HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Telecommunications Software and Multimedia Laboratory

Tomi Mickelsson

Evaluation of secure remote access software technologies

Master's Thesis
September 2nd, 2001

Supervisor	Professor Hannu H. Kari
Instructor	M.Sc. Antti Huima

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Title:	Evaluation of secure remote access software technologies
Author:	Tomi Mickelsson
Date:	September 2nd, 2001 **Pages:** 8 + 82
Department:	Department of Electrical and Communications Engineering
Professorship::	Tik-109 Telecommunications software
Supervisor:	Professor Hannu H. Kari
Instructor:	M.Sc. Antti Huima

Internet has become the essential medium for electronic data communication. The amount of confidential data exchanged over the Internet has increased dramatically over the past few years. However, the Internet is a dangerous place for the data if no security services are deployed. The protocol packets are vulnerable to unauthorized monitoring or even modification along the route from source to destination. The necessary security services which enable secure remote access between two computers are confidentiality, integrity and authentication. These services are provided by cryptographic algorithms and techniques which are briefly described in the thesis.

Various technologies exist for enabling secure remote access over an insecure network such as the Internet. This thesis evaluates three popular, widely used, and standards-based secure remote access software technologies. These are SSH — Secure Shell, TLS — Transport Layer Security, and IPSec — Internet Protocol Security. The thesis presents an overview of each technology. The main objective of the thesis is to evaluate and analyze the advantages and weaknesses of the three technologies and their field of suitability. The evaluation is performed through three different scenarios that present common remote access tasks from the real-life, such as file transfer and email access.

All of the three technologies discussed provide strong security by relying on standard and proven cryptographic techniques. However, there are great differences in the nature of the technologies. SSH is a mature solution for providing secure terminal and file transfer sessions in heterogenous network environments. It interoperates well and provides flexible user authentication. On the other hand, it lacks ease of use to function as a generic security solution. Applications with integrated TLS provide ease of use and transparent security, which are the main benefits of TLS. IPSec is the most versatile technology and suits many remote access tasks. IPSec is superior technology for providing virtual private networks. One of the weaknesses of IPSec is its complexity.

Each technology has its advantages and weaknesses. Currently none of the technologies alone can satisfy all secure remote access needs. IPSec is the most comprehensive security technology and has the potential to replace other technologies in the future. However, IPSec does not come without problems; there are many technical challenges that still require development effort. In the near future, the usage of all three technologies will increase, and all of them will be actively developed by the open security community.

Keywords:	Internet, TCP/IP, SSH, Secure Shell, SSL, TLS, IPSec, Security, Remote access, Cryptography, Evaluation, Attack
Language:	English

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Työn nimi:	Tietoturvaohjelmistojen evaluointi	
Tekijä:	Tomi Mickelsson	
Päivämäärä:	2. syyskuuta 2001	**Pages:** 8 + 82
Osasto:	Sähkö- ja tietoliikennetekniikan osasto	
Professuuri::	Tik-109 Tietoliikenneohjelmistot	
Valvoja:	Professori Hannu H. Kari	
Ohjaaja:	DI Antti Huima	

Internetistä on tullut merkittävä media elektroniseen tiedonsiirtoon. Internetissä liikennöidyn luottamuksellisen tiedon määrä on kasvanut räjähdysmäisesti viimeisien vuosien aikana. Internetissä on kuitenkin monia uhkia luottamukselliselle tiedolle mikäli viestinnässä ei käytetä tietoturvapalveluja. On mahdollista, että vieras osapuoli lukee tai jopa muuttaa protokollaviestejä niiden kulkiessa Internetin kautta lähettäjältä vastaanottajalle. Turvallinen yhteys kahden tietokoneen välillä vaatii kolme tietoturvapalvelua, jotka ovat luottamuksellisuus, eheys ja tunnistus. Nämä palvelut rakentuvat kryptografisista algoritmeista ja menetelmistä, jotka esitellään lyhyesti tässä työssä.

Internetissä suoritettavien etäyhteyksien suojaamiseksi on olemassa useita eri teknologioita. Tässä työssä tarkastellaan kolmea suosittua, laajalti käytettyä ja standardeihin perustuvaa tietoturvaohjelmistoteknologiaa. Nämä ovat SSH — Secure Shell, TLS — Transport Layer Security, ja IPSec — Internet Protocol Security. Teknologioiden tärkeimmät ominaisuudet esitellään työssä. Työn tärkein tavoite on tarkastella ja analysoida kunkin teknologian edut ja haitat sekä niiden soveltuvuus. Tarkastelu suoritetaan kolmen erilaisen skenaarion avulla, joissa kussakin määritellään tyypillisiä etäkäyttötapauksia, kuten tiedostonsiirto ja sähköposti.

Kaikki kolme teknologiaa tarjoavat turvallisen etäyhteyden käyttämällä yleisesti hyväksyttyjä ja vahvaksi todistettuja kryptografisia menetelmiä. Teknologioissa on kuitenkin suuria eroja. SSH soveltuu mainiosti terminaali- ja tiedostonsiirtoyhteyksien suojaamiseen heterogeenisissä tietoverkoissa. SSH:n yhteensopivuus on hyvä ja käyttäjä voidaan tunnistaa monipuolisesti. SSH ei kuitenkaan täysin sovellu yleiseksi tietoturvaratkaisuksi koska sen käyttö vaatii harjaannusta. TLS tarjoaa sovelluksiin helppokäyttöisyyttä ja läpinäkyvää tietoturvaa, jotka ovat TLS:n tärkeimmät edut. IPSec on kaikkein monipuolisin teknologia ja soveltuu monenlaisiin etäyhteyksiin. Toisaalta IPSec on myös kaikkein monimutkaisin teknologia.

Kaikissa tietoturvateknologioissa on etunsa ja haittansa. Tällä hetkellä mikään niistä ei yksinään pysty tarjoamaan kaiken kattavaa tietoturvaratkaisua. IPSec on kattavin teknologia, ja mahdollisesti syrjäyttää muut teknologiat tulevaisuudessa. Tässäkin teknologiassa on kuitenkin suuria teknisiä haasteita, joiden ratkaiseminen vaatii vielä paljon kehitystyötä. Lähitulevaisuudessa kaikkien kolmen teknologian käyttö tulee kasvamaan edelleen ja jokaista tullaan kehittämään aktiivisesti.

Avainsanat:	Internet, TCP/IP, SSH, Secure Shell, SSL, TLS, IPSec, Tietoturva, Etäyhteys, Kryptografia, Hyökkäys
Kieli:	englanti

Acknowledgments

I wrote this Master's thesis while working at SSH Communications Security Corp. I thank the company for giving me the time to work on this thesis despite tight project schedules. I am grateful for the company and the people for giving me the opportunity to graduate.

I thank my instructor M.Sc Antti Huima, who provided invaluable guidance especially in the beginning of the thesis in formulating the structure and contents of the thesis. The discussions with him were enlightening. He helped make sure I was heading in the correct direction during my work.

I thank my supervisor, professor Hannu H. Kari, for overseeing my work. He gave me advice in proper formulation of the problem and in defining the criteria and objectives of the thesis.

Big thanks to the review team, Sami Lehtinen, Vesa Suontama, Petri Säkkinen and Stephanie Thomas, for reading through the draft of the thesis and providing me good feedback and comments. Special thanks to Stephanie for dedicating her time to proofread and refine my English.

Finally, I wish to thank all my colleagues at SSH. The discussions with you all helped me to proceed with my thesis. You often provided me information that was not available elsewhere.

Yes, Mom and Dad, I am finally going to graduate. Thanks for having faith in me.

Espoo, Finland, August 26th, 2001

Tomi Mickelsson

Contents

List of Figures

Chapter 1

Introduction

Today, the largest data network in the world is the Internet, the network of networks. The power of the Internet stems from the fact that it is public. This supernetwork is open for all: organizations can connect their networks to it, and anybody can access all kinds of services from almost anywhere in the world. There are tens of thousands of interconnected networks and vast amount of information and services available. The Internet provides great connectivity for organizations and users.

One of the biggest weaknesses of the Internet is the lack of native security mechanisms. The Internet does not provide protection for the transmitted data. Data is vulnerable to unauthorized monitoring and even modification while in transit in the network.

Security was not a design goal at the time the Internet and its communication protocols were drafted over twenty years ago, but it is an essential requirement today. While the Internet started in the academic world, most of the usage today is contributed by companies conducting business over the network, and an increasing amount of confidential data is being transmitted over the Internet. The Internet has been commercialized, creating a strong demand for security mechanisms.

1.1 Motivation

This thesis is about the evaluation of software technologies that enable secure remote access over an insecure network such as the Internet. The thesis explains the basic building blocks of security and how they are used in technologies to provide secure remote access.

Understanding the basic security mechanisms and the operation of the technologies that use these mechanisms is important not only for security experts but for all who communicate confidential data over the Internet. Everybody should know what dangers the Internet poses to their data and how the communication can be best secured to prevent unauthorized access to it.

1.2 Technologies

There are many technical solutions for providing secure remote access over the Internet. In this thesis, the scope of the evaluation is limited to three very popular, standards-based, and widely used secure remote access software technologies.

The three evaluated technologies are: SSH — Secure Shell, TLS — Transport Layer Security, and IPSec — Internet Protocol Security.

SSH is an application that provides secure terminal and file transfer services. TLS is a generic security protocol for providing a secure transport for any TCP based applicaton protocol. IPSec is a technology for securing not just single application protocols but the whole network. The technologies operate at different layers of the networking software architecture.

1.3 Thesis objective

This thesis has three objectives. The main objective is to evaluate the advantages and weaknesses of each technology and discover their suitability to various remote access tasks. The evaluation is performed through three practical scenarios that define remote access tasks that are typical in real life.

The second objective of the thesis is to analyze the pros and cons of providing security in different layers of the networking software architecture.

The third objective is to suggest further improvements to one of the selected technologies, Secure Shell.

1.4 Thesis structure

Chapter 2, "Secure remote access", explains the concept of remote access and defines the necessary security services that must be available in order to have secure remote access. It also describes some of the most common threats and attacks in the Internet.

Chapter 3, "Cryptography basics", explains the basic building blocks of security. It provides a crash course about the basic cryptographical techniques that the secure remote access technologies are built on.

Chapter 4, "Technology overview", provides a technical overview of the three popular secure remote access technologies that have been selected for evaluation in this thesis.

Chapter 5, "Technology evaluation", contains the evaluation of the three technologies. It first defines the evaluation criteria and then evaluates the suitability of each technology in three scenarios.

Chapter 6, "Analysis", provides the results of the evaluation. An analysis is presented about how each technology fills the criteria, and what the pros and cons of the technologies are. A few improvements are suggested for Secure Shell. Finally, an estimate is presented about the future of the technologies.

Chapter 7, "Conclusions", summarizes the contents of the thesis and provides a short conclusive description about each technology.

Chapter 2

Secure remote access

Many kinds of remote access tasks are performed over the Internet. In some cases, the transmitted data has no privacy requirements, such as when a user accesses a public WWW service. However, data privacy is required in many cases. An increasing number of remote access tasks involve the exchange of confidential data over the Internet. A typical remote access task where privacy of the data is highly desired is an online bank providing financial services for the customers over the Internet.

The core communication protocols used in the Internet do not natively provide privacy for data. Security services are thus deployed to protect the transmitted data from monitoring and modification by unauthorized parties. Security services eliminate many threats that exist in the Internet.

2.1 Concept of remote access

Remote access is an operation where a service is accessed remotely over a network. The two communication parties are online and active at the same time, exchanging messages over the network. Typically, one party is a user using a client application and the other party is a server application providing services to the clients.

There are various kinds of remote access scenarios in the Internet. Users want to exchange emails or files in privacy. A shop wishes to provide a service online, and the customers want to be sure that it is safe to enter credit card information for purchase. An administrator needs to remotely administrate a router in a distant network. A travelling employee needs to access company private services and confidential data from the road with a laptop using a dial-up connection. Companies often need to exchange confidential data securely. A large company desires to connect its' subsidiares networks into their headquarters' network.

Today, an even more in the future, a very typical remote access scenario is a mobile communication device accessing a service remotely over the Internet. The world is becoming increasingly mobile: the number of mobile devices is estimated to grow at a much faster pace than the number of computers that are fixed in the network. Common mobile devices today are cellular phones, personal digital assistants, and laptop computers, but in a few years we may have our wallets, cameras or cars communicating wirelessly over the Internet. Security will be even more important when these devices become Internet-aware.

There is also a clear trend among companies to replace their private networks with Internet access. Traditionally, companies have achieved secure communication by managing their own private networks that they have leased from the telephone operator. A private network provides adequate security for the data since the access to the network is restricted and closed from the public. One of the main downsides of a private network is cost. It is expensive to rent and maintain dedicated lines.

Private lines were the solution in the past, but now it is becoming the Internet. The transition towards Internet-based communication is made possible by using technologies that enable creating private tunnels over a public network. Companies can enjoy private communication while still maintaining the cost-effectiveness and global coverage of the Internet.

The Internet has great momentum, and its technologies are under active development. The Internet may well provide the means for being a universal medium in the future for most communication, mobile or through the wire. There will be many different kinds of devices and numerous services accessible remotely. However, to fully utilize the power of the Internet, it is essential that trustworthy security services are available to eliminate the many threats that exist in the Internet.

2.2 Threats and attacks

The Internet is a dangerous place for data. *IP, Internet Protocol,* the communicaton "language" used by the computers in the Internet, has no inherent security mechanisms. IP provides no privacy for the content or the IP-addresses. The route taken by the data packets from the source to the destination can not be set and varies from time to time. Multiple subnetworks may exist along the route, which presents many possible locations for a malicious user to see or alter the data packets.

There are many different threats in the digital world, but here I will focus on threats against remote access. Computer viruses, worms, email bombs, troijan horse programs or many forms of physical threats are not discussed though they represent significant threats as well.

A simple but realistic model of a remote access scenario is displayed in Figure 2.1. A user wishes to communicate with a service over the Internet. Somewhere in the middle of the network exists an attacker. All data goes through the attacker who can monitor and possibly modify the data at will. The attacker has many ways to attack.

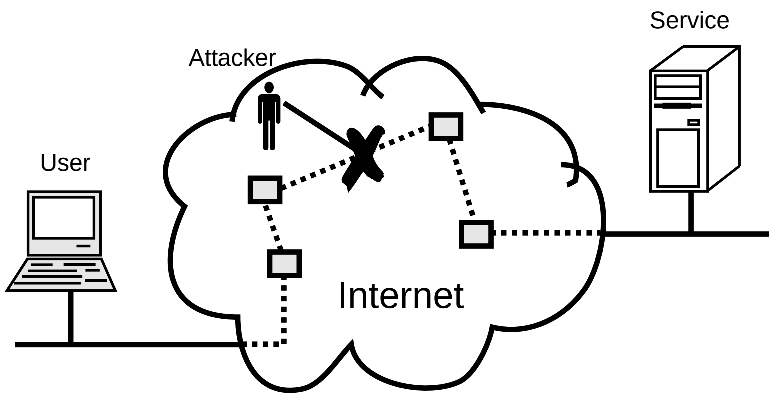

Figure 2.1: Simple model of an attack

Typical attacks against remote access without any security services are described below. A security attack is any action that compromises the security of the data. Attacks are usually divided into two categories: *passive attacks* and *active attacks*. [1]

2.2.1 Passive attacks

Passive attacks are very hard to detect since they leave little or no trace of activity. In a passive attack, the attacker monitors and maybe records the data that passes by on the network. A talented attacker doesn't even have to be physically connected to the network - a coil around a network cable will capture signals through electromagnetic induction.

Eavesdropping is the most famous type of passive attack [1]. In this attack, the attacker listens to the network and obtains communicated data. Eavesdropping has other names too: wiretrapping, sniffing, and snooping.

The most common pieces of data compromised by eavesdropping are usernames and passwords. For years, Telnet traffic has been a likely target for attackers. Telnet has been a very important protocol in the Internet, enabling remote terminal logins to a computer. The login procedure involves the users sending their username and password in cleartext to the remote computer. An eavesdropping attacker can easily capture these, and then later login to the same computer with the stolen identity. Today, the weakness of Telnet is known and users are more cautious.

Username-password-pair authentication is common among many other protocols as well since it is easy to implement and easy to use and comprehend. HTTP and POP3 are other examples of protocols that may use simple cleartext logins.

Traffic analysis is a method where an attacker obtains useful information from evaluating the flow of network traffic and not from the data itself. Useful information are frequency of transmission, identities of the communication parties, packet sizes or flow identifiers. [1]

2.2.2 Active attacks

In active attacks, the attacker takes active part in the communication. The attacker either modifies, inserts or deletes data from the stream from the legitimate party, or initiates direct connections on his own. Active attacks are usually easier to detect but they also cause the most harm.

IP-spoofing, or masquerade, is an attack where the attacker inserts a fake IP-address to IP-datagrams sent from his computer, claiming to be another party. Faking the IP-address works fine in the current Internet since IP-addresses are not authenticated at all. [2]

The Berkeley r-series are examples of tools which are vulnerable to IP-spoofing attacks. These tools can be configured to authenticate the user by the IP-address.

