS and SAS, we have implemented both schemes and measure the time delays in real-time as discussed in next subsection. But before introducing this study, it is worth trying to evaluate their delay performance theoretically as much as possible. First of all, we have seen that the network delay is composed of several delay elements so it can be expressed as a sum of these:

$$Delay_{network} = D_n = D_t + D_p + D_i + D_q + D_a \tag{IV.3}$$

Where the elements on the right side are transmission, propagation, interface, queuing and other delays respectively. The interface delay is the delay on the end hosts i.e., operating system and protocol overhead. If we ignore other delays then we can rewrite the formula for SAS and SAOTS as follows:

$$SAS : \frac{3m+4h+2s}{BW} + 3D_p + 3D_i + D_{q3}$$

$$SAOTS : \frac{2m+(n+2)h+s}{BW} + 2D_p + 2D_i + D_{q2}$$

Note that in the above formulas, BW denotes the bandwidth of the network and propagation delay and interface delay is a invariant function of number of rounds but queuing delay is not. As a matter of fact, one might expect an increase in this delay when the packet size increases, but the dependency relation is probabilistic and highly dependant on the underlying network architecture. If we perform a subtraction on these two formulas, we finally get the following inequality:

$$D_p + D_i + D_{q3} > \frac{(n-2)h - s - m}{BW} + D_{q2}$$

If this inequality holds then network delay is smaller in SAOTS otherwise it is bigger. On the top of the first term on the right side, the extra amount of bits transmitted in SAOTS is given. The experiences have shown that in most situations, transmission delay is not the dominant factor of the network delay [57]. (Section 6.6 of [57] is a valuable guideline on performance issues in computer networks in general and most of the arguments there are in favor of SAOTS.)

An Example to Compare the Network Delays: We can use SHS as the hash function to get the hash of the message (n=165) and RSA with a 1024-bit as the signature algorithm (s=1024) and assume a typical message length of 1024 bits.

One of the most important parameters we should decide in implementation of SAOTS (and SAS) is the length of random numbers and hash values used in an OTS. To have the same security level of a 1024-bit RSA signature, we see that it is sufficient to set the length of random numbers and the hash values to 80 bits [56]. For this purpose we can employ a well-known hash algorithm like SHS producing 160 bits output and folding the output of SHS with exclusive-or to produce a 80 bit output (h=80).

Then in SAOTS we require to transmit approximately 11 Kbits additionally. So, for instance if we take the bandwidth as 1 Mbits/sec and one-way propagation delay as 50 msecs and assume that the queuing delay increase is equal to the interface delay decrease if SAOTS is preferred, then the network delay in SAS is more than four times the delay in SAOTS.

IV.4.3 Implementation and Experiments

To have a more concrete comparison of SAS and SAOTS, we have implemented both of them using MIRACL library [?]. A PC running Windows 2000 with an 800 MHz Pentium III and a 128 MB memory was chosen as the VS and a PC running Windows 95 with a 200 MHz Pentium I and a 32 MB memory was chosen as the clients' machine. Note that today's high-end PDA's and palmtops have a processor speed of 200 MHz. The compiler was Microsoft Visual C++ Version 6.0. We have conducted two experiments. One of them was over a 10 Mbit Ethernet LAN and the other was over the WAN (Internet) with a very long distance between machines (the clients were running on Middle East Technical University in Ankara, Turkey and the VS was running on UCLA, Los Angeles, USA).

RSA with a 1024 bit key and SHS with a 160 bit output was used and $m = 165$, $p = 75$ and $h = 80$ were the OTS parameters. The public key e of RSA was chosen to be 65537 since choosing $e = 3$ might cause some security vulnerabilities. Remember that SAS is a three-round and SAOTS is a two-round

Table IV.3: Performance Measurements of Cryptography Primitives (msecs).

	Pentium III 800 Mhz	Pentium I 200 Mhz
SHS	0.028	0.156
RSA(verifying)	2.220	13.893
RSA(signing)	9.454	59.162
Mapping	0.02	0.1

protocol. This is why the total delay (the network delay until the signed message is received by the receiver) is smaller in SAOTS in spite of greater bandwidth utilization [5]. Table IV.3 gives the performance measurements of cryptography primitives on two platforms used and Table IV.4 summarizes our findings of the experiments.

These experimental results show that the total time required to send a signed message to the receiver(s) in SAS protocol is at least twice the time in SAOTS.

We have also seen that there is a threshold for the network delay where signing in a traditional way and send the signed message directly to the receiver becomes more delay-efficient (computation plus communication delay) for the user [6]. But SAS and especially SAOTS are still preferable in applications where a server needs to be utilized anyway e.g., e-mail, chat over a server etc.

It is straightforward to see that the gain we obtain using SAOTS will increase if

1. If the protocol is operating in an environment with greater network delays.

2. A public-key algorithm with a longer key (e.g., 2048-bit RSA) is to be used

[5] The network delay for WAN has a big variance so the numbers given here are just for the purpose of giving a general idea.

[6] The value of this threshold for the network delay is $59.162 - 23.531 = 35.631$ msecs for SAS and $59.162 - 11.906 = 47.256$ msecs for SAOTS.

Table IV.4: Experimental Comparison of SAS and SAOTS Protocols (msecs).

	SAS	SAOTS
Originator's computation	14.049	0.256
Server's computation	9.482	11.650
Receiver's computation	14.205	14.049
Network delay (LAN)	1.9	1.5
Network delay (WAN)	340	250

since as the verification time of public-key signature will increase, only the performance of SAS will get worse.

3. A public key algorithm (e.g., DSA [8]) where verification of the signature is not more efficient than generating the signature is used.

4. A more powerful server is employed.

IV.5 A Size Reduction Technique: SAOTS with Hash Chains

In SAOTS protocol we have proposed, in the first round the signer sends to the server the message and a one-time signature as well as the public key for the next signature. Is it possible to reduce the size of this bulk?

Yes, it is possible to avoid sending the public key of one-time signature if we utilize the idea of hash chaining [58]. But as we will see, it does not come free.

Now in the initialization phase of SAOTS similar to SAS each user gets a certificate from an offline certification authority for the hash array of length n

$$K_0^k, K_1^k, K_2^k, \dots K_{n-1}^k \tag{IV.4}$$

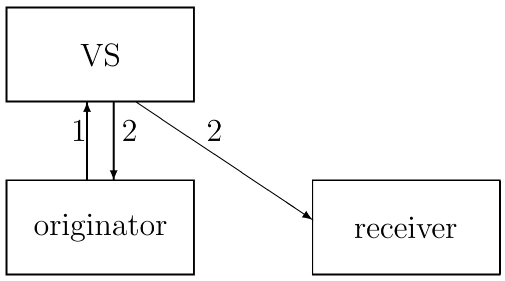

Figure IV.3: SAOTS with Hash Chain Operating in Two Rounds.

Where n is chosen to be large enough to map the hashed message (n=165 for SHS). Each element of the array is the last element of a hash chain of length k where

$$K_j^k = h^k(s_j) = h(K_j^{k-1}) \quad (for \;\; j = 0 \;\; to \;\; n-1) \tag{IV.5}$$

In this equation, $h^k(s)$ is a hash chain of length k and means we apply hash function $h()$ k times to an initial input s.

Then the modified version of SAOTS (SAOTS with hash chains) works in two rounds again as illustrated in Figure IV.3 but now the originator should receive, verify and store the signature coming from the server:

1. The originator (O) sends m and S^i to VS where
   - m is the message
   - $S^i = (K^i_{a_1}, K^i_{a_2}, K^i_{a_3}, \ldots, K^i_{a_p})$ denotes the subset of the array of length p that maps the message to an OTS (composed of i^{th} elements of hash chains). The counter i is initially set to $k-1$ and decremented after each run.

2. Having received O's request, VS performs the followings:

   - Whether O's certificate is revoked or not.
   - Computes the mapping of $h(m)$ or in other words finds out which subset S^i would correspond to the OTS of the message.
   - Checks whether for ($q = 1$ to p) $h^{n-i}(K^i_{a_q}) = K^n_{a_q}$ or in a more efficient way ($q = 1$ to p) $h(K^i_{a_q}) = K^{i+1}_{a_q}$ if $K^{i+1}_{a_q}$ has already been received (it depends on the previous message mapping).

If these are all OK, VS signs the message and sends it back to both R and O. For the VS's signature, there are two possibilities:

- VS can sign with a traditional public key signature
- If in the initialization phase VS gets a certificate for the hash chain array (it should prepare a separate hash chain array for each registered user), VS can also sign using one-time signatures thereby alleviate the verification for those who can not perform public key operations.

After receiving the signed message from VS, both O and R verifies VS's signature. For secure operation, O should sign the next message only after this (off-line) verification. Otherwise the following attacks can be performed:

- An attacker can generate the (i^{th}) hash elements required to forge the signature on a different message by applying a hash operation on the $(i-1)^{th}$ hashes. He then inserts this new signature instead of the O's signature. However the attacker cannot generate a valid signature for any message he wants. This depends on the previous messages signed and the mapping algorithm.

- If VS gets the previous signed message but does not send the signature on this message before he gets the next signed message from O, an attacker cannot forge a signature but now VS can generate a forged signature and that cheating cannot be proven. For O, the signed message will be the only proof for VS's cheating.

Off-line verification and storing of VS's signature before signing another message will be sufficient to avoid these attacks. Since VS has signed the O's signature with the i^{th} hash elements, if an attacker sends a forged signature using the i^{th} hash elements to VS, VS rejects this request (Look at the step 2 of the modified SAOTS protocol above) and the attack is not successful. The VS cannot even generate a forged signature because it has signed the genuine message previously.

Finally, we can say that with a trade of heavier computation requirement and an extra storage requirement in this new version of SAOTS protocol using hash chains we can have a total length save of $(n + 1 - p)$ hashes (for SHS, $n = 165$, $p = 75$ and $h = 80$, the save is 7.28 Kbits) in round 1 of the protocol.

The computation would be heavier not only for the originator but also for the server. In more concrete terms, the server needs to perform n hashes instead of p on average to verify the OTS of O. To conclude this section, average number of hash computations needed by the server to verify an OTS in SAOTS with hash chains is formulated:

For the OTS of a message, the server receives p out of n hashes. Let's call K^i any one of these hashes that is at depth i of the chain. To verify K^i, it requires any one of the next hashes in the chain (from $i+1$ to k). If next hash (K^{i+1}) in the chain is available, then it needs only one hash operation to compute. If not the hash computation requirement will increase. The probability that K^{i+1} has been received is $(p/n = \alpha)$. So if we assume that k is a big number, average number of hash computations required for one hash will be approximated by

$$\sum_{t=0}^{\infty} \alpha(1-\alpha)^t (t+1) \tag{IV.6}$$

Since $\alpha < 1$, this serial sum is equal to $1/\alpha$.

To find the total number of hash operations required, we need to multiply this number with p, then finally we find the average number of hash computations needed to verify an OTS in modified version of SAOTS protocol as

$$p * \frac{1}{p/n} = n \tag{IV.7}$$

As an example, to map a 160-bit message, we can choose $n = 165$ and $p = 75$ where the server needs to do 165 hash computations to verify the OTS.

IV.6 Revocation of Public Key Certificates

Increased use of digital signatures emphasizes the importance of effective and efficient revocation methods so that if a user does something that warrants revocation of his security privileges i.e., he might be fired or may suspect that his private key has been compromised, he should not generate valid digital signatures on any further messages (However, signatures generated prior to revocation may need to remain valid).