TCP hijacking is an attack where the attacker hijacks an already established TCP-session from the legitimate user. When a TCP-session is hijacked, the attacked user can see his session drop unexpectedly with an error. The user may login again, perhaps not even noticing that the attacker is still logged in and performing malicious actions. [3]

Replay is an attack where the data packets are recorded by the attacker and later retransmitted to the destination, often partly modified. Retransmission can cause various effects and may provide the attacker an anauthorized access to the attacked service. [1]

Routing spoofing attack involves the abuse of network routing mechanisms. The Internet has various protocols for exchanging routing data between routers and hosts. An attacker could try to redirect traffic to his host with several different attacks. Similar to the modification of routing data, a DNS (Domain Name System) server could be attacked to return false IP-address mappings for domain names. [3]

Denial of service attack is a simple one: the attacker sends much more requests to the service than the service can process. The service becomes overloaded and fails to serve its users in a timely fashion or at all. One of the most common denial of service attacks is TCP SYN flooding attack [4].

2.3 Security services

There are three core security services that are essential requirements for a secure remote access technology. These services are in cryptographic terms: [5]

- **Confidentiality**, also called privacy, is a service used to ensure that the data is kept secret from unauthorized parties on the network. Unauthorized parties must not be able to read the data exhanged between the communicating parties. Confidentiality is achieved through data encryption.
- **Integrity** is a service that addresses unauthorized manipulation of data. Data manipulation includes such things as insertion, deletion, and substitution. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. This is achieved through integrity checksum values.
- **Authentication** service consists of two related services. Entity authentication is used to authenticate the identity of the communicating parties. Data origin authentication assures the origin of exchanged protocol messages.

These are the core services that are required for performing secure remote access. While in a typical access scenario they are all desired, it is possible to utilize just some of the services. For example, it is possible to have data integrity and authentication without confidentiality. One motivation for such a scenario can be a legal one: some countries do not allow encrypted communication.

Non-repudiation is also usually mentioned along with the above three services. It deserves to be mentioned here too. Shortly defined, non-repudiation is a security service which prevents an entity from denying previous commitments or actions. However, in the context of remote access, non-repudiation is difficult to apply. This is explained in chapter 3.

The terms authentication and *authorization* refer to different actions. Authentication is the act of verifying the identity of an entity whereas authorization is the act of verifying whether the identified entity is allowed to perform a task such as read a file. The difference is finding out who versus finding out what. Authentication usually precedes authorization. Authorization is determined by the access control system.

Passive attacks only threaten the confidentiality of the data. Active attacks can threaten all security services.

Chapter 3

Cryptography basics

Cryptography is the science of information security. Cryptography studies the mathematical techniques that are used to provide the security services: confidentiality, integrity, entity and data origin authentication. In essence, cryptography provides the mechanisms for two entities to communicate securely over an insecure communication channel.

Confidentiality is achieved by *data encryption*. Encryption is the process of transforming original data called the plaintext into a form called the ciphertext which cannot be understood by unauthorized parties. Decryption is the reverse process, transformation of ciphertext back to plaintext. There are two kinds of encryption techniques, secret-key and public-key algorithms, which differ in the way the encryption keys are being used.

Data integrity and data origin authentication are provided by hash functions which use a mathematical transformation to irreversibly transform data. Entity authentication is provided by digital signatures which is an application of public-key cryptography.

3.1 Secret-key cryptography

In secret-key cryptography, *a single key* is used for both encryption and decryption, which is why secret-key cryptography is also called symmetric cryptography. The sender encrypts plaintext with a secret-key and sends the resulting ciphertext to the receiver who decrypts the ciphertext into plaintext using the same key.

There are several different secret-key algorithms, also called *ciphers*, that differ in the way they transform plaintext to ciphertext. Ciphers are categorized into two classes, block and stream ciphers [6]. *Block ciphers* encrypt blocks of data at a time, *stream ciphers* operate one bit at a time. The cryptographic strength of the algorithms varies based on the mathematical technique being employed and on the key length. A general rule is that the stronger the algorithm and the longer the key, the slower is the operation of the cipher.

Secret-key algorithms include algorithms such as DES, 3DES, RC4 and Blowfish, to name a few. Refer to appendix A for a short description of the most popular algorithms.

As the name "secret-key" algorithm implies, the key between the sender and the receiver must be kept secret from others. Therefore, before any communication can take place, the sender and the receiver must agree on the secret key in a secure way. This is a problem with secret-key cryptography. How can the secret key be exchanged securely if there is no secure channel? The keys could be agreed on personally by the parties prior to the communication, but this would become infeasible in a large multiuser environment. The number of keys required grows exponentially with the number of users. For n users, the number of unique keys required is $(n^2 - n)/2$, each user possessing $n - 1$ keys. Thus, a security solution based on symmetric-key cryptography alone is infeasible.

3.2 Public-key cryptography

In public-key cryptography, each person has two keys, *a public key* and *a private key*. The keys are generated simultaneously and mathematically bind together. The public key has no security requirements and can be published and distributed freely. The private key is kept secret and is never transmitted to anybody.

The sender uses the receiver's public key to encrypt data, and the receiver uses his private key to decrypt the received data. Secure communication does not involve sharing of any pre-agreed secrets. This is the power of public-key cryptography. There is no key agreement problem since the public keys used for encryption are publicly available. The primary advantage of public-key cryptography is easier key management. The number of keys required for secure communication between n users is only $2n$ (public-private-keypair counts as 2).

Public-key cryptography also carries the name asymmetric cryptography since different keys are used for encryption and decryption.

Public-key cryptography algorithms have two primary functions, encryption/decryption and digital signing. Some public-key algorithms can do both, some can only do digital signing.

The first and most popular public-key algorithm is *RSA*, named after its authors Ron Rivest, Adi Shamir, and Leonard Adleman. RSA algorithm can both encrypt and sign. The security of RSA relies on the mathematical difficulty of factoring large integers [7]. Keys have variable length. Longer keys offer better security. Today, 512 bit keys are considered insecure, 768 bit keys moderately secure and 1024 bit keys secure enough for most purposes. RSA was patented in US but the patent expired on September 2000.

DSA, Digital Signature Algorithm, is a public-key algorithm that is standardized by US government for the authentication of electronic documents. The security of DSA is mathematically based on the problem of discrete logarithms, which is considered to be of same difficulty as factoring large integers. DSA can only generate digital signatures. Keys have variable length just like in RSA. DSA requires SHA1 as the hash function since DSA operates on data blocks with the length of 160 bits. [8]

Key lengths between secret-key and public-key algorithms are not comparable. Public-key algorithms typically deploy longer key lengths than secret-key algorithms, but the key length alone can't be used in comparing the security of the two types of algorithms.

The primary disadvantage of public-key algorithms is slow encryption and decryption speed. Most secret-key algorithms are significantly faster. Because of this, public-key cryptography can't replace secret-key cryptography. However, the two techniques can be combined to get the best of the both worlds. In this approach, a public-key algorithm is first used to exchange a random secret-key securely, and then a secret-key algorithm is used to encrypt the session data with the exchanged secret-key. The public-key algorithm provides secure key exchange and the secret-key algorithm provides fast encryption of the session data. [5]

3.3 Hash functions

Hash functions transform a message of arbitrary length into a short, fixed length hash value. The transformation is performed without any key. Cryptographic hash functions must have two important properties: they must be *one-way* and *collision-free* [5]. One-way transformation means that it is impossible to produce the original message from the hash value. Collision-free transformation means that it is infeasible to find two different messages with the same hash value.

Hash functions are applied in cryptography since they can transform a long message into a short hash value with great speed. Most cryptographic operations are slow compared to the speed of hash functions. Instead of applying the cryptographic operation to the original message, the operation is applied to the short hash value, which concisely represents the original message. A hash value is a cryptographic equivalent of the original message, also called a message digest or a digital fingerprint.

The two most common hash functions are *MD5* from RSA Data Security Inc and *SHA1* from the US government. MD5, Message Digest 5 [9], is a hash function that generates a 128 bit hash value. SHA1, Secure Hash Algorithm 1 [10], is a hash function that generates a 160 bit hash value.

Hash functions are used in providing two security services, integrity and data origin authentication.

Integrity for a protocol message is provided by an integrity check value that is appended to the protocol message. The check value is a hash value computed from the message. However, since computing a hash value does not require a key, anybody could modify the message in transit, compute a new hash value, append it to the modified message and thus break the integrity of the message. To prevent this, a technique called *keyed hashing* is used. In keyed hash, the hash value is computed over the message plus the secret-key. Since a secret is involved, only the sender and the receiver can compute the correct hash value.

A hash value that is used for message integrity checks and that is provided by keyed hashing, is called a *Hash Message Authentication Code*, short for HMAC [11]. MAC is a generic term for a message integrity check value, HMAC is a MAC based on the use of hash function. HMAC specifies how the message and the secret-key are used in the computation of the hash value. HMAC can use any hashing function. Most common are HMAC-MD5 and HMAC-SHA1.

In addition to data integrity, an HMAC also provides data origin authentication. If the HMAC is correct, it must have been computed by the correct sender, since only the sender knows the secret-key. Thus, integrity and data origin authentication are coupled and offered together.

3.4 Digital signatures

A digital signature is a short block of data that has been computed from a document with a public-key algorithm using the signer's private key. A digital signature binds the document to a secret, a private key. A digital signature has many uses. Digital signatures can be used to provide integrity, entity and data origin authentication and non-repudiation.

A digital signature of a document is constructed as follows. First, the document is reduced to a short fixed length block of data by a hash function. Second, the hash value is encrypted using the sender's own private key, and the result is a digital signature. The signature is then attached to the document which is sent to the receiver encrypted or unencrypted. The receiver calculates a hash value from the document the same way and compares his hash value to the hash value received from the sender. If the two hash values match, the receiver can assure of the integrity and authenticity of the document. In addition, the existence of the signature proves that the sender actually produced the document; this is non-repudiation.

The remote access technologies that are evaluated in this thesis use digital signatures only for entity authentication; that is, for user and host authentication. Integrity and data origin authentication is provided by HMAC as explained above, and non-repudiation is not provided at all.

Let's say the user is to be authenticated to the server by using a digital signature. The user thus has a private-public-keypair. The server sends a challenge, a block of random bytes, to the user. The user digitally signs the challenge with his private key and sends the signature back to the server. The server decrypts the signature with the user's public key and if the result matches with the original challenge, the user has been successfully authenticated. The key idea to understand here is that the user was successfully authenticated since he proved to possess the corresponding private key. Of importance is also the fact that the user's private key was never sent over the wire. Digital signatures thus provide a secure authentication mechanism.

Non-repudiation is not provided by the remote access technologies since it has no clear semantics on a protocol level. Non-repudiation is an application level service that has practical usage for email messages, for example. Protocol packets have a short life span and they are not archived, so non-repudiation is meaningless. Cryptographically speaking, non-repudiation of protocol packets is not achieved because data origin authentication is being provided by HMAC, which is a secret-key integrity and authenticity algorithm. If a single packet was captured from a communication session, it would be cryptographically impossible to prove which of the two ends sent the particular packet.

3.5 Key exchange

Key exchange is the most critical part of establishing a secure communication channel. It is of uttermost importance that the secret-key used for encryption and decryption is exchanged securely. RSA public-key algorithm provides a mechanism to exchange a secret securely, as was explained in chapter 3.2. However, this mechanism has a weakness. The secret-key is encrypted with RSA algorithm and actually sent over the wire, which is not necessary. There exists better protocols that avoid the transmission of a secret and whose primary purpose is secure key exchange.

The most popular key exchange protocol is *Diffie-Hellman* [12]. This protocol does not provide encryption or signature services. Its sole purpose is to exchange a secret key securely over an insecure channel. The security of Diffie-Hellman relies on the difficulty of the discrete logarithm problem, similar to DSA public-key algorithm.

In short, with the Diffie-Hellman key exchange protocol the two parties can agree on a shared secret without sending the actual secret over the wire. Both parties generate themselves temporary Diffie-Hellman private-public-keys and exchange the public-keys with each other. Then they combine the public key from the other end with their own private key according to the algorithm. The result is a shared secret that only the two private key holders know. The Diffie-Hellman keypair is then discarded and a new one will be created for future key exchanges.

The Diffie-Hellman protocol provides *perfect forward secrecy*, which is an important concept [13]. Perfect forward secrecy means that the secret-keys used in a communication session are always fresh and have no relationship to keys used in previous sessions. If an attacker gets hold of the current secret-key, he can crack the current session but not any previous sessions. If RSA algorithm was used for exchanging the secret-key, and the RSA private key was compromised, the attacker could crack the current and all previously communicated sessions (assuming they were archived). This is the difference between RSA and Diffie-Hellman key exchanges.

In addition, in Diffie-Hellman, both parties contribute to the creation of the secret-key and neither party can dictate the use of any particular secret-key.

3.6 Digital certificates and PKI

Key management is the hardest part of cryptography systems. Even the public-key algorithms alone won't solve the problem of authentication in an open network environment. Public-key algorithms do offer strong authentication through digital signatures, but this authentication only proves that the party in the other end owns the correct private key. It does not tell anything about the true identity of the party owning the public-private keypair. What is required is a mechanism to reliably tie an identity to a public key.

Man-in-the-middle-attack is a well known attack against key exchange [7]. In this attack, an attacker sits in the middle of the two legitimite communication parties, intercepting and retransmitting packets at will. The attacker substitutes his own public key to the message flow in the beginning of the key exchange, and claims to be the legitimate receiver for both parties. The parties assume that they are securely communicating with each other while everything goes through the man-in-the-middle.

Man-in-the-middle-attacks can be prevented by validating the public-key of the other party. The parties could exchange the public-keys out-of-band before communicating, like is done in the PGP model of trust, where the participants meet personally to exchange their public keys. However, this model has severe scalability problems in larger and dynamic environments. There is a need for a more scalable validation mechanism. Such a mechanism is provided by digital certificates.

A digital certificate is a document that binds an identity to a public-key. The binding is enforced by a digital signature of a *Certification Authority*, CA, which is a trusted third party that enrolls certificates for individuals whose identity it has verified. The model of trust changes completely with digital certificates, from a pairwise trust model to a model with a central trusted third party. Parties no longer need to trust each other individually; they can trust a single CA. When they trust a CA, they trust all the certificates issued by the CA. [7]

In certificate authentication, the parties have a normal public-private-keypair. The public-key is included in a certificate which has been signed by the CA. In addition, parties have the certificate of the CA itself. During a key exchange, the parties exchange certificates. The signature in the certificate is validated with the CA certificate. If the signature is correct, the certificate and its public-key can be trusted. In a certificate system, the parties do not need to exchange public-keys with each other prior communication. Entity authentication is performed using a single CA certificate. Certificates thus provide scalability.

A certificate typically contains a public key, identity data such as the name of the person, validity time and a digital signature issued by the CA. The most popular certificate document format is *X.509 version 3* [14].

Certificate management contains many functions such as creation, distribution, validation and revocation of certificates. A Public Key Infrastructure, PKI, is a term for the infrastructure of technologies and entities that exist to provide the certificate management functions.

Certificates and PKI are an enabling technology. They provide a *scalable authentication framework* for remote access technologies.

Chapter 4

Technology overview

This chapter presents an overview of the three technologies that have been selected for the evaluation in this thesis. These technologies are: SSH — Secure Shell, TLS — Transport Layer Security, and IPSec — Internet Protocol Security.

The focus here is on general functionality, usage and feature set of the technology rather than on protocol packet details.

All of the technologies are either standards or drafts that later become standards. Standardization is driven by IETF, *Internet Engineering Task Force*, a top body overseeing the technical development of the Internet. Each technology has its own dedicated workgroup in IETF. Usage of the protocols is free. The IETF only accepts patent-free protocols for standardization.

Versions

This thesis describes and evaluates the version 2.0 of the SSH protocol. The earlier versions of the protocol do not provide the same feature set and the same level of security as the version 2.0 does. Version 2.0 will eventually replace earlier versions. The protocol versions are not compatible with each other. Special attention is paid to SSH version 3.0 from SSH Communications Security Corp since it is the only implementation providing support for certificates at this time. (The protocol version is still 2.0 - the product version is 3.0.)

TLS under evaluation is version 1.0. TLS 1.0 is based on SSL 3.0 with minor modifications. SSL, Secure Sockets Layer, was originally developed by Netscape Corporation to secure HTTP. However, for the purposes of this evaluation, TLS and SSL are considered to have equal functionality, though they do not interoperate in practise. The TLS protocol identifies itself as SSL 3.1. [15]

IPSec consists of multiple protocols and therefore cannot be referred to by a single version number.

4.1 TCP/IP model

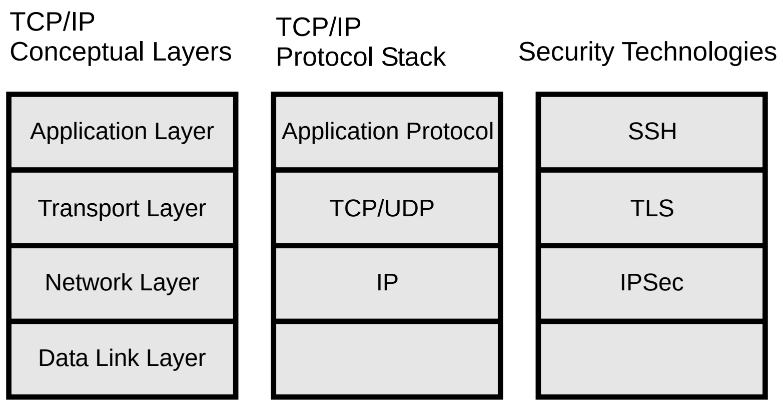

Figure 4.1: TCP/IP model

Networking components and services are traditionally modelled with a layered architecture. Each layer has well-defined functions and provides an interface for the layer above. The TCP/IP conceptual model consists of four layers.

At the top of the conceptual TCP/IP model are the applications that use the services of the transport layer to send and receive data over the network. The transport layer provides guaranteed end-to-end connectivity for applications. The network layer provides best-effort routing of single packets between interconnected networks. At the bottom are the individual networks, or data links, with several different types of physical media, electrical cabling and network cards.

The TCP/IP protocol software stack covers three layers of the TCP/IP model. Application protocols such as HTTP, FTP and Telnet operate at the application layer. At the transport layer, TCP provides connection-oriented and reliable transport service and UDP provides connectionless and unreliable transport for application protocols. At the network layer, IP provides connectionless and unreliable delivery of IP-datagrams. Since IP is at the lowest layer of the protocol stack, all data in a TCP/IP network is carried by IP.

Security services can be applied at each layer of the TCP/IP protocol stack, each solution having its own characteristic properties. SSH operates at the application layer, TLS operates at the transport layer, and IPSec operates at the network layer.

4.2 Secure Shell

SSH provides security at the application layer of the TCP/IP protocol stack. SSH is not just a protocol but a set of integrated applications and services for providing secure remote access for diverse tasks in a flexible way. SSH is a versatile security solution that has become almost ubiquitous in remote administration. [16]

SSH was developed to solve the two most acute problems in the Internet, secure remote terminal logins and secure file transfers. Telnet and FTP offer no protection for data and are easy targets for eavesdropping attacks. The primary goal of SSH has been to replace these insecure protocols with a secure one. SSH provides Telnet-like remote terminal sessions with strong authentication and encryption, and files can be copied securely between computers. By securing these two important tasks, SSH has gained ground rapidly and become an essential tool for network administration.

A famous set of insecure tools for remote logins and file copying, the Berkeley r-series, can easily be replaced by the SSH tools. SSH offers secure equivivalent drop-in commands for rsh, rlogin, rcp, and rdist.

SSH can also tunnel arbitrary TCP sessions over a single encrypted SSH connection. UDP based protocols can't be tunneled. Tunneling is a powerful feature that makes it possible to secure the communication of other applications and protocols without modifying the application code at all. By using tunnels, users can continue to use existing insecure applications in a secure manner. Tunnels originate from the client host and end up either in the SSH server itself or in another server. The latter part of the tunnel from the SSH server to another server is always unencrypted.

One common protocol that is tunneled over SSH is X11, which provides graphical remote sessions. Plain X11 is very insecure but through SSH it can be used securely. Email is another popular application that is tunneled over SSH. With tunneling, SSH can offer quite an encompassing solution for securing most of communication tasks.

Usability has been an important goal for SSH. For users with Unix or networking background, SSH applications are easy to use and do not require a steep learning curve. Security is provided transparently through similar tools that the user is already familiar with. Installation and configuration can be done with relative ease. User friendliness has been beneficial for getting wide-spread use. For average users without networking background, however, the use of SSH requires some education.

It is characteristic of SSH that it is very user-centric. SSH authenticates the user, and tools often require interactive input from the terminal. However, SSH can be used without user intervention, in automated scripts, for example.

SSH has been ported to several different platforms. All the mainstream Unix operating systems are supported, as well as Windows. SSH comes pre-installed in many open-source systems. There are millions of SSH users around the world.

Having being developed outside US (SSH from SSH Communications Security Corp in Finland and OpenSSH mainly in Canada), SSH has not been subject to weakened cryptography. Algorithms or key lengths are thus not crippled in any way by government regulations. The objective has been to create a high-security tool by using the strongest cryptography available providing protection against the strongest attacks for decades.

4.2.1 Applications

There are many implementations of SSH from different vendors. While the protocol is standardized, the applications and their features may differ. However, a typical implementation of SSH includes the following applications.

ssh is the main application providing interactive terminal sessions, program execution and tunneling. *scp* is a simple file transfer application while *sftp* offers more powerful file transfer with features similar to FTP. *ssh-keygen* generates new keypairs for public key authentication. *ssh-agent* is a helper utility used to provide single sign-on for user authentication.

4.2.2 Protocol

SSH has a client-server architecture: a server listens on a TCP port 22, which has been officially assigned for SSH, and clients initiate connections to this port. Several SSH servers can be running on the same host on different ports. SSH defines a packet-based binary protocol that runs over a reliable transport stream, TCP being the most common one. The protocol does not run over UDP.

The SSH protocol has been divided into three distinct subprotocols with well-defined functions.

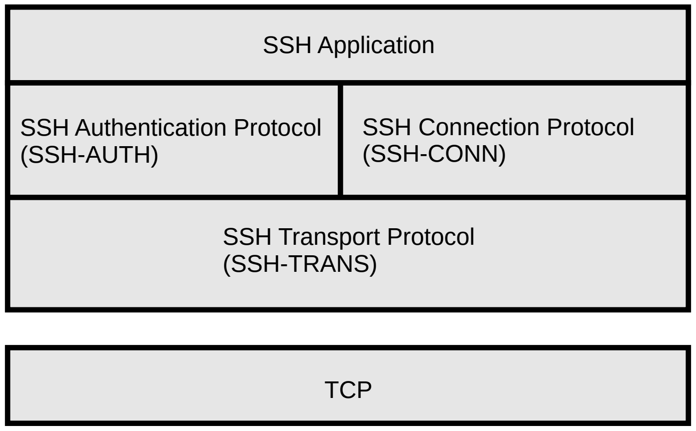
SSH Application
SSH Authentication Protocol (SSH-AUTH)
SSH Connection Protocol (SSH-CONN)
SSH Transport Protocol (SSH-TRANS)
TCP

Figure 4.2: SSH protocol architecture

SSH transport layer protocol provides server authentication, confidentiality and integrity, and optionally compression. SSH user authentication protocol takes care of authenticating the user to the server. SSH connection protocol provides multiplexing service for running several logical channels over a single encryption connection. [17]

After the establishment of the TCP connection, the transport layer protocol starts its job. Algorithms for key exchange, server authentication, encryption, MAC and compression are negotiated. Keys are exchanged and the server is authenticated. Once the transport layer protocol completes, the connection has been secured and the two other subprotocols can communicate in privacy.

Currently the only defined and required key exchange algorithm is Diffie-Hellman. This is a cryptographically strong algorithm and provides perfect forward secrecy. Key exchange can be reinitiated during the connection to provide additional protection. The following encryption algorithms are currently defined: 3DES, Blowfish, Twofish, ARCfour (RC4), IDEA and CAST128. Of these, 3DES is specified as mandatory in any implementation. Two MAC algorithms are defined: HMAC-SHA1 and HMAC-MD5, of which the first is mandatory.

The SSH protocol supports full negotiation of all algorithms and methods. The subprotocols have been designed to be simple yet flexible. New algorithms, authentication or key exchange methods, or even new subprotocols can be defined.

4.2.3 Authentication

SSH provides mutual authentication. Both server and user are authenticated.

SSH authenticates the SSH server service, which is also called the daemon in Unix vocabulary. SSH does not authenticate the server host. A host may run several SSH daemons on different ports and each daemon can have a unique identity. Thus, daemons are authenticated, not hosts. (In this thesis, the term "daemon" is not used. The more common term "server" is being used to refer to the daemon.)

The server is authenticated with a digital signature based on DSA or RSA public key algorithm. Each server must have a public-private keypair. Upon connection, the server proves its identity to the client by sending the server public key and a digital signature of the public key to the client. (SSH public keys are thus self-signed.) The client validates the signature in order to trust the server. In implementations without support for certificates, clients refer to a local database of trusted server public keys. If the provided server public key exists in the local database, the key can be trusted. If the server public key does not exist in the database, key validity can be verified through a fingerprint that the SSH client prints out for the user. The user should verify that the fingerprint matches the official fingerprint received from administrators of the server or from another reliable source. Otherwise, trusting the public key without any verification makes the client vulnerable to man-in-the-middle attack.

SSH has not supported certificate authentication until version 3.0 from SSH Communications Security Corp. Earlier and other implementations use plain public-private keypairs for server and user authentication. In SSH version 3.0, servers can authenticate themselves to the client with X.509v3 certificates. When certificates are used, the client doesn't need to have a local database of trusted server public keys nor server certificates. Instead, just the few trusted CA certificates are stored on the client, and the client trusts all the servers whose certificate is signed by any trusted CA. Certificates provide good scalability and most importantly, protection against man-in-the-middle attack.

User authentication has always been a strong asset of SSH. Multiple methods have been standardized for user authentication and new methods can be implemented with ease to the protocol. Depending on the policy on the server, the user can have freedom in selecting the authentication method during the login. It is also possible that a successful user login may require passing of multiple authentication methods. The server drives the authentication process according to its security policy.

SSH provides confidentiality for user authentication. User identity is not revealed to eavesdropping parties since the user is authenticated after the connection has been secured.

The standard user authentication methods are password, public key and hostbased authentication [18]. Of these, the public key method offers the best security. It is the only method whose implementation is required by the protocol specification. In public key authentication, the user must let the server know the public keys through which authentication is allowed to his account. In other words, the user must upload his public key files to the server and edit a configuration file. The server thus has a database of user public keys, similar to the client having its database of server public keys.

In user certificate authentication provided by SSH version 3.0, the user doesn't need to upload any public key files to the server prior the connection, and the server doesn't need to have a database of user public keys nor certificates. The server validates the user certificate by using the CA certificates that the server has been configured to trust. There is less configuration hassle and more scalability. This is the power of certificate authentication.

Password authentication is very popular, since it is easy to use and most common. Hostbased authentication authenticates the client host similar to Berkeley services authentication. It is not suited for high-security servers because of the risk of compromised server private keys. Other authentication methods supported by SSH vendors are Kerberos, SecurID, S/Key, and PAM, Pluggable Authentication Module architecture from Sun Microsystems. Support for other authentication services is under development.

4.3 TLS

TLS operates between the application layer and the transport layer of the TCP/IP stack. Although the name may suggest it, TLS does not run in the transport layer since that is usually part of the kernel of the operating system and TLS does not require kernel modifications. The name indicates that TLS provides a secure transport for application protocols.

The great advantage of TLS is that it is platform and application independent. Any application protocol running on top of TCP can secure communication with TLS. Since TLS is a pure protocol, and does not interact with the operating system the way SSH does, portability to different platforms is not an issue. TLS does not address access control issues; that is the job of the application using TLS.

The predecessor of TLS, SSL, is propably the world's most widely used security protocol. A large part of the traffic in the Internet today is secured by TLS or SSL. This is due to the fact that they have become the industry chosen protocols for securing WWW-access. The protocols are built-in to WWW-browsers and servers. Whenever an URL to a site begins with "https://", TLS or SSL is being used as a secure transport for HTTP. Another popular application using TLS is email, for secure POP3 and IMAP email access.

The TLS protocol is usually implemented as a toolkit. Application developer should be able to integrate TLS into his application with little work, but the ease depends on the design of the application. Of course, the integration is required for both client and server software. The most famous TLS toolkit is OpenSSL, which is open-source and available for free.

TLS can integrate into an application quite seamlessly. TLS should not make an application more difficult to use in comparison to an application without TLS. Integrated security offers ease of use.

Since the most popular WWW-browsers were developed in the USA, SSL has been subject to crippled cryptography. In the early implementations of SSL in WWW-browsers, the key length was limited to 40 bits due to export regulations in the USA. Since then the regulations have relaxed and strong cryptography is available, with key lengths up to 128 bits. For software developed outside the USA, such restrictions have not taken place.

4.3.1 Protocol

Similar to SSH, TLS only runs over a reliable transport stream, such as TCP. TLS is a session-oriented protocol and hence won't run over UDP.

TLS protocol consists of two layers: the record layer protocol and the handshake layer protocol, which further consists of a suite of four subprotocols [15]. Figure 4.3 displays the structure of TLS.

The record layer protocol provides a secure transport for higher layer protocols. It divides the data stream from upper protocols into blocks, applies encryption, integrity and compression, and sends and receives the data blocks. Communication with the layer above is done through data structures called records, which all have a unique identifier. New records can be defined for new subprotocols.

The TLS session always starts with a handshake where the handshake pro-

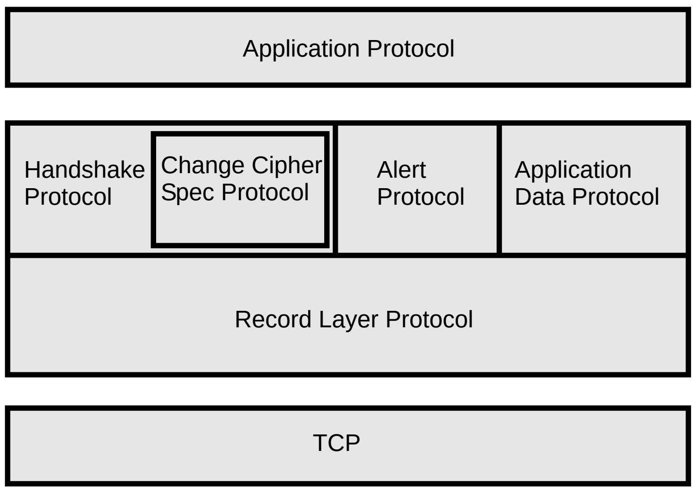

Figure 4.3: TLS protocol architecture

tocol negotiates security parameters for the secure session. It negotiates version numbers, algorithms for key exchange, encryption, MAC and compression. Optionally, it takes care of authenticating both ends.

Change cipher spec protocol is used to change encryption, MAC and compression algorithms. This protocol is used during handshake, but it can also be used later during the connection, for example if the nature of the data requires a switch to stronger or weaker algorithms.

The alert protocol is used to communicate alerts for the connection. There are two severity classes: warnings and fatal errors. A fatal error always terminates the connection. Alerts can be such as "handshake failure" or "certificate expired".

The fourth protocol is the application data protocol which simply transmits the data to and from the application.

The TLS specification defines two key exchange algorithms, RSA and Diffie-Hellman. The weakness of RSA is that it does not provide perfect forward secrecy. For this reason, Diffie-Hellman is recommended over RSA. Unfortunately, RSA is more common since it was defined and implemented first. The following encryption algorithms are currently defined in the specification: DES, 3DES, RC2, RC4, and IDEA. 3DES is specified as the mandatory algorithm. Defined MAC algorithms are HMAC-SHA1 and HMAC-MD5.

A TLS-secured service usually runs on a different TCP-port than the insecure version of the service. HTTP secured by TLS runs in port 443, Telnet in 992, IMAP in 993 and POP3 in 995, to name a few. While it has been common practise to sacrifice a different port to a secured version of the service, it has also been suggested that the same port should be used for both versions. In this approach, the application protocol must be enhanced to include a "STARTTLS" command which activates TLS [19]. However, this approach requires modifications to the application protocols, and thus has not gained popularity among all applications.

4.3.2 Authentication

The original TLS specification defines three choices for authentication: no authentication, server authentication only and mutual authentication. Both server and user authentication are based on X.509v3 certificates with DSA or RSA digital signatures.

The server sends its certificate in the first reply to the client, and the client verifies the certificate validity. The verification process is not defined by TLS since it falls beyond the scope of TLS. Typically the clients have a local database of trusted CA certificates. For example, popular WWW-browsers install with built-in certificates of well-known CAs.

If the server requires user authentication by TLS, the client must send its certificate to the server for verification. However, in practise it is very common that only the server is authenticated by TLS. Users are usually authenticated by the application protocol that runs after TLS connection has been set up. By far the most popular way for an application to authenticate user is a username-password pair.

Other authentication methods have been proposed to TLS, but they are not yet in wide use. Kerberos authentication uses popular Kerberos symmetric key authentication system [20]. SRP, Secure Remote Password protocol, proposes a mechanism to perform simple username-password authentication with TLS [21].

A major weakness of TLS is that it does not provide confidentiality for user certificate authentication. The user certificate is transmitted in plaintext before the connection has been secured, which means that an eavesdropper can see the identity of the user.

Similar to SSH, TLS authenticates the server service, the daemon, and not the server host.

4.4 IPSec

IPSec is the most comprehensive solution for securing all communications within a network. IPSec is not a single protocol but a security architecture that defines a framework of protocols and mechanisms for securing networks. IPSec tries to address the security issues of the Internet Protocol in the most fundamental way, specifying a set of standardized security extensions to the protocol. IPSec has the potential to become the dominant standard technology for securing IP-communication. IPSec will be mandatory in the forthcoming IP version 6, but it also works with the current IP version 4, which is important since the transition to version 6 is still years ahead.

IPSec secures communication at the IP-layer of the TCP/IP-stack. Since all traffic in the Internet travels on top of IP, IPSec in essence secures the whole network. By securing the network, it secures all application protocols without requiring any modifications to them. IPSec holds the promise of truly transparent security. UDP based protocols get secured as well.

Unlike SSH and TLS, which are used to secure sessions between a user and a service, IPSec secures traffic between hosts. It authenticates the hosts and creates secure paths between them according to the security policy. There may be multiple paths with different levels of security. Paths enable flexible use of security: IPSec can secure all the traffic from the host or just the selected data. The paths are connectionless and independent of any TCP-sessions. Closing a TCP-session does not close a path. In this sense, an IPSec connection between hosts is permanent.