In Online Certificate Status Protocol (OCSP) [59] (today's state-of-the-art approach to solve the revocation problem) to provide timely revocation information, upon verifier's query a validation server sends back a signed response showing the sender's certificate's current status. The drawback here is that it is impossible to ask a validation server whether a certificate was valid at the time of signing.

Immediate revocation (the user cannot sign immediately after the revocation takes place) is possible if an online VS is employed. In order to revoke a user's public key, it is sufficient to notify the server. The server maintains a list of revoked users and it rejects signing on behalf of the user if his public key is in the list.

We now want to show a deficiency in the revocation capability of SAS protocol [53]. SAS protocol works in three rounds as explained in subsection IV.1.4. Think of a situation where the user gets the public key signature from the VS in round 2 and postpones the execution of round 3. He then notifies the server to revoke his public key (e.g., claims that his private key has been stolen). Afterwards, he can cheat by executing round 3 and generating a valid signature although his public key has already been revoked.

In our proposal, this deficiency is elliminated since SAOTS protocol works in two steps in opposed to three in SAS.

IV.7 Integrating Attribute Certificates with SAOTS

Public key certificates provide the best solution for most applications however some new applications like the ones in the area of health care require more than that. Due to very dynamic rights of permissions and admissions, a certificate different than the public-key certificate is employed for professionals in healthcare and other similar organizational settings and is called attribute certificate.

Just like public-key certificates where an authentic link is established between an entity and his public key, attribute certificates link attributes to an entity in an authentic manner. An attribute certificate does not exist autonomously - it is rather bound to a base certificate: the public key certificate of the entity. Exemplarily, attributes describe some of the following characteristics: general authorizations, international or national specific data, delegations for other persons, temporary rights etc.

Similar to PKI, the attribute certificates framework defined by ITU provides a foundation upon which a Privilege Management Infrastructure (PMI) can be built [60].

IV.7.1 Previous Work on the Design of Privilege Management Infrastructure

In a PKI, the typical solution of the problem with public keys we have introduced in section II.2 is to have trusted nodes known as certification authorities (CA) which are "off-line" entities issuing public key certificates and putting them in any convenient location, such as the directory service. Being an offline entity, unlike the secret key equivalent of KDC (key distribution center) CA is neither a performance bottleneck nor a single point of failure; two parties (the signer and the verifier) can have a secure communication without a third party's involvement.

However there is potential problem with CAs as mentioned in previous section. Consider an organization where all employees have certain authorizations. Suppose that an employee named Alice does something that warrants immediate revocation of her signing capability. For example Alice might be fired and since Alice is now a disgruntled ex-employee, it is required to alert others not to accept his signature as valid. One solution to this problem is to use a certificate revocation list (CRL) which lists the serial numbers of certificates that should not be honored. We have observed that CRLs and other similar approaches can be easily outdated (it is hard to update the list instantly) and are not the perfect answer to most organizations which need to provide more timely information about revocation status of employees. Current state-of-the-art approach is to use an on-line revocation server that can be queried over the network about the revocation status.

So an up-to-date PKI should incorporate an online revocation server as well in addition to the components introduced in subsection II.2.2. When the notion of PMI is first introduced, one obvious conventional solution is to use an architecture similar to PKI for it. Alternatively, in a recent paper, the authors proposed to combine three components (certificate authority, a directory service and an online revocation server) and name it as Attribute Certificate

Service Unit (ACSU) [7]. The ACSU is now composed of three components: (1) Attribute Authority (2) database to store the attribute certificates (3) Attribute Server.

The CAs in a PKI are generally external to the organization i.e., private companies or goverment agencies. This is because ”identity” tends to have a global meaning. On the other hand attributes have a more local meaning e.g., being a project member. The new PMI scheme is therefore managed locally and the ACSU is located inside the organization. Although linked, both infrastructures (PKI and PMI) can be autonomous and managed independently. The advantages of such a design can be summarized as follows:

1. Simplified design (revocation is a built-in element in the system).
2. Local control over employees' attribute certificates and their privileges.
3. Confidentiality of private privilege information.

IV.7.2 The Problems in Dawson et al.'s Approach

In Dawson et al.'s approach when Bob receives a signature from Alice; to authenticate her, he gets her public key from a directory, he queries a revocation server to obtain the status of her public key certificate and then for authorization information he queries the ACSU to obtain her attribute certificate. The ACSU returns back only valid (not revoked) attribute certificates. We have investigated the following problems in this design:

[7] In their paper, they also propose a method to employ multiple ACSUs for scalability reasons.

1. Although it is simpler than the conventional approach, Bob still needs to query two different servers to verify the signature.

2. Immediate revocation (fine grained control over signing capabilities) is not supported. The revocation server and ACSU can return the revocation information at the time of query not at the time of signing.

The first problem can be easily handled by covering the functionality of public key certificate revocation server also inside the ACSU so that Bob queries only one server but the second problem needs a more careful treatment.

In a recent paper [62], the authors observe that in revocation techniques where the receiver of a signature asks a server for revocation information do not provide immediate revocation.

In their paper they also propose a method to provide immediate revocation. However in their paper they do not discuss how attribute certificates can be managed. We will now discuss a new design where attribute certificates are also involved. But first we would like to extend a definition of the feature from which the term immediate revocation is derived:

Modified Definition of Binding Signature Semantics: A valid digital signature on a particular message can be generated only if its owner has a valid public key certificate and a valid attribute certificate (is authorized to sign that message) "at the time of signing".

IV.7.3 When Do We Need Binding Signature Semantics?

Do we really need binding signature semantics? or it is sufficient to check the signature's validity at the verification time. The correct answer really depends on the application. In an application where signatures are used to allow the authenticated entity to have an access to a resource in the system, the protocol works in real time in other words signing and verification are almost performed at the same time. In this situation binding signature semantics can be considered as a luxury. However it is not difficult to find an application that really needs such a fine grained control.

For instance, suppose that the second manager of a company is permitted to sign purchase orders when the first manager is on leave. Since the verification of the signature on the purchase orders can be done at some later time, it is vital to check the validity of certificates at the time of signing.

IV.7.4 Our Proposal: Extending Fine-Grained Control With Attribute Certificates

Fine grained control can be easily provided by employing a semi-trusted online entity in signature generation. As a matter of fact, the ACSU in Dawson et al.'s paper [61] can act as this online server. Now the ACSU is communicating with the signer not the verifier. There are various alternatives for the implementation [8] but the main idea is the same:

[8] One is presented in Boneh et al.'s paper [62] and one other is the SAOTS protocol we have proposed.

The server would participate in the signature generation only if the user is allowed to sign the message at hand (has valid public key and attribute certificates). Without server's participation there is no way for the user to generate a valid signature.

More precisely, the proposed extended protocol involving the attribute certificate works as follows:

- The user produces his part on the signature of the message he would like to sign and sends it to the ACSU.
- ACSU checks whether the user's public key certificate is valid and also checks whether he has a valid attribute certificate to sign the message. If so, it generates its part of the signature on the message. Now the signature on the message is complete. It is sent to the intended receiver(s).
- The receiver verifies the signature on the message. He does not need to interact with anybody for verification.

IV.7.5 Final Remarks

In a recent paper [61], a design of privilege management infrastructure was proposed where the receiver of a signature queries an online server named as ACSU (which is also the authority on issuing attribute certificates) to get the signer's attribute certificate. In this section, we present a new design where the signer not the receiver communicates with ACSU. By using our proposal the verifier can check the validity of the signature (1) at the time of signing not at the time of receipt and (2) in a more convenient way without interacting with a server.

Our proposal here can be thought as the combination of clever ideas in two papers (Boneh et al's [62] and Dawson et al.'s [61])

IV.8 Summary

In this chapter we have presented a new efficient verifiable server assisted signature protocol called SAOTS. By using a chaining technique where we attach the public key of one-time signature for the next message to the current message, it is shown that neither public key operations nor signature storage is required for the user's constrained device to prove the server's cheating. Therefore our protocol is more power efficient than on-line/off-line signatures and more computation, storage and power efficient than previous verifiable server assisted signature protocols.

SAOTS protocol implies that for generating a public key signature fully trusted proxy servers are no longer the only option for pervasive devices which cannot perform public key operations by themselves.

In recent years researchers come up with other signature schemes which do not base on public-key operations. One of them was the BIBA scheme proposed by A. Perrig [63]. BIBA's advantages are smaller signature lengths and faster verifying times. Designing and evaluating the performance of a server assisted BIBA signature is a promising future work. As another future work, we also plan to compare the performance of SAOTS (and SAOTS with hash chains) with respect to signatures using proxy servers. It will not be a surprise if we see that the proxy signatures turn out to be more efficient but it is important to know how much degradation we have in the performance if we do not want to depend on a fully trusted third party.

CHAPTER V

EFFICIENT VERIFICATION OF SIGNATURES IN REAL-TIME TELERADIOLOGY

Teleradiology is already impinging on everyday practice, but rapid expansion, driven by an ongoing desire for increasing cost effectiveness, will be seen in the near future. It is no longer a fairytale dream but a useful tool to be used for greater benefit of patients.

- *Richard Wright*

There are three things to be looked to in a building: that it stand on the right spot; that it be securely founded; that it be successfully executed.

- *Goethe*

After requirement analysis and design phases, there comes the implementation phase and likewise following SCOTP and SAOTS, we present EVEREST, our third and the last proposal. In this chapter, EVEREST would demonstrate that a simple trick in the implementation could yield improved performance in the verification of digital signatures in real-time teleradiology.

Let us begin our story from the very beginning. To improve the quality, cost and access, the healthcare industry is endeavoring to attain the development and deployment of electronic patient records. Not only used by radiologists, but also by other clinicians and specialists, medical images are the indispensable part of electronic patient records and are considered to be at the heart of the patient's diagnosis, determination of therapy and follow up [64].

The medical images are traditionally recorded on film from diagnostic devices like CT, MRI scanners and X-ray systems. Medical institutions struggle, not only with the cost of film but also with the staffing and storage requirements for the infrastructure to manage all of these images. Picture Archiving & Communications Systems (PACS) [65] have been developed to represent medical images in digital form. Medical practices start converting their older analog imaging systems to digital. This conversion integrates storage and distribution of digital medical images as illustrated in Figure V.1.

Traditionally, only firewalls are used to protect the security of medical images [66] (a firewall examines each network packet to determine whether to forward it toward its destination). Intruders frequently bypass firewall security and in the literature a number of papers have been published to claim that this is inad-

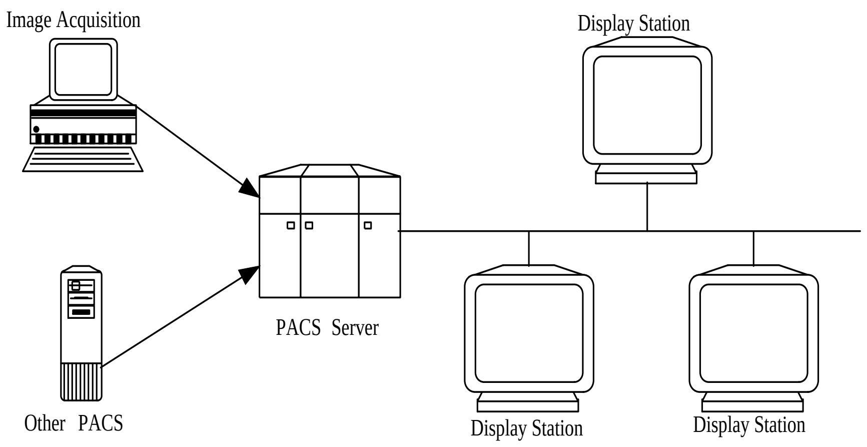

Figure V.1: Picture Archiving and Communications System (PACS)

equate for the protection and should be supported by other advanced security measures [64, 66, 67]. Meanwhile, a recent security extension to the DICOM [1] standard was proposed to describe how digital signatures can be added to DICOM images so that the authenticity and integrity of the image is guaranteed [68].