In IPSec, any host can both initiate connections to other hosts and respond to connection requests from other hosts. IPSec defines terms *initiator* and *responder* for these, respectively. IPSec can be used in three main scenarios: for host to host, host to gateway and gateway to gateway connections. The last scenario makes IPSec stand out from the other technologies. Deploying IPSec in gateways makes it possible to securely tunnel all traffic over the Internet from network to network. This kind of network is called a VPN, a virtual private network, which is a hot concept in the networking industry.

There are many IPSec toolkits available for implementing IPSec in products and systems. However, the target audience of the toolkits is not application developers but operating system vendors. Since the IP-layer is traditionally part of the kernel of the operating system, IPSec is integrated into the kernel. The integration is a much more complex task than securing a single application protocol. Implementing IPSec in a operating system requires significant effort from the operating system vendor or from an

IPSec vendor. However, the great advantage of IPSec is that the integration is done once per system, not per each application, which makes the technology very compelling. The technology is complex, but the rewards are greater.

The specifications are quite flexible. IPSec is not tied to any specific encryption or integrity algorithm, or key management and distribution protocol. Attention has also been paid to performance issues. Speed is essential in heavy-duty routers and therefore hardware based encryption is often offered by IPSec vendors.

4.4.1 Security Protocols

IPSec defines two base security protocols, AH and ESP. They both operate on connectionless IP-datagrams. [13]

AH, Authentication Header protocol, provides integrity, data origin authentication, and an optional anti-replay service for IP-datagrams. Data origin authentication is based on MAC algorithms HMAC-SHA1 or HMAC-MD5, which are specified as mandatory by the specification. AH inserts one new header to IP-datagram.

ESP, Encapsulating Security Payload, provides confidentiality, integrity, data origin authentication, and an optional anti-replay service. In short, ESP provides the same services as AH plus confidentiality. ESP inserts two new headers into IP-datagram, a header and a trailer. (It encapsulates the payload between the headers, hence the name.) The specification requires that a compliant implementation must support DES.

Since ESP an AH have overlapping functionality, typically either one of them is used at a time. If confidentiality is required for the connection, ESP is selected; otherwise it is AH.

ESP is identified as IP-protocol 50 and AH as 51. (UDP is 17 and TCP 6. [22]) These protocol identifiers are not to be confused with TCP-ports which are higher level concepts.

Both of the protocols can be used in two modes. In transport mode, the original IP-header is kept intact and AH or ESP header is inserted after the original IP-header and before the payload. In tunnel mode, the whole IP-datagram is inserted inside a new IP-datagram that is generated. In short, the difference is that transport mode secures only the payload and tunnel mode secures the whole datagram. Tunnel mode provides better protection

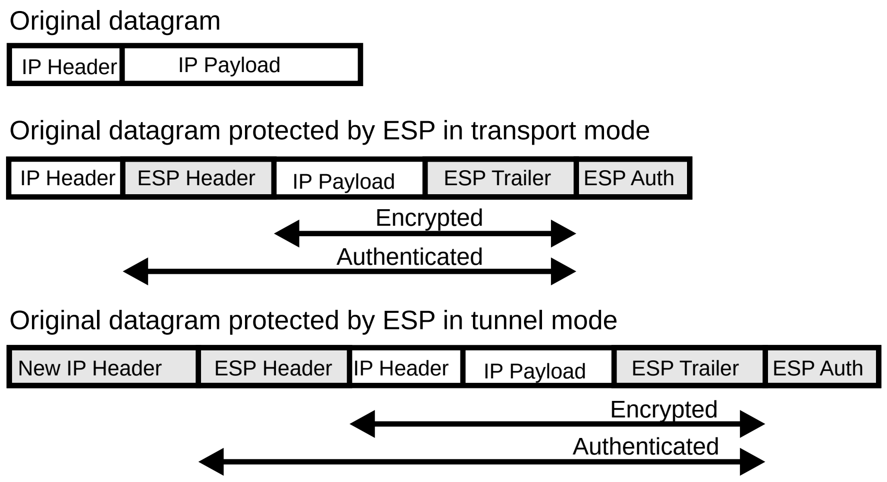

Figure 4.4: IPSec-ESP protocol

but it has a larger processing overhead.

Transport mode is only used between hosts. Tunnel mode is typically used when one or both of the end points is a gateway. Tunnel mode is suitable for creating a VPN between two private networks. In tunnel mode, the tunneled IP-datagrams are totally protected and not examined by intermediate routers. Original IP-addresses are hidden, which makes it possible to interconnect networks where a private address space is being used.

Datagrams whose authentication fails are discarded at the earliest possible moment and never delivered to upper layers of the TCP/IP stack. This prevents denial of service attacks where the attacker tries to flood the host with bogus TCP-connections. IPSec thus prevents the TCP SYN-attack efficiently: a bogus TCP-connection is not even set up because the authentication of the datagrams fails.

Anti-replay service tries to prevent an attack where an attacker replays copies of already received old datagrams. This is implemented by having a sequence number field within the protocol header. An inbound datagram is discarded if its sequence number falls outside a sliding anti-replay window. A window of accepted sequence numbers is required instead of just one sequence number since the IP-datagrams may not arrive at the destination in the order they were sent. [13]

4.4.2 Key management

Key management is the most complex part of IPSec, and only a rough overview of the functionality is presented here.

IKE, Internet Key Exchange, is the default key management protocol for negotiating IPSec protocols, algorithms, keys, certificates and other associated security parameters between communicating parties. Security parameters negotiated by IKE are stored in a security association, SA, which is an important concept in IPSec. SA's are used by the security protocols AH and ESP, and contain all the data that the communicating parties need to execute IPSec protocols.

IKE is a hybrid protocol that combines ISAKMP, Internet Security Association and Key Management Protocol, with Oakley key exchange. ISAKMP provides a generic framework for key exchange, and Oakley defines the actual key exchange protocol, which is based on Diffie-Hellman key exhange algorithm. ISAKMP specifies that key exhange is communicated via UDP-port 500.

IPSec defines two ways to exchange keys. Manual keying involves manual configuration of keys and parameters in each host. IKE is not used. Manual keying is feasible only in smaller environments, and is typically used between static gateways. It is useful for testing purposes too. Automatic key exchange uses IKE to provide dynamic on-demand creation of IPSec connections, and thus scales for larger environments. However, automatic keying with certificate authentication requires PKI software configuration and infrastructure.

Since key exchange is the most critical element in creating secure connections, IKE defines a robust key exchange mechanism with two phases. In phase 1, IKE authenticates the end points and negotiates a secure channel for itself to be used for further negotiations. There are two alternative modes for executing phase 1, main mode and aggressive mode. The benefit of main mode is that it provides confidentiality for end point identities. Main mode includes six packet exchanges. Aggressive mode is faster with three packet exchanges, but it reveals identities. In phase 2, called quick mode, the security parameters are securely negotiated for actual IPSec connections. Phase 2 is less complex than phase 1, and can be executed multiple times over the same phase 1 channel.

4.4.3 Authentication

It is very important to realize that IPSec authenticates hosts, not users or services. IPSec operates at the network level and thus has no linkage to users or services which are upper level concepts.

Host authentication is primarily based on certificates, x509v3 being the most popular with support for DSA and RSA signatures. Certificates are exchanged by IKE during key exchange. Again, verifying the certificate validity is beyond the scope of IPSec framework, which is true for SSH and TLS too.

Authentication is performed once during IKE phase 1 of the key exchange. Multiple IPSec paths with different security parameters can be created between the two hosts in IKE phase 2 without further authentication.

IKE key exchange in main mode provides confidentiality for the initiator host identity while aggressive mode does not.

The certificate is bound to either the DNS-name or the IP-address of the host. If the DNS-name or the IP-address of the host changes, authentication will fail. This is a problem in remote access scenarios where the host attaches to a foreign network and receives a new IP-address and a DNS-name. To support these remote access scenarios, the certificate can be bound to a specific user. However, technically IPSec still authenticates the host, not the user.

Since it is more desirable to authenticate the user and not the host, and because there are many legacy user authentication systems in use, vendors have implemented additional support for user authentication. There are various proposals for this, including one-way user authentication in IKE [23], [24]. These proposals are controversial since "purists" demand that user authentication should not be integrated with IKE. On the other hand, vendors want to provide the features that customers want. Vendor-specific extensions are common and since there is no IKE standard, interoperability is a problem. Most likely we'll have to wait some years for the dispute to settle and standards to emerge.

IKE defines one authentication method which doesn't require certificates. This is called the preshared secret authentication method. In this, the hosts are identified with a preshared secret that must have been delivered to hosts out-of-band before the connection. It is quite evident that this authentication method scales badly and is rarely feasible in large environments. In addition, binding a preshared secret to a user is not specified and implementation varies from vendor to vendor.

Chapter 5

Technology evaluation

In this chapter, technologies are evaluated in real life scenarios. Having a scenario-based evaluation with practical problems from real life helps to explore the advantages and the weaknesses of a technology in the most concrete way. The goal is to discuss how different technologies solve given practical problems, and not just build and present a list of random features.

Since the emphasis of this evaluation is on solving practical problems, not only is the technology explored, but existing software implementations of the technology are considered as well. A look is taken at how the technologies solve problems today. Also studied are some standard proposals that suggest improvements for a technology.

Technical criteria which are used for evaluating each technology are listed and categorized first. Then, a number of various remote access scenarios are defined. Each scenario has requirements and problems to be solved. Each technology is applied in turn to the scenarios to examine how well the technology solves the problems.

5.1 Evaluation criteria

The Figure 5.1 displays the criteria that will be used for evaluating each technology. Four main criteria are defined with subcriteria.

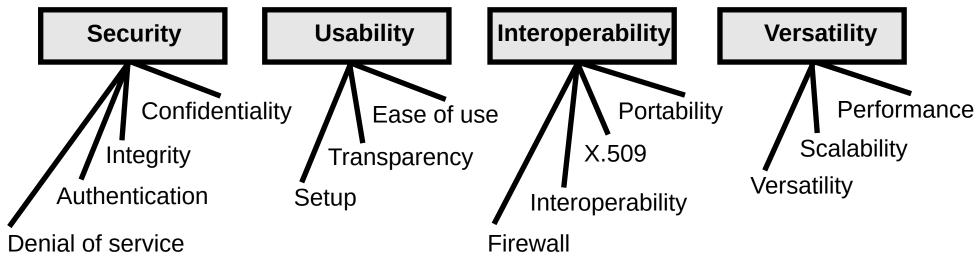

Figure 5.1: Evaluation criteria

1. Technology should provide secure remote access with strong user and service authentication.
   This the main and most important criteria. This includes the following subcriteria. **Confidentiality and integrity:** Technology should provide strong confidentiality and integrity of data. **Authentication:** Technology should have strong user and service authentication mechanisms. **Denial of service:** Protection for denial of service attacks is desirable.

2. Technology should be easy to use.
   Ease of use: Technology should be easy to use for the average user. **Transparency:** Security should be provided as transparently as possible. **Setup:** Software should be easy to deploy, install and configure.

3. Technology should interoperate with different networking environments.
   Portability: Technology should be portable to different platforms. **Interoperability:** Different implementations of the technology should interoperate. **X.509:** X.509v3 certificates and PKI should be supported. **Firewall traversal:** Technology should be transparent to firewalls.

4. Technology should be versatile and scalable.
   Versatility: Technology should secure many applications and solve the most common security problems. **Scalability:** Technology should scale to large environments. **Performance:** Security should not degrage performance.

The three security services that were listed in chapter 3, confidentiality, integrity and authentication, are essential requirements of a secure remote access technology. Authentication is covered in many scenarios. However, since a full mathematical cryptoanalysis is outside the scope of this thesis, the confidentiality and integrity are not analyzed in a great detail in the scenarios; an overview discussion is presented in chapter 6.

The criteria was chosen for the following reasons. The first criterion is obvious: a secure remote access technology should be secure. Secondly, a technology is not usually deployed if it is hard to use. Similarly, if a technology does not fit into current networking environments and solve current problems, it has little chance of success. Last, a technology should not just solve one particular security problem but be applicable in many ways.

5.2 Introduction to scenarios

Three different remote access scenarios are defined where each of the three technologies will be evaluated. The scenarios are fictitious but they represent true and common problems from real life. The intention has been to choose a few scenarios that best match the most common remote access tasks and environments in the Internet.

Each scenario defines the environment and the tasks that are to be secured. To avoid repetition, all of the criteria are not evaluated in each scenario. Instead, each scenario focuses to a set of criteria that are evaluated.

1. The university scenario is an example of a large academic or research environment with a fairly open public network, which consists of hundreds of different multiuser hosts and thousands of users. The most basic need in a university network is remote terminal access and execution of programs on servers. X11-graphical remote sessions are in active use. The methods by which terminal and X11-sessions can be secured by each technology are explored. In addition, being a large environment, scalability and support for PKI is evaluated in this scenario.

2. The home dial-up user scenario focuses on how dial-up customers of an Internet Service Provider can secure their network login, email access and file uploads. Email access and file transfer over the Internet are the most essential tasks performed by everybody, so it is worth evaluating how these tasks get solved by each technology.

3. The road-warrior VPN scenario evaluates how a nomad company employee can connect to his company private network and access various services while travelling on the road with his laptop computer. Problems in connectivity from a public network to a private network are discussed. This kind of scenario is becoming very common in the business world because company data is rapidly being transformed into electronic form and accessed while traveling.

It is recommended that the scenarios are read in order since the latter scenarios build on former ones.

The evaluation consists of specific remote access tasks such as "how to secure email access?" and includes general discussions of the essential issues in the scenarios.

5.2.1 Products

Most of the evaluation is carried out on the basis of the technology itself, but a few software products are being evaluated as well, mostly software that integrate with TLS.

I have objectively tried to select the products that best represent their class in order for the evaluation to be fair. Refer to Appendix B for the list of products that have been included in the evaluation.

5.3 Scenario 1: University

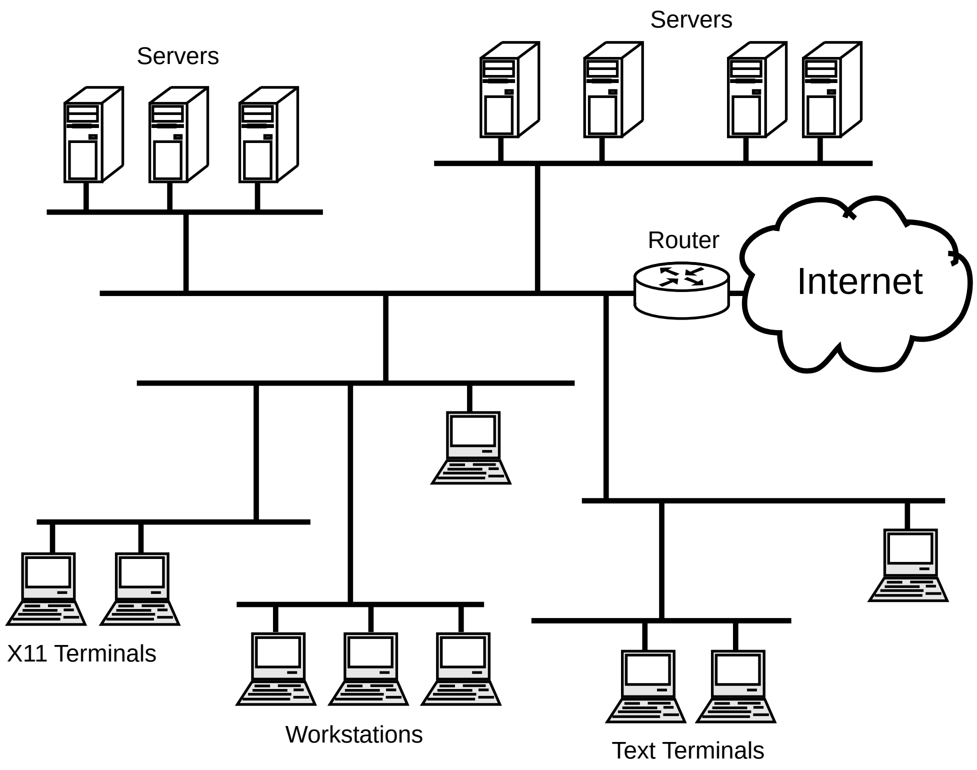

Figure 5.2: University network

Our example university has hundreds of different hosts connected to the campus network. There are many flavors of large capacity Unix servers. Workstations run Unix, Microsoft Windows and Apple Macintosh operating systems. Most of the workstations are shared and located in public computer rooms. In addition, there are dumb text and X11-display terminals without disk or processor.

Users are authenticated with simple username-password-pairs. Currently, terminal logins are conducted through the insecure Telnet-protocol.

5.3.1 Task: Secure terminal access

Terminal access from the workstations around the campus to the Unix servers should be secured.

SSH — SSH provides good terminal access functionality; this is the primary task of SSH. SSH server software is installed on Unix servers and SSH client software on workstations. SSH has been ported to many platforms. Installation is relatively easy and does not require reboot of hosts.

Because university users are already familiar with telnet, and maybe rlogin or rcp as well, transition to SSH is fairly painless. SSH applications operate similarly as their insecure counterparts. Graphical user interfaces are available for Windows and Macintosh users.