In addition, the demand for more advanced security measures increases with the rapid expansion seen in teleradiology. Teleradiology is the umbrella term to define the concepts of digitally transmitting radiographic patient images and consultative text from one location to another over public networks [2]. Teleradiology is a convenient, cost-effective and productivity-enhancing technology and its applications can be grouped into two general categories: (1) real-time on-demand services facilitating remote interactive communication and emergency

[1] The DICOM standard defines the basis of the interface mechanism allowing different manufacturers' digital imaging systems to communicate images.

[2] Today, it really becomes fuzzy where a public network starts and when it ends therefore we think the term teleradiology can also be used to cover PACS applications. As a matter of fact the current trend for PACS is to provide low-cost connectivity from outside the enterprise as well.

triage, (2) non-interactive transmission and storage of images for later interpretation.

Meanwhile, previous studies have also observed a performance problem when the security measures are employed. This problem is more severe when medical images are of concern since the image sizes might be huge. Besides, in real-time teleradiology applications efficiency is crucial since the system is expected to respond very quickly. In this chapter our objective is to provide a solution for this performance problem. More precisely, we propose an efficient methodology called EVEREST for verification of digital signatures to optimize the performance of on-demand viewing of secure medical images [69]. We believe that our work is the first one that aims to improve the real-time efficiency of secure teleradiology applications. Additionally, one other important advantage of the proposed method is the communication efficiency since getting the entire image file would not be necessary to detect a tampering.

The rest of this chapter is organized as follows. Next section discusses the importance of security and digital signatures for medical images. Section V.2 restates the efficiency problem in more detail. Section V.3 is about the previous studies and their shortcomings. In section V.4, we propose our method EVEREST for the real-time efficient verification of signatures on medical images. Section V.5 provides the results of our experiments and performance evaluations. Finally, section V.6 concludes this chapter.

V.1 Importance of Medical Image Security

As mentioned, in most of the current applications medical images are secured by a firewall only [66]. However in the Internet and WWW era where networks and internetworks become more open than ever before in an attempt to make information available everywhere, security problems become more alarming than ever before.

Today attacks are changing in nature and becomes much more dangerous, meanwhile many of us are critically dependant on the security of PACS systems and medical images. Consequently, it is evident that current protection mechanisms are inadequate and needs to be supported by other advanced security techniques. We believe that health care field is in urgent need of the support of security protocols and cryptographic algorithms and without being late should follow other sectors (e.g., banking, military etc.) to employ them effectively and efficiently.

Some might think that information systems in healthcare do not need such security countermeasures simply because unlike banking and military there is no profit an attacker can acquire. While you can argue whether a profit exists or not in healthcare [3], we would like to remind you the “publicity attack” [1] where the attacker is only in pursuit of to get his name in the newspaper [4]. As a matter of fact hacking into a bank’s web server happens so often and ordinary that a hospital’s information system might serve better for the purpose

[3] In both United States and Britain there are people who earn their living by only violating medical record privacy of others [37].

[4] Why did the man burn down the Temple of Artemis in ancient Greece?

of getting famous. Moreover without proper security measures it takes less time and less money. Lastly as a supporting argument we would like to mention HIPAA Security Rule in USA [5], which has been recently passed into federal law and provides a conceptual framework for healthcare information security and sets out strict and significant federal penalties for non-compliance.

We would like to restate one important point here. That is in security, "context matters more than technology [1]." In other words before choosing the specific security protocol or cryptographic tool, to be effective, it is essential to decide on the security target by answering the question, "what kind of security do we need?"

As we see in Section I.1, three main information security targets can be listed as:

- Confidentiality: prevention of unauthorised disclosure of information.
- Integrity: prevention of unauthorised modification of information.
- Availability: prevention of unauthorised witholding of information.

While the medical professions are rapidly coming to depend on computers and networks, attacks that degrade the availability of digital information (such as viruses and distributed denial of service kind attacks) might have serious consequences and therefore the importance of the third target, availability can never be underestimated. Moreover the existing techniques used to protect the availability of information are observed to be more premature and less effective.

[5] http://www.hhs.gov/ocr/hipaa/, US Department of Health and Human Services, Last access: September 17, 2003.

However we will not discuss this issue in detail here (we will add a few words in the last section of this chapter). Instead in this chapter the second main target, “integrity” is our main concern.

A medical image consists of two parts, a short simple image header and a big anonymous image body. In most scenarios only the header containing the sensitive patient information needs to be kept confidential however the integrity of the entire image file needs to be assured. This is the reason why unlike encryption it is not secure to sign only a portion of the image file for efficiency reasons [66]. In fact, Cao et al. have demonstrated with some visual examples how easy to insert artifacts within the image, which causes confusion and contradiction during diagnosis [66].

V.2 The Performance Problem

Today, digital signatures explained in subsection II.1.6 are accepted as the defacto standard for ensuring data integrity and authenticity. In the implementation of digital signatures if the message has an average size (e.g., in kilobytes range), computing the hash value of the message can be performed in microseconds and is not a dominant factor in the overall delay of signature generation/verification (see Figure II.1). However in teleradiology applications, a typical examination generates between 10 MBs and 40 MBs or much higher with volume magnetic resonance imaging (MRI), volume zoom CT and digital mammography. The high extreme is in digital mammography, which generates 160 MBs per examination [67]. Therefore, when medical images are of concern, in contrast to other applications a considerable amount of time for signing and verification is spent to compute the hash value of the image (In a Pentium III 500 MHz machine computing the hash of a 50 MB image takes around 10 seconds).

As far as real-time teleradiology and on-demand viewing is concerned (images are not batched long before review [6]), verification efficiency is more important than efficiency for signing because of three main reasons:

1. Signing is usually performed offline but the verification needs to be carried out in real-time.

2. Generally an image is signed only for once but needs to be verified many times over a period of years.

3. Server machine that signs the image is chosen to be a powerful machine to serve multiple receivers at the same time however the receiver machine might be a constrained device (the recent studies [70] show that to display medical images even the handheld devices can be practically used).

So far, what we have said is about computational delay efficiency. Generally speaking, another issue in a typical network application is communication efficiency i.e., how to avoid wasting bandwidth resources. Think of a situation where after receiving the entire image file verification of the signature fails. In all of the digital signature algorithms available today the determination of the region the tampering is not possible hence a message should be sent back to PACS server requesting for retransmission of the entire image file. Worse than

[6] Batching is not practical (a) in emergency scenarios (b) when the storage capacity of the display station is limited.

that the retransmission repeats if the verification continues to fail in subsequent receipts.

From the above discussion it becomes clear that a method for verification of digital signatures which is more efficient in terms of both computational and communicational load is higly desired in an application where big files are of concern especially as in teleradiology where medical images are tens of megabytes in size.

V.3 Previous Studies

In a recent article [71], the authors discussed a technique for efficient encryption of medical images. Their work is based on so called “partial encryption” or selective encryption” which protects only the most important part of the image. A similar approach for digital signatures is not applicable because usually the integrity of the entire image file needs to be protected. It is reasonable to assume that if a medical image has some unnecessary parts then in a pre-processing phase these parts can be removed from the image. This pre-processing would be beneficial not only for the efficiency of digital signatures but also to decrease the storage and tranmission requirements in general.

In a bigger picture there are two components that affect the performance of on-demand viewing of digitally signed medical images:

- The transmission time from the PACS server to display station (Figure V.1).
- The time for the computation of hash and verification of signature.

At this point, we would like to introduce some of the techniques employed to decrease the transmission delay, the first delay component:

- **Increasing Bandwidth Capacity:** Of course it is obvious that the ultimate solution and the current trend is based on an investment to increase the bandwidth capacity of the infrastructure. As a result in the near future when available bandwidths are measured in Gb/s the delay to compute the hash value of the image will become more significant.

- **Applying Image Compression:** Other than digital signatures, another state of the art and emerging technology in teleradiology applications is image compression. There are two types of compression; while a 100:1 ratio is possible in lossy compression, the current maximum limit for lossless compression is 3:1 [72]. In practice, medical community is reluctant to accept the lossy compression to be used for medical images and this is why we should deal with tens of megabytes in teleradiology applications even when compression techniques are employed.

- **Speed-optimized Transmission Method:** The other solution to optimize the transmission has been implemented in popular freeware download programs. The idea behind these simple tools is to split the long file into several pieces on the server side and open a separate connection to transmit in parallel each piece upon receiver's request.

The second delay component, the inefficiency incurred by digital signatures has been discovered previously [64, 66, 67] (but none of them proposed a solution). In [67], the authors present a method to sign digital mammography images and their findings about the inefficiency were much more severe. This is because their proposal includes embedding the digital signature into the image itself instead of attaching it to the head or end of the image file. As a consequence an extra amount of time would be necessary for the embedding computation.

Just like other teleradiology applications, digital mammography uses DICOM standard [68] that did not previously support digital signatures. The image embedded with the digital signature still confirms with the DICOM image format standard. That is the main reason behind the authors' design rationale. However, recently a third security extension to the DICOM standard describes how digital signatures can be transmitted with the DICOM images [68]. So for the time being we believe that embedding the digital signature to the image does not have any advantage over attaching. Moreover, the proposed embedding method has some security weaknesses, as we will mention in short.

The Security Weakness in Zhou et al.'s Method: One additional goal of embedding the signature to the image itself is to make it difficult to remove from the image [7], Zhou et al. proposed a method where the least significant bit of a random pixel of the image is replaced by one bit of the digital signature [67]. In order to select the random pixel, they used a linear congruential generator with a modulus of 1771875. Then, the seed value that is agreed on between the

[7] We think the reason to try to make the signature unremoveable was not clear in their paper. These so-called steganography or watermarking techniques are usually employed either for privacy or copyright purposes [6].

sender and the receiver has a space of around 21 bits (Another drawback here is to assume that the receiver and the sender can agree on a shared secret).

Seeding the random number generator with a seed that is from too small a space is a typical mistake for picking random numbers [10]. The problem is that there would be inadequate number of possible numbers it would ever choose to avoid a "brute-force attack" where without using any intelligence an attacker try every seed value to discover the hidden signature (2^{21} in our case which is not considered to be safe today for an adversary having an average amount of computing power).

At this point we would like to state that the goal of embedding an unremovable watermark that can be verified by a public key is currently a hot topic in security world usually studied under the term "asymmetric watermarking". In a recent paper [73], the authors reviewed the current state of the art and concluded that the problem of designing a perfect asymmetric watermarking scheme has not been solved yet (In their definition "perfect" means secure, unremovable, embedded with a secret key and verified with a public key without contacting a third party).

In another previous work [56], the authors proposed an efficient method for signing digital streams (very long potentially infinite sequence of bits, e.g., a movie or a live broadcast). The solution to digitally sign these streams are of two types. In the one for the finite stream, the basic idea is to divide the stream into blocks and attach authentication information of the current block to the following block by a chaining technique. Although this work has some inspiring ideas for us, it is not directly applicable to the medical images because of two reasons: first of all, images in teleradiology do not match with the properties of digital streams therefore the complexity of the chaining technique used is unnecessary [8]. Secondly, since it assumes to receive the parts in sequential order this work has a poor performance when the speed-optimized transmission method introduced earlier is used.

V.4 Our Solution

In our solution, the key observation we have made is that in the traditional verification the processor of the verifying machine is idle (I/O blocked) while the image is downloaded (the processor is utilized for hash computation only after all the entire image file is transferred). If we come up with a new verification methodology in which we do not need to wait until the entire image file is downloaded, we can utilize the previously idle processor for hash computation and as a result we improve the real-time efficiency of verification or in other words the total time necessary for downloading and verification is decreased. The method proposed can also be considered as a way to parallelize the steps so that the receiver can perform most of the computation in background while he is receiving the image itself. Apparently, our proposal does not decrease the total amount of computation carried out by the processor. It just shifts the time the processor is used for this computation.

[8] One of the characteristics of digital streams, lack of need to buffer large amounts of data is the main motivation behind this design where a block can be deleted after receiving and verifying the next one. However to verify signatures on long files we do not have this limitation.

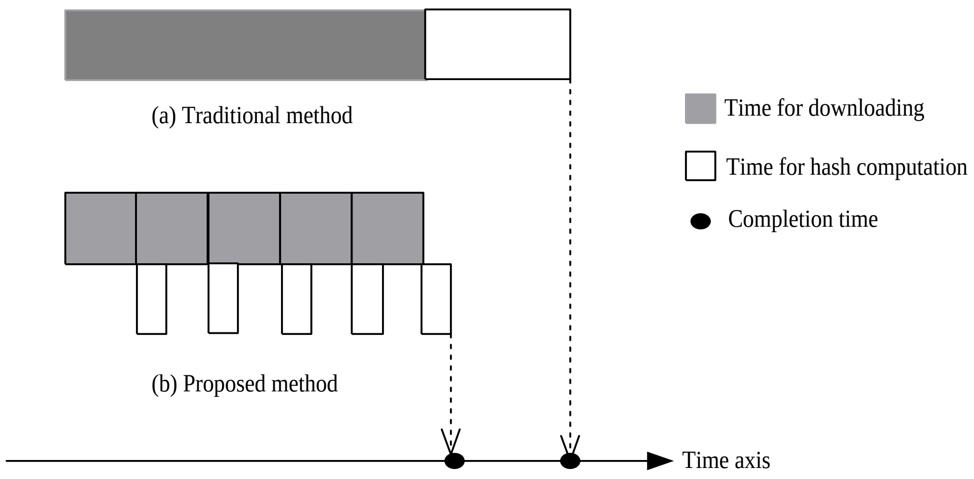

Figure V.2: Comparison of the traditional method and the proposed method EVEREST.

As seen from Figure V.2, there is a similarity between our solution and the technique known as "pipelining" that is widely used in designing more optimized computer processors. With pipelining, the computer architecture allows the next instruction to be fetched while the processor is performing arithmetic operation. The staging of instruction fetching is continuous. The result is an increase in the number of instructions that can be performed during a given time period.

A signature verification method for teleradiology should be flexible to work reliably and efficiently no matter how the underlying transmission takes place. It can be whether a traditional one with a single connection or the speed-optimized method introduced previously. We now introduce one such method called EVEREST (**E**fficient **VER**ification of **E**lectronic (digital) **S**ignatures in real-time **T**eleradiology), which consists of two parts: the set-up part working on the sender side and the downloading and verifying part working on the receiver side.

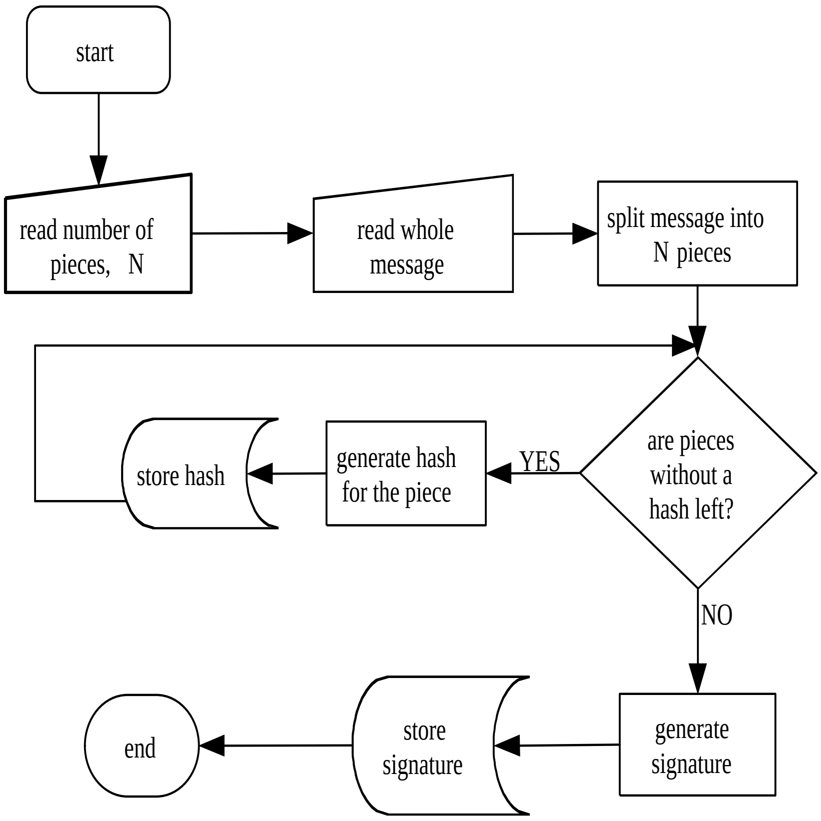

Figure V.3: Signature Generation in EVEREST.

V.4.1 Signing on the Server Side

First, the issue of how many pieces the image file is split into is resolved. It is good in practice to have as many pieces as the number of connections the server would open per receiver when speed-optimized method is used. Optimum number of connections depends on the resources the server machine has and also on the number of independent receivers served at the same time. Note that more connections not always mean more speed. The transmission medium characteristics also affect the optimum number of connections.

Having decided on the number of pieces, then the server executes the procedure illustrated in Figure V.3 for each image it needs to sign:

As we mentioned earlier, in traditional signatures, the hash of the message, $h(x)$ is signed to get the signature $S(x)$ as seen from equation V.1:

$$S(x) = Sign(h(x)) \tag{V.1}$$

In our method, the signing procedure is slightly modified [9]. Equation V.2 shows the generation of signature when the message is split into 3 pieces ("||" means concatenation).

$$S(x) = Sign(h(h(x_1)||h(x_2)||h(x_3))) \tag{V.2}$$

After an image is signed and the hash values and signature is stored, the server is ready for secure transmission.

One important point worth mentioning is that our method keeps the signing computation almost same (due to linearity of hash computation that can be seen in Figure V.5).

V.4.2 Verification on the Receiver Side

Unlike the traditional verification where all the steps are executed in a single main thread, in our method we have two threads working in parallel. While in the MAIN thread, the pieces of the image file, the hash values and the signature are downloaded, the VERIFY thread is used to verify the signature and the hash values as shown in Figure V.4. The verification of the signature and hash values

[9] This modification is unnecessary if the hash function $h()$ statisfies $h(x) = h(h(h(x_1)||x_2)||x_3)$ or $h(x) = h(h(x_1)||h(x_2)||h(x_3))$. However most of the hash functions used in practice does not satisfy any of these equalities.

only begins when the required input becomes available otherwise the VERIFY thread waits in a loop. In the MAIN thread, when downnloading a message piece starts, an I/O block occurs and the CPU can entirely be used by the VERIFY thread until that downloading is finished.

In the VERIFY thread after verifying the signature on the hash values, to verify the hash values, the hash value of each piece received is computed and compared with previously received one. In Figure V.4, "end" means a successful completion of the verification and "quit" means the verification is unsuccessful (a tampering has occurred) and the program terminates and discontinues downloading (results in communication efficiency). Also notice that the operation of EVEREST does not assume to get any specific part first. If speed optimized method is used the receiver opens n separate parallel connections (threads) to the server machine to download each part and in this case we have a total number of $n + 1$ threads working in parallel.

V.4.3 Security Analysis

We claim that EVEREST achieves the same security level of a traditional public key signature scheme. As seen from equation V.2, to forge a signature, an adversary can attempt either to

- Forge the public key signature of the sender.
- Find a message such that the hash value of that message is equal to the hash value of any one of the blocks or the hash value of hashes of the blocks.

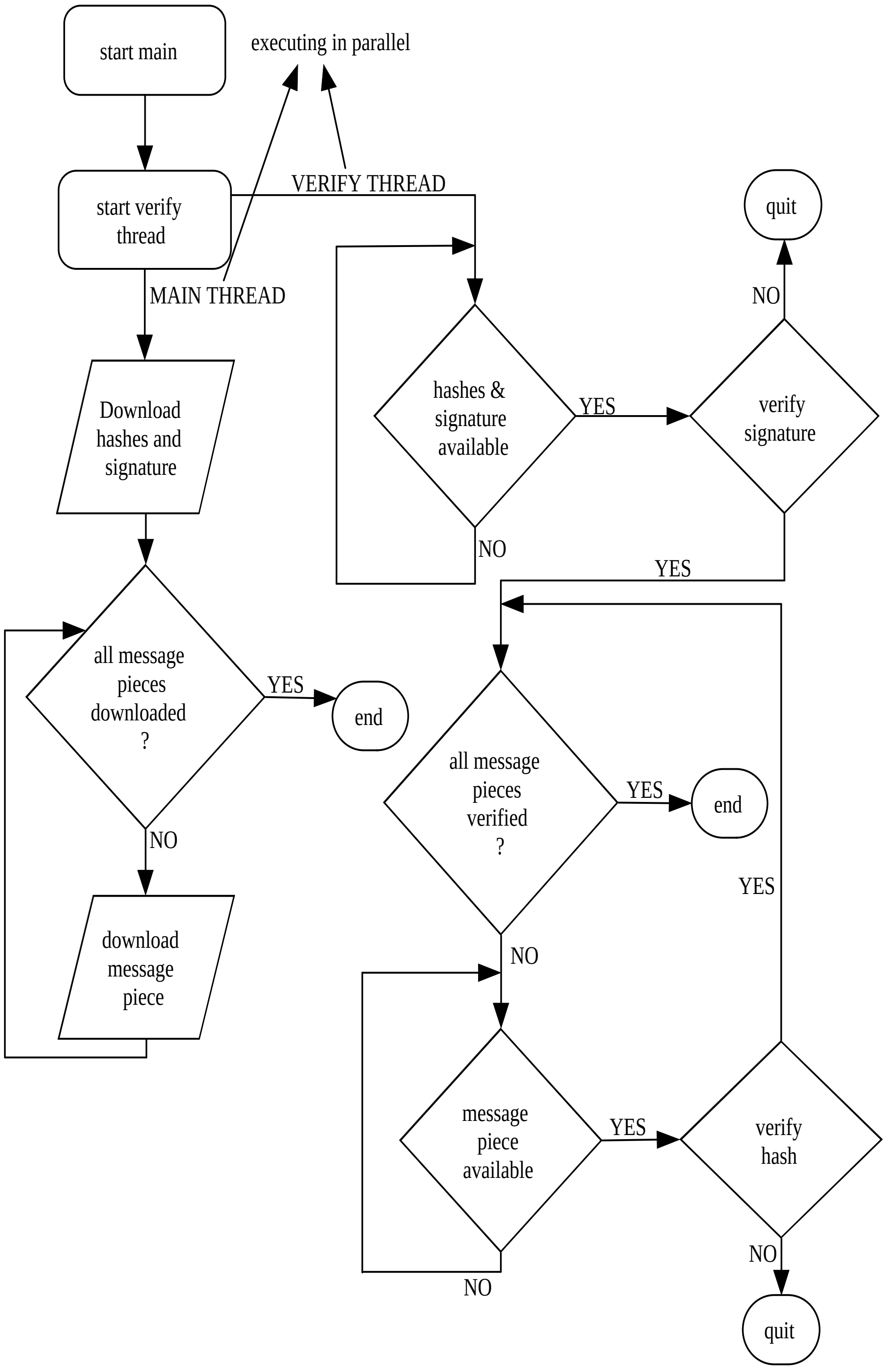

Figure V.4: Signature Verification in EVEREST.