Most SSH implementations do not yet support certificates so let's first evaluate SSH without certificates. Each server host has a private-public keypair that identifies the server. Host keypair is either generated during installation, or manually after. Server configuration files and especially the private key must be protected and made accessible only by the network administrators. The server is authenticated by its public key, which is sent by the server to the client in the beginning of the connection. The client refers to a local database of server public keys to verify authenticity of the server public key. If a copy of the server's public key is found in the local database, the public key can be trusted. If the server's public key is not found in the database, the user is prompted with the fingerprint of the key and asked to accept it. The user should take steps to validate the provided fingerprint. If the fingerprint is accepted without proper validation, the client becomes vulnerable to man-in-the-middle attack.

In the above scenario, each user has his personal database of trusted server public keys. Unix-workstations with secure file systems could have a system-wide administrator-maintained database of all server public keys.

However, maintaining such a database on each workstation may become impractical in large environments, unless workstations access the same database via network file sharing. (Helsinki University of Technology has such an arrangement. The security of network file sharing is a topic in itself, and is not covered here.) In practise, users usually build their own personal database under their home directory. The first connection to a new SSH server is usually vulnerable to man-in-the-middle-attack, but if the key is validated properly and then stored to the database, future connections to the same server can be trusted.

If certificates are supported, scalability problems and man-in-the-middle-attacks can be avoided by using X.509 server certificates. Certificates are generated by a CA. A certificate revocation mechanism for server certificate revocation is optional but recommended. However, all SSH implementations that are in use must then support server certificate authentication. Using certificates can cause interoperability problems with some SSH implementations. For example, OpenSSH is a popular free SSH implementation without support for certificates at this time.

Revocation of server public keys is practically impossible without using certificates in a large environment such as the university. If a server private key gets compromised, it must be discarded and a new public-private keypair must be created. However, the old public key is still stored in the personal public key databases of all users. When the users connect to the server which has a new public-private keypair, the SSH client correctly alerts the user about a possible man-in-the-middle-attack. This security alert can create confusion among users, and raise a flood of queries to the network administrators. Revocation is difficult with public keys since there is no mechanism to notify all users about the revocated keypair. This problem does not exist with certificates. If certificates were used, revocation would be transparent. The new server certificate would be accepted by the SSH clients as long it was signed by a trusted CA, and no alert about a changed certificate would be displayed to the user.

User authentication follows server authentication. Simple password-authentication is the default user authentication method supported by SSH, so SSH integrates with existing password authentication infrastructures effortlessly. In addition, security-focused users can start using public key authentication if the administrators have enabled it on the server, which usually is the case. Public key authentication provides better security and single sign-on when used with the ssh-agent application.

TLS — TLS itself is just a protocol that provides secure transport for an application protocol. To have a secure terminal service, TLS is integrated into a Telnet software. Telnet server software is installed on Unix servers and Telnet client software to workstations.

There are not many Telnet implementations that integrate with TLS. Only two Unix packages were found: STelnet and start_tls_telnet. They are both Unix-only software and the documentation says they run on most Unix platforms. For Windows platform, a freeware Telnet client called Tera Term is available with a TLS extension. The Macintosh platform seems to be unsupported.

The STelnet server listens on port 992, the standard port assigned for Telnet over TLS, and starts communicating using TLS at the beginning of the connection. The start_tls_telnet server listens on the standard Telnet port 23, and starts to communicate with insecure Telnet, then switches to TLS upon START_TLS command issued by the client. START_TLS command for Telnet is documented in Internet Draft [25].

TLS servers are authenticated using X.509-certificates. A CA must be available for certificate creation. Each TLS secured Telnet server has a certificate that is validated by the client upon connection. For the validation process the clients must have the CA certificate locally available in a secure storage. The user is authenticated using username-password-pair. The TLS protocol does not provide a standard password authentication method, so it is the Telnet application code that implements the password authentication over the secure TLS connection. The distinction between TLS provided user authentication method and application provided method is good to understand.

start_tls_telnet is developed most frequently and seems to be the best choice of the evaluated TLS secured Telnet softwares. Tera term, the only Windows client, is not actively developed. Its last modification dates to 1998. It is most likely that the infrequently developed software have interoperability problems.

A small survey on the most popular Telnet clients was done, and it was found that there are a number of Telnet clients that implement both Telnet and SSH in the same client. In other words, it is more common to enhance Telnet applications with SSH than it is to enhance them with TLS. Telnet is used for insecure terminal access while SSH is used for secure terminal access.

IPSec — IPSec secures all communication between hosts. Since all communication is secured, any standard Telnet software can be used for providing a secure login service. Telnet software is not modified. IPSec software is installed to servers and workstations.

IPSec offers two methods for host authentication, preshared secrets or certificates. The preshared secret method is not feasible in a large university environment since distributing secrets to all hosts would be very laborous and impractical. Each host would need the secrets of other hosts. This is similar to SSH without server certificates: SSH clients would need to store the public keys of all servers. Certificates provide a better solution.

A certificate and its associated private key are created for each host. The host certificate, private key and CA certificate are delivered securely to each host by the network administrators. An IPSec security policy is configured on each host. The policy can be the same on each server: only encrypted IPSec connections are allowed to Telnet-port 23 and the host certificates must have been signed by the trusted CA. Private keys and policy files must be stored securely and kept non-accessible by regular users.

With IPSec, authentication has two steps. First, a two-way host authentication is done by IPSec and then a one-way user authentication by Telnet server. Telnet server is not aware of IPSec functionality and continues to authenticate users by simple username-password-pairs.

IPSec is fully transparent to the user and does not interact with the user. The user may not even be aware that IPSec is being used. All the user sees is the familiar Telnet login prompt.

Full transparency can also be a problem. Because IPSec operates at low level in the protocol stack, it is technically difficult to provide feedback for the user when a problem occurs in IPSec connectivity. For example, if an IPSec connection attempt to a Telnet server fails due to an expired host certificate, the Telnet client may just return a generic IP error "Host unreachable" to the user, which doesn’t reveal anything about the real cause. Providing meaningful feedback for an application is a fundamental problem of IPSec since it operates at a low level and is meant to be hidden from the applications.

The portability of IPSec is most likely a problem in a university environment with a variety of different systems. Being a new and evolving technology, IPSec has not been yet ported to all platforms. Most popular operating systems get support first, but other systems may have to wait a long time for an up-todate IPSec implementation. A multi-vendor solution has the risk of interoperability problems.

5.3.2 Task: Secure X11-sessions

Users need to run graphical X11-programs such as Netscape or Mathematica on Unix-workstations which have X11-display servers installed. X11-sessions should be secured.

SSH — Securing X11-sessions with SSH is almost trivial since SSH has built-in support for it. The user just needs to request X11 tunneling in the client and SSH will configure X11 on the server to forward traffic securely over the SSH connection. X11-forwarding using SSH is very transparent to the user.

TLS — start_tls_telnet is the only TLS enabled software that claims to support secure X11-forwarding, but it conforms to an expired Internet draft that does not exist anymore in the IETF archive. Based on the fact that the draft has expired and no other Telnet software supports it, X11-forwarding over TLS is a rarity and interoperability nonexistent.

IPSec — Because IPSec secures all communication, securing X11-sessions is not an issue. X11-sessions get secured without any additional effort or configuration.

5.3.3 User certificates

In the above evaluation, users were authenticated with passwords. SSH and TLS also support user certificate authentication. Certificate authentication provides improved security since a server doesn't need to maintain a top-secret user password database or communicate with a central password server for user authentication. In other words, a server doesn't need to know any user secrets. It just validates user logins with the public CA certificate, and checks the CRL for revocation. This results in better security and scalability. If a server gets compromised, only the server identity is lost and not the password database of thousands of users. The certificate of a compromised server is then inserted into a revocation list, or revocation is done using another PKI mechanism, and a new identity is created for the server. Damage can be done, but not all users are in danger.

However, the deployment of user certificates is a big step towards full PKI. Infrastructure is required for certificate management including creation, validation, revocation and renewal of certificates. These are components of PKI that must be planned with care. PKI infrastructure is not evaluated in this thesis.

IPSec does not authenticate users but hosts. Hence, strictly speaking, user certificates are not supported. With IPSec, applications are responsible for user level authentication, and propably the majority of applications just perform password authentication.

It is good to understand that SSH and TLS server certificates and IPSec host certificates can be deployed without heavy infrastructure. These certificates and their associated private keys can be managed internally by network administrators, so a public distribution mechanism is not required. In addition, usually there are far less server and host certificates than user certificates.

5.3.4 Insecure workstations

Workstations that do not provide proper security mechanisms, such as secure file system, present a high security risk in a multi-user environment with any of these technologies. Most consumer operating systems such as Microsoft Windows 95 and 98 and Macintosh fall into this class.

It is extremely important that the user can securely store private keys and trusted CA certificates. On an insecure workstation this is not possible. There may be prestored CA certificates on disk, but trusting them is foolish since anyone could replace them with fake certificates. The only way to really ensure server identity is to rigorously validate the fingerprint of its certificate, or always carry a personal disk with trusted CA certificates. In practise, these steps are not usually done.

One of the most secure storage for user private keys and CA certificates is a smart card, a small computer with the size of a credit card. A workstation with an attached smart card reader can provide relatively secure user authentication. However, smart card technologies are still under heavy development and standards have yet to be agreed on. Smart card deployment sets requirements for network infrastructure, which makes it a big project both technically and financianally. And if a university decides to install readers on campus workstations and require smart card authentication, how would people abroad make connections to university servers? Security is a complex issue.

Smart card technology has huge potential. It is very likely that smart cards will be used for strong user authentication in the future. All of the three technologies can benefit from smart cards.

The university text terminals and X11 display terminals without disk or processor can not be secured. A processor is mandatory for performing the cryptographic operations. Lack of disk storage would not be a problem if terminals had smart card readers.

5.3.5 Summary

SSH is a good choice for a university environment. SSH provides easy to use applications for secure terminal login and X11-forwarding. SSH is implemented on many platforms and integrates well with the current infrastructure with legacy authentication systems. SSH works without and with PKI. However, key management in large environments is difficult without PKI.

TLS has been integrated into Telnet software, but there are only few implementations. Windows and Macintosh lack up-todate implementations. X11 forwarding is implemented only by one software package. Based on the usage and the number of implementations, SSH is preferred to Telnet over TLS. Server certificate authentication is an advantage that TLS has over those SSH implementations that don't have certificate support.

IPSec provides the best transparency for the users since it doesn't have any visibility. Any Telnet software can be used. IPSec is the most comprehensive solution since it secures not only Telnet but all communication. However, IPSec is not yet supported on all platforms, and is difficult to deploy.

5.4 Scenario 2: Home dial-up user

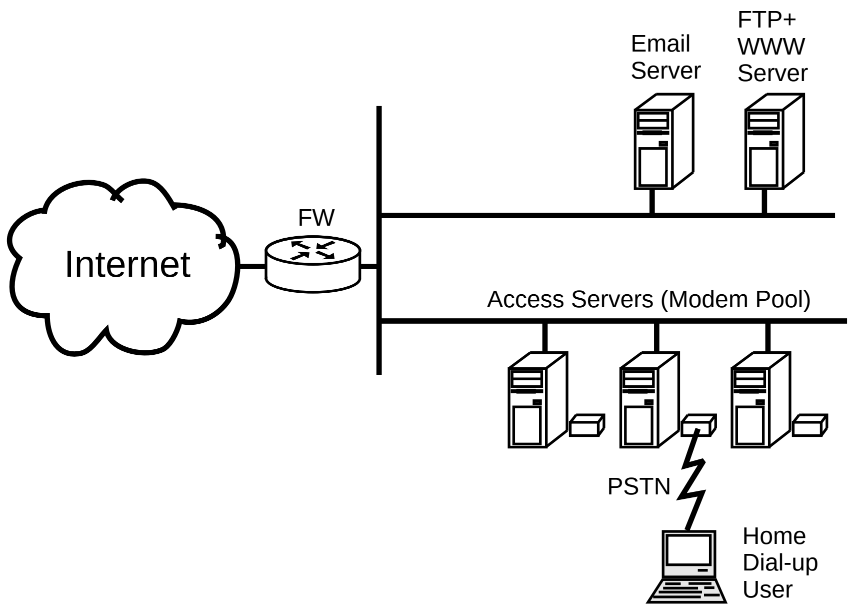

Figure 5.3: ISP dial-up network

An Internet Service Provider, ISP, provides internet access to its users. The user dials in to the ISP access server from home via PSTN or ISDN and logs in to the network by authenticating himself with a username and password. Point-to-Point Protocol, PPP, is being used for authentication and tunneling IP-datagrams over the serial link. A dynamic and public IP-address is assigned to the host. The ISP provides the user an email account and storage for his WWW-home pages.

5.4.1 Task: Secure PPP login

The ISP needs to authenticate the dial-up user when he connects to the ISP access server. The network authentication should be secured.

PPP [26] is a layer 2 protocol that can tunnel any protocol datagram over a single link, typically the link being a serial link over a phone line. Connection setup may include optional user authentication. PPP defines a few authentication methods such as PAP and CHAP [27]. PAP, Password Authentication Procedure, is the original and very insecure authentication method where the user provides his username and password in the clear. CHAP, Challenge-Handshake Authentication Protocol, is an improved method based on challenge-response where the password is not sent over the wire, just a hash value computed over a challenge and a password.

The standard PPP authentication methods are weak. The ISP could disable PPP authentication and require authentication through SSH, TLS or IPSec. In this case, PPP would finish happily and the IP-connection to the ISP access server would be up and running but the server wouldn't route datagrams to the global Internet until the user authenticates himself.

It is important to realize that PPP is not replaced by SSH, TLS or IPSec. PPP is required for transporting IP-datagrams over the serial link. PPP has no encryption capability.

TLS — HTTP over TLS offers a good login solution. Current WWW-browsers readily support TLS and a browser is installed on practically every computer. WWW-interface is very intuitive for most users, and no configuration is required. Users can be directed to the ISP home-page and have enter their username and password securely. The server can be authenticated with a certificate that has been signed by a well known CA whose certificate is already included in the browser.

SSH & IPSec — SSH and IPSec would seem an overkill solution for just the login. They both require software installation and configuration, which can be troublesome for the average user. A server public key must be provided to the SSH user (or a CA certificate if the SSH supports certificates). Host and CA certificates must be provided to an IPSec user. These files should be provided securely, maybe personally at the time of the purchase. To ease configuration, the ISP should provide configuration files for the most popular SSH and IPSec clients that they support.

A login over a dial-up line does not actually need the strongest authentication. A dial-up line is a dedicated point-to-point link between a user and the ISP, and it is quite difficult for anybody to eavesdrop this line. Thus, a simple password authentication over encrypted channel is satisfactory. In this light, SSH and IPSec look like too heavy solutions compared to HTTP over TLS. However, they both have advantages to HTTP over TLS. In addition to PPP login, SSH and IPSec could be used to setup secure tunnels to other ISP services.

Since the IP-address of the user workstation is assigned dynamically, the IPSec host certificate must be bound to the user. Because binding to a user is not clearly standardized, different IPSec implementations may not interoperate well.

5.4.2 Task: Secure email access

Email access from the user workstation to ISP email server should be secured. Simple password authentication is enough. (POP3 and SMTP are used.)

SSH — SSH can secure email access with tunnels but this requires quite a few configuration steps. The SSH client must be installed and configured to create tunnels for SMTP and POP3 protocols and the email client must be configured with "localhost" listed as the destination for SMTP and POP3 server. The configuration can be difficult. While the ISP can provide configuration files for the user, SSH is not as troublefree as TLS. The SSH server would be installed on the same host as the regular email server.

The standard SSH client has one major disadvantage: it provides a terminal window. A terminal is something a common ISP customer has never seen or will never use and which has generally little value for him. While there may be few advanced customers willing to have a terminal access, for strong security reasons it makes sense for the ISP to disable terminal access from the SSH server. But the terminal windows and its functionality is still there in all SSH clients, and this propably confuses the customer quite a bit.

TLS — Most popular email clients, such as Microsoft Outlook Express and Qualcomm Eudora, have TLS built-in and they can run SMTP and POP3, the outgoing and incoming email protocols, over TLS. Good availability and ease of use makes TLS a very attractive solution for email access. Users are typically authenticated with a password. Free sendmail or UW IMAP software can be used as the email server on Unix. Sendmail implements SMTP and UW IMAP implements POP3 over TLS. Email servers can be authenticated through the same CA certificate as the WWW server in the PPP login.

IPSec — IPSec provides transparent security for email access. A host-to-host transport mode IPSec connection is negotiated to the email server which has regular unmodified email server software and IPSec installed, and the client and server hosts are authenticated with host certificates. The email server then authenticates the user with a password.

5.4.3 Task: Secure file upload

File uploads of WWW-files from a user workstation to an ISP file server should be secured.

Why should the transfer of WWW-files be secured, when they are made public anyway? WWW-files are public, that is true, but while anybody is allowed to view them, only the authorized user should be able to modify and update them on the server. Thus we need authentication and integrity for the file transfer. Confidentiality is not necessary, but, if a password is used to login to the file server, the connection should be encrypted.