Clearly, the first attack is applicable to the traditional signatures as well. As seen from Figure II.1, traditionally a hash function is first applied to the message in order to produce a fixed-length digest, which is then signed. Hence a successful second attack is at the same time an attack on the traditional signature.

Since there is no attack that can be applied only to our method, our method is as secure as traditional signing.

V.5 Performance Evaluation and Experiments

In this section, we first would like to make an analytical evaluation of the performance of the proposed method and determine the maximum achievable theoretical gain with respect to traditional verification. Next, we provide the results of the experiments we have conducted and compare these results with the analytical findings.

V.5.1 Performance Evaluation

In our method, the transmitted file size is slightly larger than the original case due to extra hash values transmitted. To make a comparison, let T_d denotes the extra time delay for transmission of this extra amount. When the network delay to download each part is bigger than the time required to verify the hash values, it is reasonable to assume that at the time the receiver completes downloading the last part, only computing the hash value of this last part is left to verify the entire image since the processing for hash computation of all the previous parts

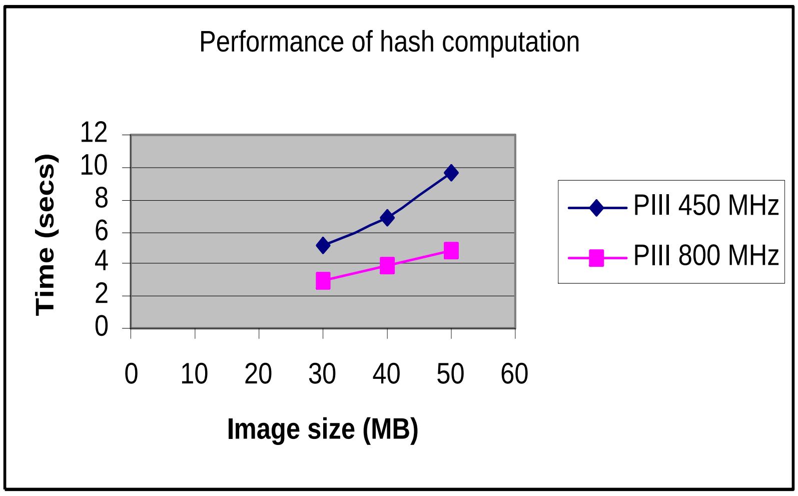

Figure V.5: Performance of Hash Computation Using SHS Algorithm.

and signature verification has been completed by utilizing the idle (I/O blocked) CPU.

We have observed that the time for signature verification is very small compared to compute the hash value of image parts (around 80 msecs if 1024 bit DSA [8] is used as the signature algorithm). As shown in Figure V.5, the time to compute the hash value has a nearly linear dependence on the size of the file (using SHS [9] as the hashing algorithm). So if the time to compute the hash value of the entire file is T_h, then the time to compute the hash value of the last block is approximately $\frac{T_h}{n}$ where n is the number of parts. So traditionally, the total time necessary for verification is as follows:

$$T_{ver\_trad} = T_h + T_s \tag{V.3}$$

T_s corresponds to the time for verification of the digital signature on the hash value. Whereas in our proposed method, by taking into account the value

T_d, the time for verification is

$$T_{ver\_prop} = \frac{T_h}{n} + T_d \tag{V.4}$$

To simplify, if we ignore T_s and T_d (a very small quantity considering network bandwidths available today), then the overall performance gain will be

$$T_{diff} = T_{ver\_trad} - T_{ver\_prop} = T_h(\frac{n-1}{n}) \tag{V.5}$$

Lastly, note that the performance gain in Equation V.5 is achieved when the verification is successful. If verification fails, the computational gain increases because of early detection.

V.5.2 Experiments

A Windows 98 PC with 450 MHz Pentium III and 128 MB memory and a Windows 2000 PC with 800 MHz Pentium III and 256 MB memory were chosen as the slow and fast receiver machines, respectively. A Windows XP Pentium IV with 1.7 GHz Pentium IV and 512 MB memory is the server. JCE (Java Cryptography extension) integrated to the Java 2 SDK Standard Edition v.1.4 was used for the implementation which downloads the image using a single thread. The server and the receivers are connected via 10Mbps LAN. 1024-bit DSA [8] is used for signing and SHS [9] is used as the hash algorithm.

In the programs we have written for precise timing measurements the CPU usage is queried instead of simply measuring the elapsed time. However profiling CPU usage in Java language is not so straight-forward. For those who are

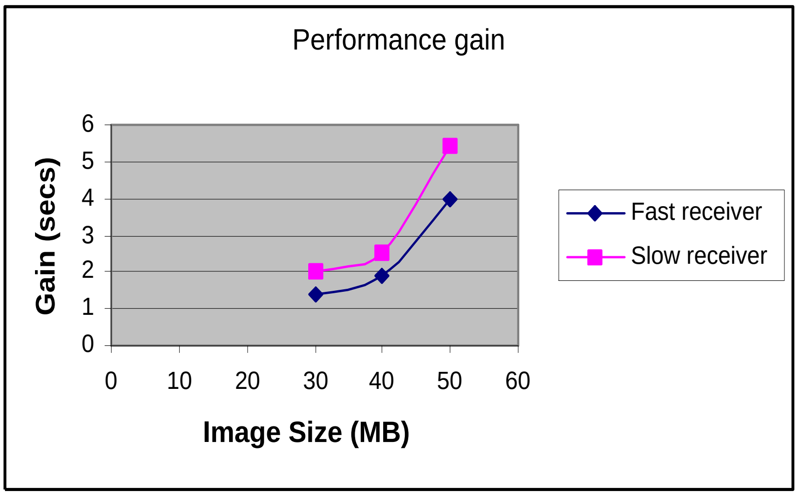

Figure V.6: Performance Gain of EVEREST Using Two Different Receivers.

interested, the techniques we have employed is explained in Appendix C.

Figure V.6 shows the achieved gain by using three message pieces and different types of receiver machines. Using different number of pieces, Figure V.7 is for demonstrating the difference between theoretical gain and experimental gain (using the slow client and 50 MB image size).

As seen from the figures, a significant gain is obtained if our new method is preferred. Overheads due to thread usage, context switches etc. are the possible reasons for the difference between theoretical and experimental gains. However we see that when the number of message pieces increases, experimental gain is approaching to the theoretical gain. The time to download a 50 MB image is observed to be around 50 seconds using 10 Mbps connection. If we would like to specify how much speed-up we obtain in total time to download and verify using our new method then we calculate it as 6-12%. We expect this gain to increase when a higher speed network is used. This is because the time gain

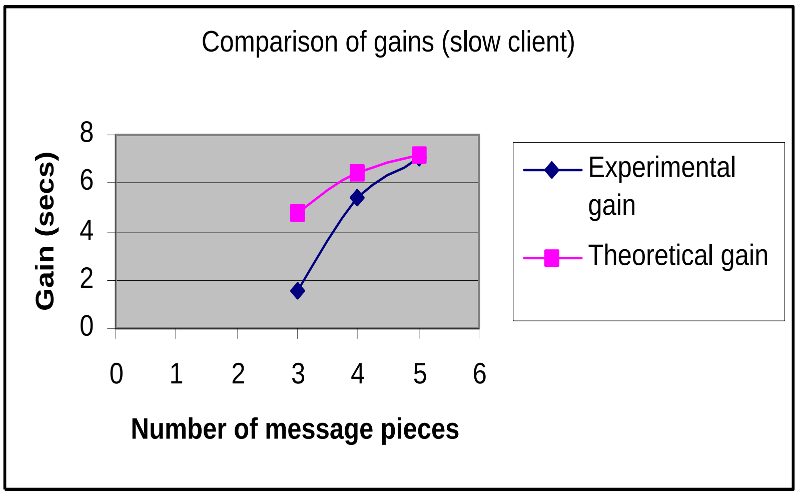

Figure V.7: Theoretical and Experimental Gains in EVEREST.

we obtain will remain almost same but the time to download the image file will drop significantly. We also would like to point out that the theoretical maximum speed-up in total time of our proposal is 50% (when time to download and time for hash computation are equal and the message is split into infinite number of pieces).

V.6 Summary

In spite of high investments in infrastructure for high-speed networking, medical imaging still has performance problems due to huge image file sizes. The introduction of digital signatures makes this problem only worse, since to verify the signature a substantial amount of extra time is required to compute the hash of the image file.

In this chapter, to solve the problem of efficiency in on-demand secure medical imaging applications, we have proposed a method called EVEREST where the receiver can perform most of the hash computation while he is receiving the image itself. EVEREST is the first one in literature to address the efficiency concerns for digital signatures in teleradiology. The performance evaluation study and experiments show the prospect of our method.

One other advantage is the communication efficiency since the receiver does not need to get the entire file to detect a tampering. Traditionally when a tampering has been detected, the whole image file needs to be retransmitted. On the other hand in our proposed method in case of tampering the receiver requires retransmission of only the tampered part to complete verification of the signature. Consequently if the PACS system is a heavily used one and the bandwidth resources are limited, our method would also help to improve the "availability" of medical images.

An alternate option and a more obvious trick for real-time efficiency is that the image is displayed immediately and the status of the digital signature will be displayed when available. The disadvantage of this option is that the receiver sees a disturbing status window that may easily be ignored. On the other hand our method provides real-time efficiency without degrading the security and user-friendliness of the system. In [64], the author has defined a "self-enforcing protocol" as the best type of cryptographic protocol because it is independent of the trustworthiness of people. We believe our method can be accepted as one such protocol which provides efficiency as well.

Throughout this chapter we have assumed that the PACS server is the one that signs the medical images. In [74], the authors presented an alternative solution where an embedded system performs the signing operation immediately after the image acquisition. We have observed that the architecture of PACS system illustrated in Figure V.1 is very appropriate for the so-called “threshold cryptography”. By definition in threshold signatures at least two party should participate to complete the signing operation. This property improves the security because if an attacker steals one part of the private key, he will not be able to forge signatures. As a result we think that implementing threshold signatures in a PACS environment is a promising future work.

Another promising future work is adapting our method to encryption. In the entire time interval the downloading of an encrypted image takes place, the CPU of the receiver machine is again unused. Therefore the real-time performance of confidential medical image transfer is highly improved if we can perform most of the decryption computation before we finish receiving the image file.

Lastly, this method would also be applicable in other applications where big files are required to be verified efficiently. The method EVEREST proposed in this chapter has some unique features that will make it an exclusive choice for some applications while excluding others.

CHAPTER VI

CONCLUSIONS AND FUTURE WORKS

I like the dreams of the future better than the history of the past.

- Thomas Jefferson

Learn from yesterday, live for today, hope for tomorrow. The important thing is to not stop questioning.

- Albert Einstein

In this last chapter, I would like to provide the concluding remarks in Q&A form. Afterwards, the extensive list of future works is provided especially for those who want to make research on these topics.

Q: In this thesis, what is the fundamental question you have dealt with?

A: I have tried to find out the alternatives for improving the performance of authentication protocols, digital signatures and their applications in e-health.

Q: What are your findings?

First of all, we have observed that there are a bunch of authentication protocols proposed in the literature ranging from primitive Unix passwords to very advanced zero-knowledge proofs. Choosing one of them really depends on the context i.e., requirements and threats. For instance one-time passwords can be an efficient entity authentication alternative for those who only worry about so-called passive attacks where the attacker only listens the network and tries to eavesdrop the password.

Secondly, like all other engineering disciplines, “design” is an important subject in security engineering. As a matter of fact, designing a more efficient authentication protocol or a digital signature scheme is more challenging because of security considerations but a careful design might result in efficiency we are looking for without degrading the security level.

Third and the last approach to have a better performance is based on some clever tricks in the implementation phase. For instance using a faster modular exponentiation algorithm for the cryptosystem RSA might be a typical example. However we come up with a much simpler and a natural way to improve the real-time efficiency of digital signatures in teleradiology.

In this study, not only the third one, but all of the three alternatives are demonstrated by original examples.

Q: What do you mean by original examples?

A: As far as one-time passwords are of concern, we first suggest a new construction called “signature chain” as an alternative to “hash chain” that is widely used in one-time passwords and then based on the signature chain idea, we propose a one-time password protocol called SCOTP which improves the flexibility and security of previous approaches. The disadvantage of signature chain is the larger verification time with respect to hash chain based approaches.

The second and the most important contribution of this dissertation we think is a new design of a server assisted signature protocol. This new design employs one-time signatures and improves the real-time and round efficiency of getting assist from a “verifiable” server. By the help of our new protocol called SAOTS, a “proxy” server is no longer the only option for those are uncapable of generating public key signatures.

Third one has already been mentioned. EVEREST, the third method we propose in this thesis is designed for verification of signatures on large medical images. This approach is again more delay-efficient than traditional methods.

From another view, in proposing these three methods, we have actually answered the following three questions:

1. One-time passwords offer a viable alternative but have some deficiencies with respect to flexibility and security they provide. How can we improve their security and flexibility?

2. In designing a new server assisted signature protocol, how can it become feasible to use one-time signatures which do not based on a difficult mathematical problem and can be realized only by using very efficient one-way functions?

3. To verify digital signatures in teleradiology a considerable amount of time is spent to compute the hash of the image because the image size is huge

(tens of megabytes). On the other hand the CPU of the verifying machine is idle until the downloading of the entire image file is completed. Motivating by these facts, how can we come up with an idea that improves the real-time efficiency?

Q: OK, that looks nice. But I wonder if you have any additional contribution in this thesis?

A: Yes, we have also examined some of the sub-problems that we have encountered while doing the actual research work. I prefer to express these problems in question form again:

- In smartcard authentication there is a significant risk in locations where there is no smartcard reader available. What are the options to have a survivable authentication framework? And how can one-time passwords contribute to the survivability?
- Using the BAN logic, can we analyze the security of these one-time password protocols?
- In e-health, attribute certificates are very useful tools especially when the roles and authorizations dynamically change. How can we integrate attribute certificates to a server-assisted signature protocol?

Q: A few last words?

A: Yes, before the future works, I would say cryptography and network security contains very joyful and interesting topics and I hope the future works that will be provided next will help to those who are interested to work in this area.

I would try to provide the future works in a logical order therefore they are grouped again in a top-down approach:

Future Works in Requirement Analysis Phase:

- **New security requirements in pervasive computing:** One of the application domains that is in urgent need for the efficiency provided by our proposed methods is pervasing computing where for the sake of nearly ubiquitous information access, computers get smaller and occupy everywhere. However throughout this thesis, we do not take into account the new security requirements this new era of computing brings. For instance while designing SAOTS we have assumed that the parties involved can have an access to a server. Being capable of having an access to useful certification information is another assumption not only ours but also almost every traditional authentication protocols have made. It has been argued by a number of researchers that these assumption are no longer valid in the pervasive domain [75, 76, 77]. As a matter of fact in a conference [78] I have attended, it was surprising to see that this field is so young that most of the papers presented have introduced the requirements and problems rather than the solutions. Finding the required solutions will really look like a promising and challenging research subject. We highly recommend the interested readers to [78] and especially the paper about the requirements of pervasive computing in e-health [77].

- **Economics of Security:** In this thesis, we have repeatedly emphasized the importance of the requirement analysis phase for choosing an authentication protocol or a digital signature scheme. Actually thinking in terms of your security requirements is a big step but not enough.

In real life for instance you might really need a brand new car, but your financial resources could allow you to drive only a 10-year old Renault Spring. Therefore considering both your requirements and resources is essential. Nothing is different for security. "The economics of security is a hot and rapidly growing field of research [1]". Studying on applying the results of this new field to e-health systems would seem to give fruitful results.

Future Works for the Design Phase:

- **Secure signature schemes when quantum computers are realized:** Quantum computers use the concepts in quantum mechanics to improve the computational efficiency of classical computers. A "qubit" is a quantum-bit, a bit of information that can be both zero and one simultaneously. Thus, a qubit rather than a standard bit based computation can make calculations using both values at the same time. Similarly a "qubyte" is composed of eight qubits and can be all values from zero to 255 simultaneously. Extending this concept to multi-qubyte systems, it is seen that there is a potential for computational efficiency growing exponentially beyond anything possible with traditional computers.

  To realize practical quantum computers there are some practical limitations. But reserarchers believe that they will be available in the near future

[1] http://www.cl.cam.ac.uk/∼rja14/econsec.html, Last access: September 17, 2003.

and try to come up with algorithms to solve some complex problems difficult with classical computers. One of the famous quantum algorithms is due to Shor [79] that solves the factorization and discrete logarithm problem in polynomial time. Since the security of most public key signature algorithms (e.g., RSA [16] and DSA [8]) depend on the difficulty of these problems, Shor's invention implies that these signature algorithms become insecure when quantum computers will become available.

Hence the big challenge is to design a signature scheme which protects its security level even when quantum computers are used for the cryptanalysis. On the other hand the security of one-time signatures does depend on only a one-way function rather than the difficulty of a complex mathematical problem therefore the solution we seek might be already very close to us. However to make things even more complex I would say that designing provably secure one-way functions in case when quantum computation is to be used is also an open question that we do not know the answer yet.

As a conclusion, designing a signature scheme secure against quantum cryptanalysis is one of the really ambitious future work we present in this chapter.

- **Improving the efficiency of joint compression and hashing:** When a large message such as the concerned medical image file in Chapter V is to be transmitted, sometimes it is first compressed to reduce its size. Before the transmission takes place if it is also required to perform the signing for security reasons, the hash of the message should be computed as well. So to prepare the message for transmission, the delay overhead is equal to the sum of the time required to compress the message and the time to compute the hash of the message (if the time to sign the message is comparably small and therefore ignored). Then, the open problem is to design a joint compression (either a lossy or a lossless one) and hashing algorithm that can be named as “comprehashing” which satisfies the following inequality:

$$Delay(Compress\&Hash) << Delay(Compress) + Delay(Hash) \quad \text{(VI.1)}$$

In formulating the inequality above, we are inspired by [80] which has proposed and solved the same problem for joint signing and encryption.

- **Robust authentication techniques surviving compression:** In teleradiology if compression is to be employed, it is a best practice to do the compression before the signing. However in some situations it might be necessary to compress the message after it is signed (e.g., to serve also the receivers with more limited resources). If this is the case, to avoid the cost and complexity of re-signing, we would need a signature algorithm which remains valid after the compression is applied.

  In fact, since usually a one-way hash function is applied before the signing, a secure hash function which gives the same output for both the inputs of uncompressed and compressed data will be sufficient for our purposes.

One of the authentication techniques that survives the JPEG lossy compression is the work by C.Y. Lin [81]. In our survey we see that these techniques were usually proposed for lossy compression simply because it is the one mostly preferred in practice on the other hand as far as medical images are of concern, the situation is contraversial and lossless compression techniques are recommended. As a result we can say that a design of a robust (and efficient) authentication technique that survives the lossless compression will be worthful in e-health applications.

- **Server assisted verification of digital signatures:** As it is evident from Chapter IV, there are numerous proposals in the literature to increase the performance of digital signature generation. However in a similar fashion getting aid for the verification of the signature is not well-studied [2]. While it is possible to think about the suggested future work independently, considering the capabilities and limitations of SAOTS with respect to verification of signatures might also be good starting point.

- **An OTP Protocol with a stronger authentication goal:** In section III.9, we have seen that all of the current OTP protocols achieve a weak authentication goal that safeguards against only passive network attacks. A secure design of an OTP protocol which attains a stronger authentication goal while preserving the protection capability against the attacks on the client side would be very useful but at the same time could be not so

[2] As a matter of fact we are not aware of any previous study except the email message of P. Kasselman. Available at http://lists.oasis-open.org/archives/dss/200301/msg00010.html, Last access: September 17, 2003.

straight forward. This is because of the lack of trust on the client machine, one of the sides the protocol should execute in. The interested readers might find the recent work of Micali and Leyzin [3] stimulating.

- **Designing a one-time signature based special purpose digital signature primitive:** After Diffie and Hellman have proposed the concept of digital signatures [7], researchers discovered a large number of signature primitives with additional special properties. For instance "blind signatures" are proposed to generate a signature on a message without knowing what the message is [6]. While flexibility of public key operations allows us to design such signature schemes, it is really difficult to do it similarly with one-time signatures which depend on one-way functions.

- **Reducing the length of messages in SAOTS protocol:** We think that the only real weakness of the SAOTS protocol proposed in chapter IV is the longer length of messages exchanged. Obviously to reduce the length, SAOTS would benefit from the future research that shortens the length of one time signatures and/or one-time public keys. So the problem to investigate is really about the concept of one-time signatures.

Future Works for Implementation, Security Analysis and Performance Evaluation:

- **Threshold cryptography for PACS environment:** In the scenario of a typical cryptography application there is one sender and one receiver.

[3] S.Micali and L. Reyzin, "Physically Observable Cryptography", Cryptology ePrint Archive, Report 2003/120, Available at http://eprint.iacr.org, Last access: September 17, 2003, Last update: June 9, 2003.

"Threshold cryptography" allows one (either the sender or the receiver) to share the power of a cryptosystem [82]. For instance in a " (n, k) threshold signature scheme", the signing key is split up among n parties so that only any k out of n can sign a message. The advantages of threshold signatures are two-folds. First of all they are valuable tools to implement some real-world business transactions e.g., large bank transactions require two people to sign. Secondly, since the secret key is shared among a number of participants, an attacker is not able to break the system (e.g., generate a forged signature) if he has obtained only one of the shares.