SSH — SSH provides a good solution for secure file transfer. Due to the bad design of FTP, the creators of SSH decided to implement a new and clean file transfer protocol, and hence SSH has built-in secure file transfer software called SFTP [28]. There is no need to install regular FTP software along with SSH. SFTP provides FTP-like commands, and clients with intuitive graphical user interfaces are available. On the other hand, SFTP is not yet as featurefull as the best FTP software. For example, ASCII-transfer, transfer resume and robust auditing are to be developed.

TLS — The great majority of the FTP implementations do insecure transfers. Only a few implements FTP over TLS, CuteFTP Pro on Windows being one of them. Lack of security is due to the fundamental fact that FTP is a problematic protocol in today's firewalled networking environments due to FTP's separate control and data connections, nonstatic ports, and multiple transfer modes. Integrating TLS into an aged protocol is not easy. Only recently has a draft specification been issued for FTP over TLS [29].

IPSec — IPSec secures file transfers with ease. All of the hundreds of available FTP software can be used securely. IPSec authenticates the hosts for host-to-host transport mode and FTP server authenticates the user with a password.

5.4.4 Summary

TLS provides the easiest solutions for network login and email access. FTP over TLS is not well supported by existing software since a standard is unfinished and FTP has fundamental weaknesses.

SSH is overkill for PPP-login and has features that have no use for common dial-up users. The configuration of secure email access is somewhat difficult. Simple, out of the box, SFTP makes SSH attractive for file transfers.

IPSec requires initial configuration but can provide good transparency. The user has the freedom of selecting any regular email and FTP client software. Transport mode is used for host-to-host connections.

5.5 Scenario 3: Road-warrior VPN

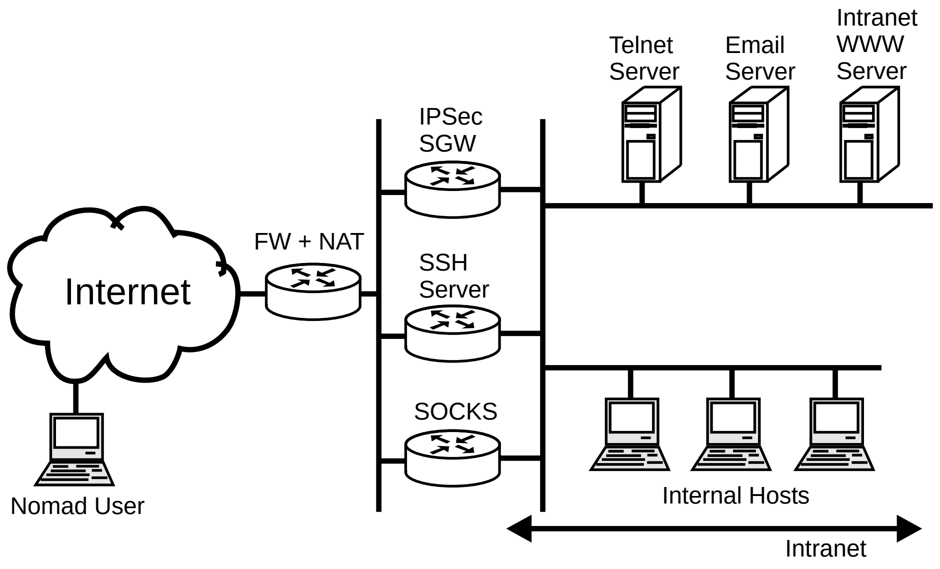

Figure 5.4: Road-warrior VPN network

A company private intranet with a private addressing scheme connects to the Internet via single firewall that performs network address translation. There are several high-security services in the private network that a company employee needs to access with his laptop computer while travelling to customer locations. The IP-address of the laptop varies from network to network. Strong authentication is required.

The intranet has a two-level access control, which is very common. To access the intranet service, a connection must first pass the firewall and then the IPSec security gateway, SSH server or SOCKS server, depending on the technology. SOCKS is installed to provide level two access control for TLS-enabled services in the intranet. SOCKS works like an additional firewall, letting only the configured TCP-connections through to intranet services [30].

5.5.1 Firewall traversal

There are two issues when connecting to the intranet through a firewall. First, the firewall must allows access to the network by opening the appropriate ports for incoming traffic. Second, the security technology must be able to cope with address translations.

Firewall ports

SSH — For SSH, a single TCP-port must be opened for incoming SSH-traffic at the firewall. The standard port assigned for SSH is 22. The firewall relays SSH traffic to the designated SSH server in the intranet.

TLS — TLS has the disadvantage that a number of TCP-ports must be opened at the firewall, one for each desired service. The firewall relays traffic from each port to the configured server in intranet. Opening several holes in the firewall is definately not a good security decision and is against the whole purpose of the firewall. The firewall should be as closed as possible. The more holes, the more trust is put into the services that are made available in the intranet.

IPSec — To allow incoming IPSec traffic to the private network, UDP-port 500 must be opened for IKE and IP-protocols 50 and 51 for ESP and AH protocols at the firewall. The firewall relays IPSec traffic to the designated IPSec security gateway.

The security services could be deployed in the firewall as well, but usually firewalls are stripped down to the minimum, running only the firewall software. Figure 5.4 displays a scenario where security services are deployed behind the firewall.

NAT

Network address translation, NAT for short, is widely deployed in the current networking environments. NAT is the translation of an IP-address from one network realm into another realm. The translation is usually performed by a firewall between a public network and a private network. For outgoing datagrams, the private source IP-address is replaced by a public IP-address, and for incoming datagrams, the public IP-address is replaced by the private IP-address of the receiving host. The NAT box maintains a mapping table for the address translations for each active connection.

In essence, all hosts in the private network share a single public IP-address when communicating with the external hosts on the Internet. The foremost reason why NAT exists is that it conserves global IP-address space. A company needs to have just a single global IP-address. But NAT brings other benefits too. NAT provides security for private networks since the internal network layout is hidden from the public. And NAT makes it possible for a company to freely select its private addressing scheme without dependencies on the addressing schemes of an ISP.

IPSec conflicts with NAT since by changing the source IP-address of a datagram, NAT modifies the IP-header which is exactly what IPSec was designed to detect and prevent. Integrity check fails for a datagram whose IP-header has been modified. Incompatibility with NAT has been a major problem with IPSec, slowing down the widespread deployment of IPSec, but there are solutions to make IPSec work with NAT. Let's see how a solution called NAT traversal works.

In short, NAT traversal is an IETF draft [31], that solves NAT problems by extending the IKE protocol. Instead of passing AH and ESP packets directly in IP-datagrams, they are encapsulated into UDP-datagrams destined to IKE port 500. The IP-header of the UDP-datagram gets modified by NAT, but since AH and ESP packets are in the UDP-payload, their contents are unmodified. In the destination host, IKE receives the UDP-datagram, discards the modified IP-header and sends the unmodified UPD-payload, AH and ESP, to the IPSec engine.

Since there are currently also other competing drafts to address NAT and IPSec interoperability, it will propably take a few years for a common and widely supported standard to emerge.

SSH and TLS do not have problems with NAT since they both operate on top of TCP. NAT operates at the IP-layer and does not modify TCP payloads where SSH and TLS apply their security services.

5.5.2 IPSec VPN

IPSec is the only technology that can provide a true IP-level VPN solution.

Since the other end is an IPSec security gateway, IPSec is used in tunnel mode. In this mode, all traffic from the laptop is tunneled through the firewall and the security gateway and destined to the private network behind the gateway. The laptop thus becomes part of the private network and accesses the network services as if it was directly connected to the intranet. All the IP-datagrams from the laptop that are destined to the intranet are routed into the IPSec tunnel. Other datagrams from the laptop are routed to the foreign network that the laptop is connected to, and are not processed by IPSec.

However, the laptop is not fully part of the intranet because it has an external IP-address that it received from the foreign network. This can be a problem since the services on the Intranet may enforce access control to accept connections only from the intranet, from hosts with private IP-addresses. Hence, there should be a mechanism to assign a local IP-address to the laptop. IPSec provides this mechanism.

It is possible to specify a virtual IP-address that is a permanent and private IP-address assigned to the laptop in the private network. This IP-address is configured in the IPSec software in the laptop. When the firewall receives an IP-datagram over the IPSec connection from the laptop, the firewall decrypts the datagram and changes the source address of the plaintext datagram from the external IP-address to the local IP-address. The services in the intranet then see the datagram arriving from a valid IP-address and accept it. In fact, the services can't detect whether the datagram actually originated from inside or from outside the intranet.

To route the datagrams back to the laptop, the firewall must act as an ARP-proxy. It advertises itself as the owner of the laptop's IP-address in the intranet.

There are two drafts that propose mechanisms for dynamic configuration of the virtual IP-address and other configuration data to the remote host during the connection setup. The "ISAKMP Configuration Method" [32] -draft proposes a simple extension to IKE to pass configuration data to the remote host. "DHCPv4 Configuration of IPsec Tunnel Mode" [33] -draft proposes a solution that does not require IKE to be modified and is based on standard DHCP.

IPSec connection ends up in the security gateway and the datagrams in the intranet are in plaintext. There is no end-to-end security. To have true end-to-end security from the remote host to a host in intranet, IPSec provides nested tunnels. For example, we may have one IPSec connection to a firewall and inside it another IPSec connection to a host in the intranet. However, current IPSec implementations do not support nested tunnels, due to technical problems and difficult configuration. Nested tunnels may have more importance in the future when IPSec usage gets more widespread.

5.5.3 Task: Secure terminal access

Let's say the nomad user is an administrator who needs terminal access to servers in the intranet while travelling. Terminal access to the intranet should be secured.

An administrator remote login with terminal access is a high security risk for a company. A terminal is a very powerful tool to which access should be controlled with strong user authentication.

SSH — With SSH, the administrator connects to the company firewall with the SSH client. The firewall relays the connection to the SSH server which authenticates the administrator. SSH provides strong user authentication. The SSH server can be configured to require multiple authentication methods. For example, both password and certificate authentication could be required. Or the administrator could have a SecurID electronic card, which is quite a popular hardware authentication token. Upon entering a correct PIN code, the SecurID card displays a secret number that is bound to the current time. The secret number is sent over the secure connection to the SecurID server for validation, and the administrator is authenticated if the secret number matches the one calculated by the server. Authentication is thus based on two factors: what you know, PIN, and what you have, the card.

Once authenticated, the administrator is given terminal access to the SSH server. He is free to do his administration tasks with the tools that are installed on the server. He can initiate further SSH connections to other servers in the intranet with the command line SSH applications.

TLS — With TLS, the connection from a TLS-enabled Telnet client in the laptop is relayed through the dedicated port in the firewall and the SOCKS server to the TLS-enabled Telnet server in the intranet. The Telnet server authenticates the user. Password authentication may be sufficient for connections that originate from inside the intranet, but for access from the public internet, a password is too weak of an authentication method. Administrator level remote login demands better authentication. Thus, user certificates should be used for authentication. However, while the two Unix Telnet clients support user certificate authentication, Teraterm client on Windows does not.

A fundamental weakness of the TLS protocol is the fact that user certificatea are sent as plaintext. The identity of the user is thus revealed to a malicious attacker who may be eavesdropping the connection. For high-security terminal access, this is not acceptable.

IPSec — With IPSec, the user connects to the firewall which relays the connection to the security gateway which authenticates the user. Certificate is used for authentication rather than a preshared secret because of better security and easier management. Since the IP-address of the laptop varies from network to network, the certificate must be bind to a user instead of to IP-address or DNS-name of the host.

Once authenticated by IPSec, the user can access all services in the intranet as if he was directly connected to it. Traffic in the intranet is plaintext. To have terminal access, the user connects to a regular insecure Telnet server. Telnet client has the same configuration as when used locally in the intranet. Telnet server authenticates the user with password.

To have end-to-end security, IPSec is also installed to the host that is running the Telnet server, and nested IPSec tunnels are configured in the laptop. However, as was explained above, nested tunnels are not yet supported by existing IPSec software.

5.5.4 Task: Secure email access

The nomad user needs to read and send his emails with his graphical email program that is installed in his laptop. Email exchange with the company email server should be secured.

SSH — With SSH, the SSH client is used to connect to the firewall. Two TCP tunnels are configured to the desired intranet email server over the single SSH connection, one for SMTP and one for IMAP. Both SSH client and email client software need to be configured, just like was explained in the dial-up scenario. The first half of the SSH tunnels from the laptop to the SSH server is encrypted and the second half from the SSH server to the email server is in plaintext. The email client is launched after the SSH connection has been set up. SSH authenticates the user with strong methods and email server just with password.

End-to-end security to the email server is practically infeasible with SSH. Technically it is possible to have the SSH server installed also on the email server, and run SSH over SSH, first SSH connection to the SSH server behind the firewall, and the second SSH connection over an SSH tunnel to the second SSH server running on email server. However, this requires running two SSH clients simultaneously on the laptop. Three user authentications are required: two for the SSH servers and one for the email server. This makes the solution impractical.

If only email access is desired, terminal access should absolutely be disabled. This is possible with SSH. However, if terminal access is wanted for the administrator, then the terminal is available for everybody else too. While SSH does provide generic access control per user, it could be more fine-grained. Use of services like terminal and file transfer should be controllable per user. Currently, it is common that the services are either provided for all or nobody. The lack of flexibility of access control is a missing feature of current SSH implementations. In addition, SSH servers need better control for tunnel creation. The administrator should be able to configure what kind of tunnels the server allows the user to set up. Currently this applies only to X11 tunneling, which can be enabled or disabled.

TLS — With TLS, the TLS-enabled email client connects to the email server through the dedicated two email ports in the firewall and the SOCKS server. The problem with TLS is that current email clients do not support user certificate authentication. Thus, we are restricted to simple password authentication, which is unsatisfactory.

IPSec — With IPSec, the user is strongly authenticated by the security gateway. The standard email server authenticates the user with password. The email client requires no additional configuration. In the future, nested tunnels could be used to provide end-to-end security from the laptop up to the email server.

5.5.5 Task: Secure intranet WWW access

Access to the intranet WWW server should be secured.

SSH — With SSH, the SSH client is used to connect to the company firewall. One TCP tunnel is configured to forward HTTP traffic from the laptop to the WWW server. The WWW-browser on the laptop is configured to connect to the local host. The user is authenticated strongly by the SSH server and the Intranet WWW server does no authentication.

TLS — With TLS, a regular browser is used to connect to the firewall. The firewall relays traffic through SOCKS to the intranet WWW server. The user must be authenticated strongly with user certificates. Popular browsers have built-in support for user certificate authentication.

Since user authentication is not required for connections from inside the intranet, there can be two instances of WWW servers running, one with authentication and one without. The SOCKS server would relay HTTP over TLS traffic to the one that requires authentication.

IPSec — With IPSec, the user is strongly authenticated by the security gateway and no authentication is thus required from the intranet WWW server. The browser requires no additional configuration.

5.5.6 Task: Secure voice calls over IP

The nomad user uses voice over IP software to make phone calls to his secretary in the office. Phone calls should be secured.

SSH & TLS — Since voice over IP commonly operates over UDP, SSH and TLS can't be used. They both require and run over a reliable transport stream such as TCP. While it would be technically possible to tunnel UDP over TCP, it is not implemented by SSH.

IPSec — IPSec secures UDP based protocols. The voice over IP software connects to the secretary's host. Since phone calls can be quite sensitive, it is desirable that the IP-traffic in the intranet is encrypted to prevent other employees from eavesdropping on the call. Before IPSec nested tunnels are implemented, we have to rely on the voice over IP software to provide such security in the intranet.

5.5.7 TLS tunneling

It is fair to mention that there exist TLS based products that make it possible to tunnel a single or multiple insecure TCP-connections over a single secured TLS-connection, similar to what SSH does. One such a product is Stunnel. TLS tunneling makes it possible to secure insecure applications without integrating TLS in them. However, TLS tunneling has not been covered in the evaluation for the reason that TLS tunneling of multiple TCP-connections is not standardized by the TLS-specification. Tunneling products from different vendors may not interoperate. In addition, having an external tunneling software would negate the ease of use benefits that integrated TLS brings to an application.

5.5.8 Summary

IPSec is a winner in providing a VPN connection to a private network. IPSec secures all IP-traffic and provides access to the whole network and not just point-to-point connections to the selected hosts like SSH and TLS do. Client applications need no configuration; they have the same configuration as when used locally in the intranet. Incompatibility with NAT has been a major problem but solutions are available and being standardized.

SSH provides simple TCP-level VPN functionality through tunnels. This can be sufficient in scenarios where the number of accessed services is small. Applications need no modifications but they must be configured to use SSH tunnels. Currently SSH provides the most flexible set of standards-based user authentication methods.

TLS has several weaknesses. TLS must be integrated in each client and server software. And even if TLS has been integrated in a software, not all of them implement user certification authentication. Each service being accessed requires a port to be opened in the firewall. A security risk exists due to the lack of confidentiality in user authentication by TLS.

IPSec and SSH both provide security services in a central location, separate from the application services. This is good from the security and management point of view. The disadvantage of TLS is that security services must be implemented in each service. It is hard to guarantee security in an environment with widely distributed security.

Both SSH and TLS lack support for UDP based protocols.

Chapter 6

Analysis

In this chapter, the results of the technology evaluation are evaluated. The evaluation criteria are reviewed one by one and an analysis is presented on how well the technologies fullfill the criteria.