We have observed that the PACS system (Figure V.1) is very appropriate to get benefit of the aforementioned advantages of threshold cryptography because there are already more than one party participating (the image acquisition modality and the PACS server) to respond to a receiver.

- **Adapting EVEREST for encryption and message authentication codes:** The fact that the CPU of the verifying machine is idle during the transmission of the medical image can be used not only to improve the performance of employing digital signatures but also other cryptographic tools as well. Although it seems to be not so tricky, this future work is again promising for obtaining real-time efficiency in secure medical imaging applications.

- **Filling the gap between formal methods and cryptography:** In applying formal methods, a cryptographic operation is generally represented as a black box which gives a certain output given a certain input e.g., The BAN logic [22]. However this abstraction turns out to be too simplistic. Instead, a formal technique which makes its abstractions with more proper justifications is to be modelled. Visit the web page [4] for information on combining cryptographic and formal proof techniques.

- **Power efficiency of authentication protocols:** In section I.4, we have mentioned the power limitation of pervasive devices. In section IV.1, we have argued that on-line/off-line signatures are not suitable for devices with restricted power resources. This restriction is more serious in some sensor networks where changing the battery of a sensor node is almost practically impossible. Therefore the maximum operation life of the sensor network depends on the battery life of sensors and the power efficiency of communication protocols employed.

  Today, in networking community yet another current hot topic is "sensor networks" and there are a lot of recent work on designing secure protocols suited to their special requirements. The appropriateness of authentication protocols to sensor network applications including the ones presented in this thesis really depends on their power efficiency. As a result, evaluating the energy performance of these protocols is required. Of course this will be more difficult than simply measuring the time delay of the protocol by a few lines of software code [83].

[4] Project web page, "linking formal methods and cryptography", Available at http://www.zurich.ibm.com/security/models/, Last access: September 17, 2003.

Lastly, widely employed WTLS (Wireless Transport Layer Security) protocol, the standard for secure wireless access to Internet services, is another candidate for such a performance evaluation study.

REFERENCES

[1] B. Schneier, *Secrets and Lies, Digital Security in a Networked World*, Wiley Computer Publishing, 2000.

[2] D. Gollmann, *Computer Security*, John Wiley and Sons, 1999.

[3] A. Menezes, P. Van Oorshot, and S. Vanstone, *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications, CRC Press, 1996.

[4] G. Eysenbach, "What is e-health?", *Journal of Medical Internet Research*, June 18; 3(2):e20, 2001.

[5] O.A. Rayis, "Total Security Management - A Paradigm for Developing Secure Information Systems", Ph.D. Thesis, Middle East Technical University, Computer Engineering Department, April 2000.

[6] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley and Sons, 2001.

[7] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Trans. Information Theory*, IT-22(6), pp.644-654, November 1976.

[8] National Institute of Standards and Technology (NIST), *FIPS Publication 186: Digital Signature Standard*, May 19, 1994.

[9] National Institute of Standards and Technology (NIST), *FIPS Publication 180: Secure Hash Standard (SHS)*, May 11, 1993.

[10] C. Kaufman, R. Perlman and M. Speciner, *Network Security, Private Communication in a Public World*, Prentice Hall Series, Second Edition, 2002.

[11] D. R. Stinson, *Cryptography: Theory and Practice*, CRC Press, Inc., Second Edition, Boca Raton, 2002.

[12] B.Schneier, *Applied Cryptography*, John Wiley and Sons, 1996.

[13] S. Burnett and S. Paine, *RSA Security's Official Guide to Cryptography*, McGraw-Hill, 2001.

[14] National Institute of Standards and Technology (NIST), *FIPS Publication 46-2: Data Encryption Standard*, December 30, 1993.

[15] National Institute of Standards and Technology (NIST), *FIPS Publication 197: Advanced Encryption Standard*, November 26, 2001.

[16] R. L. Rivest, A. Shamir and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, 21(2), 1978, 120-126.

[17] R. L. Rivest, "The MD5 message-digest algorithm", *RFC 1321*, April 1992.

[18] P.R. Zimmermann, *The Official PGP User's Guide*, MIT Press, Cambridge, Mass., 1995.

[19] L. Lamport, "Password authentication with insecure communication", *Communications of the ACM*, 24(11): 770-772, November 1981.

[20] R. L. Rivest and A. Shamir, "Payword and Micromint: Two simple micropayment schemes", *CryptoBytes*, volume 2, number 1 (RSA Laboratories, Spring 1996).

[21] N. Asokan, G. Tsudik and M. Waidner, "Server-Supported Signatures", *Journal of Computer Security*, November 1997.

[22] M. Burrows, M. Abadi and R. Needham, "A logic of authentication", *ACM Transactions on Computer Systems*, 8(1):18-36, February 1990.

[23] CCITT X509 and ISO 9594-8, "The Directory - Authentication Framework", *CCITT Blue Book*, Geneva, March 1988.

[24] C. Meadows, "Formal verification of cryptography protocols", *Proceedings of ASIACRYPT 1994*, 1994.

[25] M. Burrows, M. Abadi and R. Needham, "A logic of authentication", *In Proceedings of the Royal Society of London*, volume 426, 1989, pp 233-271.

[26] R. J. Anderson and M. G. Kuhn, "Tamper Resistance - a Cautionary Note", *The Second USENIX Workshop on Electronic Commerce Proceedings*, Oakland, California, November 18-21, 1996, pp 1-11.

[27] T. Ylonen, "SSH - Secure login connections over the Internet", *The Sixth USENIX Security Symposium*, pp 37-42, July 1996.

[28]]B. C. Neuman and T. Tso, "Kerberos: An Authentication Service for Computer Networks", *IEEE Communications*, 32(9):33-38, September 1994.

[29] N. Haller, "The S/Key One-Time Password System", *Proceedings of the Symposium on Network and Distributed Systems Security*, Internet Society, San Diego, CA, February 1994.

[30] N.Haller, “The S/KEY One-time Password System”, *RFC 1760*, February 1995.

[31] D. de Waleffe and J. J. Quisquater, “Better login protocols for computer networks”, *In Proc. of ESORICS*, Toulouse (France), Oct. 1990. (Published as a chapter of a book edited by J. Vandewalle, Springer-Verlag).

[32] A. D. Rubin, “Independent One-Time Passwords”, *USENIX Journal of Computer Systems*, February, 1996.

[33] K. Bicakci and N. Baykal, “Infinite Length Hash Chains and Their Applications”, *Proceedings of IEEE 11th International Workshops on Enabling Technologies (WETICE 2002)*, June 10-12, 2002, Pittsburgh, USA.

[34] K. Bicakci and N. Baykal, “Improving the Security and Flexibility of One-time Passwords by Signature Chains”, *accepted to Turkish Journal of Electrical Engineering and Computer Sciences*

[35] National Institute of Standards and Technology (NIST), *FIPS Publication 190: Guideline for the Use of Advanced Technology Alternatives* ,September, 1994.

[36] G. J. Ahn and D. Shin, “Towards Scalable Authentication in Health Services”’, *Proceedings of IEEE 11th International Workshops on Enabling Technologies (WETICE 2002)*, June 10-12, 2002, Pittsburgh, USA.

[37] ”Digital Rights and Wrongs”, *The Economist*, July 1999.

[38] R. Anderson and R. Needham, “Robustness principles for public key protocols”,*Proceedings of CRYPT 1995*, 1995.

[39] K. Bicakci and N. Baykal, “Survivable Authentication for Health Information Systems”, *Proceedings of 2003 AMIA Symposium on Computer Applications in Medical Care*, November 8-12, 2003, poster paper.

[40] R. Rivest, “The MD4 Message-Digest Algorithm”, *RFC 1320*, April 1992.

[41] N. Courtois, L. Goubin and J. Patarin, “Quartz, 128-bit long digital signatures”, *In Cryptographers’ Track RSA Conference 2001*, San Francisco, 8-12 April 2001, LNCS 2020, Springer-Verlag.

[42] N. Courtois, M. Finiasz and N. Sendrier, “How to Achieve a McEliece-based Digital Signature Scheme”, *Asiacrypt 2001*, Gold Coast, Australia, 9-13 December 2001, LNCS 2248, Springer-Verlag.

[43] K. Bicakci and N. Baykal, “Wireless and Mobile Security”, *Proceedings of 5th KES International Conference, KES’2001*, September 2001, Japan.

[44] D. W. Carman, P. S. Kruus and B. J. Matt, “Constraints and approaches for distributed sensor network security”, *NAI Labs Technical Report 00-010.*

[45] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ubiquitous Computing", *First Security and Privacy supplement to IEEE Computer*, April 2002.

[46] S. Even, O. Goldreich and Silvio Micali, "On-line/off-line digital signatures", *In Proceedings of CRYPTO 1989*, LNCS 435, Springer-Verlag, 1990.

[47] A. Shamir and Yael Tauman, "Improved on-line/off-line signature schemes", In Proceedings of CRYPTO 2001, LNCS 2139, Springer-Verlag, 2001.

[48] L. Lamport, "Constructing digital signatures from a one-way function", *Technical Report CSL-98, SRI International*, October 1979.

[49] K. Bicakci, G. Tsudik, B. Tung, "How to construct optimal one-time signatures", *Accepted to Computer Networks, Elsevier Science journal.*

[50] R. C. Merkle, "A digital signature based on a conventional encryption function",*In Proceedings of CRYPTO 87*, LNCS 293, Springer-Verlag, 1988.

[51] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas and R. Rivest, "Proxy-Based Security Protocols in Networked Mobile Devices", *Proceedings of the 17th ACM Symposium on Applied Computing, (Security Track)*, March 2002.

[52] M. Jakobsson and S. Wetzel, "Secure Server-Aided Signature Generation", *Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001)*, LNCS 1992, Springer, 2001.

[53] X. Ding, D. Mazzocchi and G. Tsudik, "Experimenting with Server-Aided Signatures", *2002 Network and Distributed Systems Security Symposium (NDSS'02)*, February 2002.

[54] D. Boneh, X. Ding, G. Tsudik and B. Wong, "Instantaneous revocation of security capabilities", *In Proceedings of USENIX Security Symposium 2001*, Aug. 2001.

[55] K. Bicakci and N. Baykal, "Design and Performance Evaluation of a Flexible and Efficient Server Assisted Signature Protocol", *In Proceedings of IEEE 8th Symposium on Computers and Communications, ISCC 2003*, June 30-July 3, 2003, Antalya, Turkey.

[56] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams", *In Proceedings of CRYPTO 1997*, LNCS 1294, Springer-Verlag, 1997.

[57] A. S. Tanenbaum, "Computer Networks", 3rd edition, Prentice Hall, 1996.

[58] K. Bicakci and N. Baykal, "SAOTS: A New Efficient Server Assisted Signature Scheme for Pervasive Computing", *In Proceedings of 1st International Conference on Security in Pervasive Computing, SPC 2003*, To appear as LNCS 2802, March 2003, Germany.

[59] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, "Internet public key infrastructure online certificate status protocol - OCSP", *RFC 2560*, June 1999.

[60] ITU-T Rec. X.509, "The Directory: Authentication Framework", *ISO/IEC 9594-8*, 2001.

[61] E. Dawson, J. Lopez, J. A. Montenegro and E. Okamoto, "A New Design of Privilege Management Infrastructure for Organizations Using Outsourced PKI", *Information Security, 5th International Conference, ISC 2002*, Sao Paulo, Brazil, September 30 - October 2, 2002, LNCS 2433, Springer, 2002.