Then, the pros and cons and suitability of each technology are discussed and an analysis is given about providing security in each layer of the TCP/IP protocol stack. Future improvements for SSH are also suggested. Finally, a look is taken into the future of each technology.

6.1 Results

The evaluation results in respect to the criteria are analyzed here. Refer to Appendix C for a concise list of the evaluation results.

6.1.1 Security criteria

Confidentiality and integrity

Confidentiality and integrity are fundamental requirements for a secure remote access technology. A detailed cryptoanalysis of the algorithms that provide these services is outside the scope of this thesis. Here I present an overview of the strengths of the confidentiality and integrity services.

All of the technologies provide strong confidentiality and integrity. First, the protocols are not hardcoded to particular cryptographic algorithms;

they can take advantage of the best algorithms available. When a given algorithm is found to be flawed, it can be removed from the protocol. Similarly, when a new algorithm is invented, it can be integrated into the protocol without modifying the specification. Second, the protocols and most of the algorithms are public and developed and analysed internationally by the open security community. The best minds in the world have contributed to the development of the protocols. In this respect, all three technologies are trustworthy.

For interoperability reasons, typically a few algorithms have been specified as mandatory to implement. IPSec specifies DES as the mandatory symmetric algorithm, TLS and SSH specify 3DES. DES is a very old algorithm and is considered to be insecure by today's standards. 3DES provides good security. In practise the products implement a variety of newer algorithms that provide strong security. This year, US government selected AES, Advanced Encryption Algorithm, as the future encryption algorithm to be used for high-security communication. AES is already implemented in the evaluated SSH and IPSec products. It is expected that the new algorithm will be implemented in more products.

SSH has provided strong algorithms with at least 128 bits key length since its creation. Due to US export regulations, the TLS specification still defines weakened algorithms with key lengths of only 40 bits. This has caused a bad reputation for TLS outside US. Fortunately, TLS defines the 128 bit versions as well. It is recommended that most TLS implementations should disable the use of 40 bit keys.

All of the technologies specify HMAC-SHA1 and HMAC-MD5 to be used for integrity checks. Both of these algorithms have been subject to detailed analysis by the security community and are considered to be very strong.

Perfect forward secrecy is a very important concept and is provided by the Diffie-Hellman key exchange protocol in IPSec and SSH. TLS specifies the Diffie-Hellman key exchange as well, but most of the implementations still use RSA key exchange which does not provide perfect forward secrecy.

In summary, all of the three technologies can provide strong data confidentiality and integrity as long as the selected algorithms are strong and trustworthy.

Authentication

All of the technologies provide mutual, two-way, authentication. SSH and TLS provide user and service authentication, and IPSec provides authentication of hosts.

SSH is very good at user authentication. SSH offers the largest and strongest set of authentication methods, and the protocol allows easy implemention of new methods. SSH fits well into existing network infrastructures with legacy authentication methods. X.509 certificates provide a scalable authentication framework.

TLS has the most limited authentication. Authentication is heavily married to X.509 certificates. Other authentication methods have been proposed but are not really implemented. In addition, TLS user certificate authentication is most often not implemented or is disabled, and applications use weak password authentication instead. The lack of confidentiality in user certificate authentication is a concrete design error of TLS.

IPSec provides strong host authentication based on X.509 certificates but lacks standard user and service authentication. Various user authentication methods have been proposed but smooth integration of these still requires much standardization effort.

Denial of service

SSH and TLS do not provide protection for denial of service attacks. Regular attacks against TCP/IP, like TCP SYN flooding, can be applied to a host running SSH and TLS based software since TCP/IP-protocol stack operates normally.

IPSec provides protection for regular denial of service attacks by applying cryptographic processing to the IP-datagrams at the earliest moment possible. The delivery of false datagrams can be prevented. Normal TCP/IP services can be disabled on an IPSec protected host. However, there are and will be new kinds of denial of service attacks against IPSec.

6.1.2 Usability criteria

Ease of use

For people with Unix competence and strong networking background, SSH applications are easy to use since they operate similarly to their insecure counterparts. For most people, however, SSH is not very intuitive. This derives to a large degree from the fact that the primary function of SSH is to provide a remote terminal service, which is an unfamiliar concept to most computer users. SSH is very good in providing terminal service but as a generic security solution, SSH lacks ease of use compared to TLS and IPSec. Usage of SSH requires more user education.

Also, the spirit of SSH applications is that everything is open for configuration. This may not be a good approach when providing the software for the average user.

TLS is usually completely hidden in the application and has little visibility to the user. There is very little to configure, usually just an option to control whether to use TLS or not. Minimum configuration is good from the usability point of view.

IPSec provides the best ease of use. The initial configuration process by the administrator can be laborous, but once its done, no application configuration by the user is required. The less configuration, the better. Interaction with the user is minimal.

Transparency

SSH is the least transparent security solution. After all, SSH is an application that must be explicitly started to achieve security. Transparency is not an issue for remote terminal and file transfer services since they are built into SSH. Lack of transparency appears when SSH is used to secure other application protocols. For example, securing email access requires running two applications: SSH first and an email client second, with two user authentications. This is not very transparent security.

TLS and IPSec both provide good transparency of security. The user doesn't have to perform extra steps to get security.

Since IPSec is intended to be fully transparent to applications, it is technically difficult to provide meaningful feedback to the user about IPSec connectivity problems. When error reporting is unadequate, it is difficult to solve connectivity problems.

In addition, if the security services have no visibility to the user, how can the user assure that his communication is secure? No transparency is bad but full transparency is not always desirable either.

Setup

SSH deployment is easy in small environments. It installs effortlessly onto many platforms and works out of the box with default settings. In large environments, distributing all server public keys out-of-band into client hosts can be laborous. Certificates can solve this problem, but on the other hand, setting up a PKI can negate the ease of deployment benefit of SSH.

Securing other applications with SSH tunnels requires configuration effort.

Applications with integrated TLS require little setup in addition to the application setup. However, CA certificates and possibly user certificates need to be created and distributed to the client applications.

IPSec setup is the most complex because IPSec is configured to secure all traffic, not just one application. Since IPSec installs parts of it into the kernel, there is a danger that something may break in the system. Troubleshooting IPSec setup and connections can take time and resources.

6.1.3 Interoperability criteria

Portability

Both SSH and TLS are very portable as protocols. As a product, the SSH server does have some porting issues because it interfaces with the operating system to provide the login services, which vary from system to system. However, much effort has been done by many to port SSH to numerous platforms. Wide platform support indicates that SSH is an important technology that is worth porting.

The portability of TLS depends on the application being ported. Problems arise with the particular product implementing TLS, and not from the protocol itself.

IPSec is the most troublesome to port because it integrates partly into the TCP/IP protocol stack which is part of the kernel of the operating system. The implementation of the kernel portion varies greatly from platform to platform.

Interoperability

SSH implementations are quite interoperable. This is greatly due to the IETF standardization work. The essential items of SSH have been standardized well, minimizing false interpretations of the standard among vendors. SSH1 and SSH2 do not interoperate on a protocol level by design.

TLS has a clear and simple standard specification, which indicates good portability. However, user authentication is not implemented in all applications that use TLS.

IPSec is an immature and rapidly evolving technology with dozens of draft specifications under construction. It is thus no surprise that IPSec has the biggest interoperability problems. Many vendor-specific extensions are being pushed to IPSec products, which does great harm to interoperability.

X.509

All of the technologies support the use of X.509v3 certificates for authentication. In fact, TLS does not accept other certificate formats than X.509v3. SSH and IPSec are not tied up with X.509v3 only, but in practise only X.509v3 is supported since it is the dominant certificate infrastructure.

X.509v3 support for SSH is currently available only with SSH version 3.0 from SSH Communications Security Corp. Support for certificates for SSH has been a long-awaited feature since it provides protection for man-in-the-middle-attacks and helps address key management tasks.

Firewall traversal

Firewall traversal has been a major problem for IPSec because of the IP-address mappings performed by NAT. Solutions are available, but a common standard is still to emerge.

SSH and TLS are not affected by NAT since they both operate on top of TCP. SSH requires one TCP-port to be opened in the firewall, TLS requires one TCP-port per each service.

6.1.4 Versatility criteria

Versatility

IPSec is the most versatile technology with the largest scope of applicability. IPSec can be applied between hosts and between security gateways. IPSec secures all networking applications and provides both host-to-network and network-to-network VPN connections.

SSH is best in securing remote terminal and file transfer, but can secure other application protocols as well. SSH and TLS only provide point-to-point connections from client to server. They are not deployed between gateways. UDP based protocols are not supported.

Scalability

All technologies achieve scalability mainly due to certificate support. Certificates ease the management of an authentication framework in a large network environment. SSH without certificates scales badly, just like IPSec does with preshared secrets.

On a host level, IPSec has inherent scalability since it is a one-shot solution securing all applications on a host. SSH can also secure many applications. TLS has the least scalability on a host level in the sense that all applications must be integrated with TLS separately.

Performance

All technologies provide about the same level of performance. The cryptographic algorithms being used dictate most of the performance. The protocols have insignificant performance differences.

To be precise, IPSec can perform a bit faster than SSH and TLS since the IPSec engine operates in the kernel and is not subject to process context switches between the kernel and user mode. There are also hardware accelerators available for IPSec engines.

6.2 Discussion

A concise list of the most important benefits and weaknesses of each technology is presented below.

6.2.1 SSH pros and cons

+ Remote terminal and file transfer. SSH is very good at providing remote terminal and file transfer services. These are the primary tasks of SSH. The evaluation revealed that there is little competition in these services, which proves that SSH handles them well.

+ User authentication. SSH provides the largest and strongest set of user authentication methods, from simple passwords to full-blown PKI. This makes SSH fit into current networks with legacy authentication systems, and in future networks with modern authentication systems. Standardized authentication methods provide interoperability.

+ Tunneling. Tunnels enable SSH to provide security for other application protocols as well. This is an important feature in the current world where most application protocols are still insecure.

− Ease of use. SSH applications are easy to use for people with networking background, but for the great majority of people there is a learning curve. The remote terminal is an unknown and unnecessary service for the masses.

− Application specific. SSH is a specific application protocol and lacks generality.

6.2.2 TLS pros and cons

+ Application independency. TLS is a good generic security protocol for securing any application protocol running over TCP.

+ Ease of use. TLS integrates smoothly to an application and works transparently on the background.

− User authentication. User authentication is quite limited. Too much emphasis on X.509. Currently applications are forced to implement their own simple user authentication methods. Need exists for other authentication methods.

− User certificate confidentiality. Due to design error in the protocol, there is no confidentiality provided when a user authenticates with a certificate.

6.2.3 IPSec pros and cons

+ Transparency. IPSec is the most transparent security solution. IPSec operates in the background with little interaction with the user. No application configuration required.

+ Versatility. IPSec has the widest scope of applicability. IPSec secures all traffic between hosts or between gateways. It is the best technology for providing VPNs.

− Complexity. IPSec is the most complex technology. It evolves at a rapid pace and standards are under construction. There are dozens of different draft specifications.

− Interoperability. Due to lack of standards or their misinterpretation, vendors extend IPSec into various directions. Vendor specific extensions harm interoperability and slow down widespread deployment.

− Lack of user and service authentication. IPSec authenticates hosts, and does not provide mechanisms for user and service authentication, which are essential services as well. Standards are required.

6.2.4 Suitability

It can be roughly said that SSH provided the best solution for the university, TLS for the dial-up and IPSec for the road-warrior VPN scenario.

SSH is the dominant technology for providing secure remote terminal and file transfer. Whenever there is a need for these two services, SSH is a good choice. SSH fits well into old heterogenous environments since it has been ported to numerous platforms and many user authentication methods have been implemented for interfacing with legacy authentication frameworks.

SSH tunnels are of great importance. There are many insecure applications today and will be for a long time since the applications won't be secured overnight. There is a clear need for a simple generic mechanism for providing security services into existing applications. SSH tunneling is a such a mechanism.

In the past, SSH has been mainly targeted to networking professionals, but today the target user group also includes more novice users. Use of SSH requires some education, but products are getting easier to use thanks to graphical user interfaces that are available.

TLS is a great technology for securing any application protocol. If an application developer wants to the secure his new application, TLS is a natural choice. Since TLS is integrated into applications, the end user doesn't have to worry about having separate security mechanisms. Ease of use of integrated security is an important benefit. This is especially true in the dial-up scenario where most users are novices to networking.

On paper, IPSec is a comprehensive technology that solves many security problems in networking. However, in the current world IPSec is best deployed in a closed company environment where it is possible to control the selection of software and network architecture. Interoperability problems prevent using IPSec in large multivendor environment.

IPSec is a good choice when there are a multitude of existing insecure applications that need to be secured. IPSec can secure them all in one shot. As a VPN technology, IPSec is superior to SSH and TLS.

It is good to realize the immediate targets of each technology. The immediate target for SSH is an end user since SSH is a ready to be used application. The immedate target for TLS is an application developer who wants to secure his application. And finally, the immediate target for IPSec is an operating system or network hardware or software vendor who integrates IPSec into the TCP/IP protocol stack.

6.3 Security in layers

This thesis evaluates three technologies which operate at different layers of the TCP/IP protocol stack. Let's analyze the advantages and disadvantages of providing security in each layer.

SSH is an application layer protocol that secures only one application, SSH. It is not applied as a generic security protocol in other applications. TLS is a transport layer protocol that can secure all applications that communicate over TCP. IPSec is a network layer protocol that can secure all IP-traffic. (See layering in Figure 4.1.)

(Actually, SSH is not purely positioned as an application layer solution. SSH also provides generic transport layer services since it can secure other applications with tunnels. For the purpose of this analysis, however, let's consider SSH as an application layer solution.)

When security is implemented in the application level, the application has full control of the security services, complete access to data, and understanding of the user context. Services can be tailored to fit perfectly with the needs of an application. In the case of SSH, the three defined subprotocols are designed to provide exactly those services that are essential in secure remote terminal access.

Other popular application level security solutions are S/MIME [34] and PGP [35] for securing the delivery of email messages. They are not online protocols but rather mechanisms that protect messages. They have a crucial advantage to the evaluated remote access security technologies: they can both secure messages not only in transit but in storage as well. SSH, TLS and IPSec only secure data while it is in transit between hosts. In addition, SMIME and PGP can provide non-repudiation of messages.

Application level security solutions provide independence from external systems. An application doesn't need to rely on the operating system to provide security services. The developer can assure that his application is secured on every platform.

The disadvantage of application level security is that the common security services such as confidentiality, integrity and authentication get implemented over and over again in each application. Security protocols are complex and their development takes time. It would become impossible to validate the soundness of each application level security protocol. Thus, it makes perfect sense to have a common and generic transport level security protocol such as TLS that can be applied in many applications, instead of reinventing the wheel for each application.

When security is implemented in the transport layer, the application doesn't need to know the details of the security mechanisms. This is good for most applications. Applications can be secured with ease. On the other hand, the application is limited to use only those security services that are available by the transport layer solution. The application can build its additional security services on top of the transport layer but it cannot change the way core services are applied by the transport layer security solution.

Another disadvantage of transport layer security solution is that the applications still need modifications; TLS must be integrated into each application.

The network layer is the lowest layer that extends from end to end. Hence, implementing security in this layer seems to offer the most generic solution. Network layer security solution can transparently secure any protocol running above IP and any medium IP runs over. Security is applied only once and not per each application. However, the problem with this low level solution is that it is difficult to provide security services for the application layer. In theory, the network layer cannot provide any services for the application layer since it is not immediately above the network layer.

Finally, the link layer is not feasible for implementing security since it wouldn't provide end to end security.

In summary, the higher the security solution is in the layers, the easier it is to deploy, the more flexible it can be, and the more specific services it can provide for the application. On the other hand, the higher the solution, the less generality and transparency it has.

Network or transport layer solutions alone won't solve all our security problems. Security mechanisms are required on the application level as well.

6.3.1 SSH vs TLS

IPSec is technologically very different from SSH and TLS, but the latter two have similarities. SSH and TLS protocols perform some of the same tasks; they both provide a secure transport for an application protocol. In case of SSH, the application is SSH and in case of TLS, the application can be anything.

The SSH protocol can be considered to be a superset of TLS. The SSH transport layer subprotocol is quite equal to TLS. They both provide a secure channel for upper layers. In addition, SSH also defines user authentication and connection layer subprotocols that run on top of the SSH transport layer. These two subprotocols provide user authentication methods, channel multiplexing, generic TCP tunneling, X11 tunneling and authentication agent proxies. There is much functionality in these layers. The benefit of SSH is that it has standardized these functionalities. Standardization results in better interoperability.

Why then there are two so similar transport protocols? Reasons are historical. The protocols have had different roots. TLS, or SSL as it was called back then, was a proprietary technology privately developed by Netscape to secure WWW-access. SSH was publicly developed to secure remote terminal logins. SSL has a commercial background and SSH an academic background. The development of each took place in opposite parts of the world and neither one of the protocols was being standardized at the time.

SSH might run on top TLS, had it been invented long after TLS, and had there been no export restrictions for TLS. In fact, in the first official meeting of the SSH IETF working group, there was discussion about adopting TLS as the transport to SSH version 2.0 [36]. But this never took place, for technical and political reasons.

6.4 Secure Shell development

The evaluation revealed some shortcomings in SSH. The shortcomings are listed here and improvements are proposed to solve them. Only the UDP tunneling is a protocol improvement, others are product improvements.