[62] D. Boneh, X.Ding and G. Tsudik, "Fine-grained control of security capabilities", *ACM Transactions on Internet Technology*, to appear.

[63] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol", *ACM Conference on Computer and Communications Security* 2001.

[64] S. T.C. Wong, "A Cryptologic Based Trust Center for Medical Images", *Journal of American Medical Informatics Association (JAMIA)*, 3:410-421, 1996.

[65] H.K. Huang, "Picture Archiving and communication systems: principles and aplications", New York: Wiley, 1999.

[66] F.Cao, H.K. Huang, and X.Q. Zhou, "Medical Image Security in a HIPAA Mandated PACS Environment", *Computerized Medical Imaging and Graphics*, Volume 27, Issue 2-3, page 185-196, 2003.

[67] X.Q. Zhou, H.K. Huang and S.L. Lou, "Authenticity and Integrity of Digital Mammography Images", *IEEE Transaction on Medical Imaging*, Volume 20, No. 8, August 2001.

[68] National Electrical Manufacturers Association (NEMA), Rossyln, VA, "Digital Imaging and Communications in Medicine (DICOM), Part 15: Security Profiles", PS 3.15-2001.

[69] K. Bicakci and N. Baykal, "EVEREST: An Efficient Method for Verification of Digital Signatures in Real-Time Teleradiology", In submission to 2004 MEDINFO Symposium September 7-11, 2004, San Francisco, USA.

[70] T. Schweitzer, U. Engelmann, A. Schroter, E. Boraelv, H.P. Meinzer, "Teleradiology on a Personal Digital Assistant", *Second Conference on Mobile Computing in Medicine*, Workshop of the Project Group MocoMed, 11.04.2002, Heidelberg.

[71] R. Norcen, M. Podesser, A. Pommer, H. -P. Schmidt and A. Uhl, "Confidential Storage and Transmission of Medical Image Data", *Computers in Biology and Medicine*, Volume 33, Issue 3, Pages 277-292, May 2003.

[72] P.C. Cosman, R.M. Gray, R.A. Olshen, “Evaluating quality of compressed medical images: SNR, subjective rating, and diagnostic accuracy”, *Proc. IEEE*, 82(6), 1994.

[73] G. Hachez and J.J. Quisquater, “Which directions for asymmetric watermarking?”, *Proceedings of XI European Signal Processing Conference (EUSIPCO 2002)*, Toulouse, France, September 2002.

[74] M. Kroll, B. Schutze, T. Geisbe, H.-G. Lipinski, D.H.W. Gronemeyer, and T.J. Filler, “Embedded Systems for Signing Medical Images Using the DICOM standard”, *International Congress Series*, Volume 1256, page 849-854, June 2003.

[75] S. Criese, M. Goldsmith, B. Roscoe and I. Zakiuddin, “Authentication for Pervasive Computing”, *In Proceedings of 1st International Conference on Security in Pervasive Computing, SPC 2003*, To appear as LNCS 2802, March 2003, Germany.

[76] L.. Bussard and Y. Roudier, “Authentication in Ubiquitous Computing”, *UBICOMP 2002, Workshop on Security in Ubiquitous Computing*, September 2002.

[77] J. Bohn, F. Gartner and H. Vogt, “Dependability Issues of Pervasive Computing in a Healthcare Environment”, *In Proceedings of 1st International Conference on Security in Pervasive Computing, SPC 2003*, To appear as LNCS 2802, March 2003, Germany.

[78] Editors: D. Hutter, D. Muller, M. Ullmann, W. Stephan, *Proceedings of 1st International Conference on Security in Pervasive Computing, SPC 2003*, To appear as LNCS 2802, March 2003, Germany.

[79] P. W. Shor, “Algorithms for quantum computation”, *in Proc. 35th Annual Symposium on the Foundations of Computer Science*, edited by S. Goldwasser, IEEE Computer Society Press, Los Alamitos, California, 1994.

[80] Y. Zheng, “Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption)”, *in Proc. of CRYPTO 1997*, LNCS 1294, Springer, 1997.

[81] C.Y. Lin and S.-F. Chang, ”A robust image authentication method surviving JPEG lossy compression”, *SPIE Security and Watermarking of Multimedia Content*, Vol.3312, pp.296-307, 1998.

[82] A. Shamir, “How to share a secret”, *Communications of ACM*, Vol. 22, November 1979.

[83] A. Levi, personal communication.

APPENDIX A

How to Use DSA to Construct a Signature Chain

We first want to remind you how DSA works. In DSA [8], we first generate a random secret number k. The signature for a message x and the secret number k is the pair (γ, δ).

$$sig(x, k) = (\gamma, \delta) \tag{A.1}$$

where

$$\gamma = (\alpha^k \bmod p) \bmod q \tag{A.2}$$

and

$$\delta = (x + \alpha\gamma)k^{-1} \bmod q \tag{A.3}$$

In the above formulas, p, q, α as well as β (calculated as in equation A.4) are public and a is the private key.

$$\beta = \alpha^a \bmod p \tag{A.4}$$

For a 160 bit message x, both γ and δ are also 160 bits in length resulting in a total signature length of 320 bits. Verification will work as follows:

$$e_1 = x\delta^{-1} \bmod q \; and \; e_2 = \gamma\delta^{-1} \bmod q \tag{A.5}$$

$$ver(x, y, \delta) = true \; if \; (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma \tag{A.6}$$

We left the details of DSA to cryptography textbooks (e.g., [11, 3]) and would like to give you the information on constructing a signature chain which safeguards against attacks of computing previous OTPs from the later OTPs.

In constructing the SC by using DSA, the user first computes the signature (γ_1, δ_1) for the seed s (a 160-bit random number) as the first OTP to be used. Since to generate the second OTP, we need to sign a 160 bit number, we first apply "exclusive-or (+)" operation as follows:

$$x_2 = \gamma_1 \oplus \delta_1 \tag{A.7}$$

This computation is also done (offline) on the server side in order to verify the second OTP which can be generated by applying the following formula:

$$P_2 = sig(x_2, k_2) = (\gamma_2, \delta_2) \tag{A.8}$$

Where k_2 should be a newly generated random secret number. The procedure shown in equations A.7 and A.8 repeats to generate further OTPs.

APPENDIX B

Encoding a Message for One-Time Signature

Given a vector R of m random numbers, and a V of subsets each containing p of those numbers, we have shown that we can sign any one of $C(m,p)$ distinct messages. In this appendix, we describe the mapping M between messages and the elements of V, and demonstrate how to compute them efficiently.

Assume that the domain of M is composed of 2^b messages, and we have a way of representing each of a message as a $b - bit$ integer k. Let any subset S in V be expressed as $R_{a_1}, R_{a_2}, R_{a_3}, \ldots, R_{a_p}$. Arrange the subsets in V such that their indices a_i are in ascending order. For example, for $m = 4$, $p = 2$, the subsets are ordered

$$\{R_1, R_2\}, \{R_1, R_3\}, \{R_1, R_4\}, \{R_2, R_3\}, \{R_2, R_4\}, \{R_3, R_4\}$$

Then the mapping $M(S, \mathcal{V})$ of each subset S is defined as the integer position of S in this list representation of $\mathcal{V}$. For example, in the above case, $M(\{R_1, R_3\}, \mathcal{V}) = 2$ and $M(\{R_3, R_4\}, \mathcal{V}) = 6$. In general, for any n and

p, the mapping of any subset $S = \{R_{a_1}, R_{a_2}, \ldots, R_{a_p}\}$, where $a_0 = 0$ and $a_1 < a_2 < \cdots < a_p$ is given by:

$$M(S, \mathcal{V}) = 1 + \sum_{i=1}^{p} \sum_{j=n-a_i+1}^{n-a_{i-1}-1} \binom{j}{p-i} \tag{B.1}$$

Note that in order to compute the mapping for any subset, for a given n and p, we need only compute the binomial coefficients $C(j, p-i)$ for i from 1 to p, j from $p+1-i$ to $n-i$. Thus, each mapping requires $n-p-(n-a_p) = a_p - p$ additions.

Similarly, the mapping $M^{-1}(m, \mathcal{V})$ of a message represented by the integer m can be computed by subtracting binomial coefficients until zero is reached. This requires $a_p - p$ additions and comparisons. Pseudocode to do this conversion is as follows:

```
$m_0 = m$ /* copy message to temporary value */
$q = 1$
for $i = 1$ to $p$ do begin
while $m_0 > C(n-q, p-i)$ do begin
$m_0 := m_0 - C(n-q, p-i)$
$q := q + 1$
end /* while */
$a_i := q$
$q := q + 1$
end /* for */
```

To put things in perspective, consider that a single SHS hash computation requires approximately 500 arithmetic operations. Thus, our mapping (in both directions) costs less than one SHS hash.

APPENDIX C

How To Measure CPU Time in JAVA

In Java language, to evaluate the performance, a comfortable approach is to call “System.currentTimeMillis()” before and after the code we have written. The trouble with this approach is that it may give a result more than the real execution time of our code because the time used by other processes or time spent for I/O is also included in the time we have measured.

Alternatively to have a more accurate timing measurement, we can query the CPU time spent by the process our code is executing in. The problem is that unlike programming languages C and Pascal, programmatically querying for CPU usage is impossible by using only pure Java.

Fortunately, in our research for a solution, we have seen there are (at least) two different ways to measure the CPU time of programs written in Java [1] [2]:

[1] V. Roubtsov, “Profiling CPU usage from within a Java application”, Available at http://www.javaworld.com/javaqa/2002-11/01-qa-1108-cpu_p.html, Last access: September 17, 2003, Last update: November 11, 2002.

[2] J. Gortz, “Java Tip 92: Use the JVM Profiler Interface for accurate timing”, Available at http://www.javaworld.com/javatips/jw-javatip92.html, Last access: September 17, 2003.

- Implement the function measuring the CPU usage in C language and integrate it with the Java application via Java Native Interface (JNI) [3].
- Use the new API called Java Virtual Machine Profiler Interface (JVMPI) introduced by Java 2 that allows access to the necessary timing information.

I have preferred to use the second approach in our performance evaluation study because it seems to be the easier one. In brief, I would like to introduce this approach here:

The JVMPI provides a function called “GetCurrentThreadCpuTime()” which returns the CPU time in nanoseconds for the current Java thread. In [**?**], a simple profile agent written in C++ language is provided to access this information. This agent may be called from a Java program if in executing the Java class you tell the JVM to use the DLL (dynamic link library) that loads the agent:

$$java\ -X''nameofdll''\ classname$$

Then to measure the CPU time, inside your Java code you simply call the function getCurrentThreadCpuTime() instead of System.currentTimeMillis().

[3] The JNI allows Java code that runs within a Java Virtual Machine (VM) to operate with applications and libraries written in other languages, such as C, C++, and assembly.

APPENDIX D

CURRICULUM VITAE

Kemal Bıçakcı received the B.S. degree in electronics engineering from Hacettepe University, Turkey in 1996 and the M.S. degree in electrical engineering (computer networks) from University of Southern California, USA in 1999. Between January 1998 and March 2000, he worked on various security projects as a graduate research assistant in USC Information Sciences Institute.

He is now in Informatics Institute, Middle East Technical University, Turkey. His research interests include network security, e-health, health information security and applied cryptography.