UDP tunneling

SSH can't tunnel UDP based protocols over the SSH connection. Technically this is possible. SSH connection layer protocol can be extended to enable tunneling of UDP-datagrams. There are several important UDP based application protocols where tunneling over SSH could be beneficial, like NFS, NIS and DNS, to name a few.

File transfer

SSH secure file transfer is simple and reliable but it lacks features when compared to the best FTP software. It should be enhanced to match the insecure counterparts.

File transfer clients should implement ASCII/binary-transfer, which is currently implemented only in the SSH version 3.0 Windows client from SSH Communications Security Corp. This feature is important for ASCII file transfers between Windows and Unix. Transfer resume is a feature that would allow transfer resumption of a partially transferred file.

The file transfer server should implement robust auditing. No auditing of transferred files is currently performed. High security sites need to keep track of who transfers what files.

Ease of use

SSH is not the most intuitive to use for novices. This is to a large degree due to the terminal window and its configuration. Terminal is an unfamiliar and unnecessary feature for most people. It would make sense to have a stripped down SSH client with hidden or removed terminal features for those usage scenarios where an SSH client is used to transfers files or tunnel other application protocols without terminal access.

Configuration of tunnels is difficult for the average user. The SSH client for Windows could provide help in tunnel configuration by automatically modifying the network settings of the most popular client applications. However, this has dangers. It is difficult to know the configuration data structure of external applications.

Access control

Access control for the SSH server could be controllable on a more granular level. There could be mechanisms to control access to specific SSH services, login, terminal, file transfer and tunneling, on a per user or per group level. Tunneling configuration could include the list of allowed ports and hosts. User authentication methods could be specified per user or group (administrator access would require a certificate, others could login with password, for example).

PKI

The only version supporting X.509v3 certificates and PKI environments at the time of the evaluation is SSH version 3.0 from SSH Communications Security Corp. Support for PKI is a very important feature which scales SSH for large environments. In order for SSH to gain further success as a technology, certificate support should find its way to other SSH implementations as well.

Of paramount importance is also the standardization of the usage of certificates in SSH. The interpretation of fields in certificates should be specified to enable interoperability among SSH implementations.

6.5 Future

In the near future, deployment of all three technologies will increase. Security awareness in the Internet community is growing and there is a strong demand for security technologies. All three technologies are under active development and standardization effort; none of them will disappear in the near future.

SSH and TLS have been implemented in products already for many years, and they have a huge installation base on the market. IPSec based products are still on their way to mass-markets. SSH is a mature technology that works well in current networking environments. The next big step for SSH is the transition to PKI environments with certificate based authentication.

An interesting question is how far SSH can fly. There will be increasing need for secure terminal access, but since this service is not meant for the masses, the biggest usage growth of SSH comes from other services that SSH can offer. SSH secure file transfer can be the service that will increasingly attract the mass-markets. In the file transfer arena, SSH leads the way and there is little competition. FTP over TLS has not been productized on a wide scale, or even clearly standardized.

TLS will find its way to most popular applications. New applications will have TLS integrated by default and the standard will be enhanced to include new user authentication methods. It is important that there is a standards-based security technology such as TLS for securing generic TCP-connections. As a generic security solution, the biggest competitor to TLS seems to be IPSec.

IPSec has huge potential to become the dominant security technology of the future. IP version 6 already specifies IPSec as the chosen security solution. IPSec is an excellent technology, but currently there are too many unsolved problems. The main problem of IPSec can be summarized in one word: complexity. The technology is not yet mature. It will take time until standards stabilize, vendors integrate IPSec into their systems, and we will have robust and interoperable IPSec products.

In the distant future, IPSec, or another solution built-in to IP, will replace SSH and TLS. Most of the security services will be provided in the TCP/IP protocol stack, and applications will access the services through clearly defined interfaces. Until we get there, all of the three technologies have their use.

Chapter 7

Conclusions

In this thesis I have evaluated three software technologies that provide secure remote access over an insecure communication channel such as the Internet. The need for these technologies exists because the core communication protocols used in Internet do not provide any security services for transmitted data. There are several kinds of attacks that enable unauthorized access to data if the data is not secured. The evaluated technologies provide the three necessary security services: data confidentiality, data integrity and authentication for both data and entities. The security services are built from cryptographic protocols and algorithms, which rely on difficult mathematical techniques.

I selected three popular and widely deployed technologies for the evaluation. SSH is an application that provides secure terminal access and file transfer. TLS is a generic transport level protocol that can secure any application protocol running over TCP. IPSec is a network level security framework that can secure everything over IP.

The evaluation was conducted by defining three scenarios with real-life remote access tasks, and applying each technology in the scenario. The pros and cons of each technology were analyzed through criteria that was categorized into four main criteria: security, usability, interoperability and versatility.

SSH is very good at providing secure terminal access and file transfer services. SSH is portable and easy to deploy in current networking environments. User authentication is the most versatile in SSH. Legacy user authentication methods are supported as well as modern methods based on certificates. The third important service SSH offers is TCP tunneling which enables SSH to secure other insecure applications and provide simple VPN functionality. However, use of tunnels requires user education.

As a generic security solution, SSH lacks ease of use.

The applications that have TLS integrated are usually easy to use. Integrated security provides good transparency for the end user. This is the biggest advantage of TLS. The two most popular applications using TLS are WWW-browsers and email software. Technologically, it is important that there is a generic and standard security protocol such as TLS for securing any application protocol. The weaknesses of TLS include the lack of VPN functionality, the limited set of standardized user authentication methods, and lack of confidentiality in user certificate authentication.

IPSec is the most complete security technology since it can secure all applications without requiring any modifications to them. IPSec is not visible to the end user and thus provides the best transparency. As a VPN technology, IPSec is superior to SSH and TLS, providing IP-level tunneling. On the other hand, IPSec is also the most complex security technology. The standard is a moving target and interoperability is weak due to vendor-specific extensions.

In conclusion, the technologies can be summarized with respect to the main criteria as follows. All of the three technologies provide about the same level of security since they are built from standard cryptographic techniques, and the development goes through public scrutiny. TLS and IPSec provide the best ease of use. SSH provides the best interoperability in current network environments and among other implementations. IPSec provides the most versatile security solution.

A few shortcomings were identified in SSH. The technology could be improved by providing UDP tunneling, enhanced file transfer service, more fine-grained access control, better ease of use, and PKI support.

IPSec has the best potential of becoming the dominant remote access security technology in the future. It is possible that SSH and TLS will be replaced by IPSec. However, before we get there, IPSec needs to be extended to provide the necessary security services, such as user authentication, to the application layer. A network layer security solution has crucial benefits but security services are required on the application layer as well.

Since all technologies have their advantages, they are all being actively deployed. Currently none of the technologies alone can satisfy all of the the remote access security needs.

Bibliography

[1] W Stallings. *Network and Internetwork Security Principles and Practise*. Prentice Hall, 1995.

[2] Anonymous. *Maximum Security*. Sams.net, 1997.

[3] S.M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *Computer Communication Review, Vol. 19, No. 2, pp. 32-48*, 1989.

[4] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a Denial of Service Attack on TCP. *IEEE Symposium on Security and Privacy*, 1997.

[5] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[6] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, 1996.

[7] RSA Laboratories. RSA Laboratories FAQ 4.1. http://www.rsasecurity.com/rsalabs/faq/index.html, 2000.

[8] Douglas R. Stinson. *Cryptography Theory and Practice*. CRC Press, 1995.

[9] R. Rivest. RFC 1321: The MD5 Message-Digest Algorithm, 1992.

[10] National Institute of Standards and Technology. SHA1, Secure Hash Standard. http://www.itl.nist.gov/fipspubs/fip180-1.htm, 1997.

[11] H. Krawczyk, M. Bellare, and R. Canetti. RFC 2104: HMAC: Keyed-Hashing for Message Authentication, 1997.

[12] E. Rescorla. RFC 2631: Diffie-Hellman Key Agreement Method, 1999.

[13] N. Doraswamy and D. Harkins. *IPSec, The new security standard for the Internet*. Prentise Hall, 1999.

[14] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999.

[15] T. Dierks and C. Allen. RFC 2246: The TLS Protocol, 1999.

[16] D Barrett and R. Silverman. *SSH The definitive guide*. O'Reilly, 2001.

[17] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. Internet-Draft: SSH Protocol Architecture, 2001.

[18] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. Internet-Draft: SSH Authentication Protocol, 2001.

[19] C. Newman. RFC 2595: Using TLS with IMAP, POP3 and ACAP, 1999.

[20] A. Medvinsky and M. Hur. RFC 2712: Addition of Kerberos Cipher Suites to TLS, 1999.

[21] D. Taylor. Internet-Draft: Using SRP for TLS Authentication, 2001.

[22] J. Postel. RFC 790: Assigned numbers, 1981.

[23] S. Beaulieu and R. Pereira. Internet-Draft: Extended Authentication within IKE (XAUTH), 2001.

[24] M. Litvin, R. Shamir, and T. Zegman. Internet-Draft: A Hybrid Authentication Mode for IKE, 2001.

[25] Michael Boe and Jeffrey Altman. Internet-Draft: TLS-based Telnet Security, 2000.

[26] W. Simpson. RFC 1661: The Point-to-Point Protocol (PPP), 1994.

[27] W. Simpson. RFC 1994: PPP Challenge Handshake Authentication Protocol (CHAP), 1996.

[28] T. Ylonen and S. Lehtinen. Internet-Draft: SSH File Transfer Protocol, 2001.

[29] Paul Ford-Hutchinson, Martin Carpenter, Tim Hudson, Eric Murray, and Volker Wiegand. Internet-Draft: Securing FTP with TLS, 2001.

[30] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928: SOCKS Protocol Version 5, 1996.

[31] M. Stenberg, S. Paavolainen, T. Ylonen, and T. Kivinen. Internet-Draft: IPsec NAT-Traversal, 2001.

[32] D. Dukes and R. Pereira. Internet-Draft: The ISAKMP Configuration Method, 2000.

[33] Baiju Patel, Bernard Aboba, Scott Kelly, and Vipul Gupta. Internet-Draft: DHCPv4 Configuration of IPsec Tunnel Mode, 2001.

[34] B. Ramsdell. RFC 2633: S/MIME Version 3 Message Specification, 1999.

[35] D. Atkins, W. Stallings, and P. Zimmermann. RFC 1991: PGP Message Exchange Formats, 1996.

[36] Pekka Nikander. Minutes of the Secure Shell BOF at 37th IETF. http://www.sprog.auc.dk/ magnus/MHonArc/ssh/msg00641.html, 1996.

[37] AES encryption algorithm. http://www.nist.gov/aes/.

[38] Blowfish and Twofish encryption algorithms. http://www.counterpane.com.

Appendix A

Secret-key algorithms

List of popular secret-key algorithms.

Algorithm	**Key length**	**Description**
DES	56 bits	DES, Data Encryption Standard, is the the most famous block cipher that operates on 64-bit blocks. DES was designed in 1970s by IBM and standardized by US government. DES is considered insecure by today's standards, and has been cracked with efficient computers. [6]
3DES	168 bits	3DES is essentially DES applied three times, normally in an encrypt-decrypt-encrypt sequence with three different, unrelated keys. (168-bit key includes 3 56-bit keys.) 3DES is a secure but slow algorithm. [6]
RC4	1 - 2048 bits	Very fast stream cipher designed by Ron Rivest in 1987. RC4 is used in many commercial software products such as WWW-browsers. Also known as ARCfour. [6]
AES	128, 192 and 256 bits	AES, Advanced Encryption Standard, was selected by US government on February 2001 as the next encryption standard to replace DES. AES is small and fast block cipher that can be implemented efficiently on many processors and in hardware. [37]
Blowfish	32 - 448 bits	Fast 64-bit block cipher invented by Bruce Schneier in 1993. Designed for 32-bit processors and significantly faster than DES. Considered to be very strong. [38]
Twofish	128, 192 and 256 bits	Based on Blowfish, 128-bit block cipher by Bruce Schneier, developed in 1998. Fast in hardware and smart cards. One of the candidates for AES. [38]

Appendix B

Evaluated products

List of products that were included in the evaluation.

Product	**Type**	**Description**
STelnet	TLS	free Telnet software http://www.quick.com.au/ftp/pub/sjg/
start_tls_telnet	TLS	free Telnet software ftp://ftp.runestig.com/pub/starttls/
Tera Term	TLS	free Telnet client cite http://www.infoscience.co.jp/eng/products/sslterm/index.html
Outlook Express	TLS	free email client from Microsoft http://www.microsoft.com/Windows/oe/
Eudora	TLS	email client from Qualcomm http://www.eudora.com/
Sendmail	TLS	free SMTP email server http://www.sendmail.org
UW IMAP	TLS	free POP3 server from University of Washington http://www.washington.edu/imap/
Internet Explorer	TLS	free WWW-browser from Microsoft http://www.microsoft.com/windows/ie/default.htm
Navigator	TLS	free WWW-browser from Netscape http://www.netscape.com
CuteFTP Pro	TLS	file transfer client from GlobalSCAPE http://www.globalscape.com/products/cuteftppro/
SSH version 3.0	SSH	SSH software from SSH communication Security http://www.ssh.com/products/ssh/
OpenSSH 2.9	SSH	free SSH software http://www.openssh.org
SOCKS	SOCKS	generic proxy software from NEC http://www.socks.nec.com/
Stunnel	TLS	free universal SSL wrapper http://www.stunnel.org
SSH Sentinel	IPSec	IPSec client from SSH communication Security http://www.ssh.com/products/sentinel/
F-Secure VPN+	IPSec	IPSec client from F-Secure http://www.f-secure.com/products/vpnplus/

Appendix C

Evaluation results

Evaluation results in respect to the evaluation criteria. Score scale is from 0 to +++.

	SSH	**TLS**	**IPSec**
Security criteria			
Confidentiality + integrity	+++	+++	+++
Authentication	+++ (most versatile)	++ (only X.509)	+ (no user auth)
Denial of service	0	0	++ (prevents tradional attacks)
Usability criteria			
Ease of use	+ (requires tunnel setup)	+++ (integrated into app)	++ (problem of providing feedback)
Transparency	+ (external app)	++	+++
Setup	++	+++	+ (difficult configuration)
Interoperability criteria			
Interoperability	+++ (good standardization)	++	+ (unmature tech, vendor extensions)
Portability	++	+++ (plain protocol, simple to port)	+ (laborous)
X.509	+++ (only in 3.0)	+++	+++
Firewall traversal	(not affected)	(not affected)	+ (NAT problems)
Versatility criteria			
Versatility	+ (primarily terminal+file transfer)	++ (protocol independent)	+++ (secures all traffic, VPNs)
Scalability	++ (certs only in 3.0)	++	+++ (secures all apps in one-shot)
Performance	++	++	+++ (operates in kernel mode)

Appendix D

Protocol connection setups

D.1 TLS connection setup

A short overview of connection setup with TLS is presented. C stands for client and S for server. Authentication is mutual.

```
-C initiates TCP-connection to S and sends its version
number and preferred list of algorithms for key exchange,
encryption, MAC and compression.
-S selects algorithms from the list and sends the selection
to C. If S doesn’t support version or algorithms, S sends
an alert record to disconnect.
-S sends its certificate to C.
-S sends key exchange data to C (not always, depends on key
exchange).
-S sends a request for C certificate.
-S sends "server done" to indicate handshake is complete.
S waits response.
-C verifies certificate validity and alerts if invalid.
-C sends its certificate to S.
-C requests S to verify C certificate.  S verifies
certificate and alerts if invalid.
-C sends key exchange data to S.
```
-Keys have been exchanged.
```
-C informs S that future packets are encrypted.
```
-Communication is secure.
```
-C sends finished message.
-S informs C that future packets are encrypted.
-S sends finished message.
```
-Connection is complete. Application protocol starts.

D.2 SSH connection setup

A short overview of connection setup with SSH is presented. C stands for client and S for server.

-C initiates TCP-connection to S, which is listening on TCP-port 22. Both send their version numbers to each other.

-C verifies that it supports the same protocol version. If not, C disconnects.

-Transport layer protocol:

-Key exchange starts. Both ends send a preferred list of algorithms for key exchange, S host key, encryption, MAC and compression. The last three are negotiated separately for both directions.

-Both ends iterate C’s list of algorithms and select the first algorithm that is also in S’s list. If common algorithms are not found, both disconnect.

-Keys are exchanged according to the selected key exchange algorithm (currently only Diffie-Hellman).

-During key exchange, S also sends its public key to C.

-C verifies the validity of the public key. If invalid, C disconnects.

-Keys are taken into use. Communication is secure. C requests authentication service "ssh-userauth".

-User authentication protocol:

-S sends a list of authentication methods than can be used for login.

-C authenticates itself with one authentication method.

-S lets C in unless more authentications are required in which case the authentication dialogue continues.

-User has been authenticated. C requests connection service "ssh-connection".

-Connection protocol:

-C requests any number of session channels for remote execution of a program, which may be a shell, an application, or a SSH subsystem. Tunnels get requested as well.

-Channels are opened and programs started on server. Connection is complete.

D.3 IPSec connection setup

A short overview of connection setup with IPSec is presented. I stands for initiator and R for responder. IKE uses main mode for key exchange.

```
-No IPSec connection exists.  IP-layer receives data from
the upper layer (from an application running on the local
computer).  IPSec must negotiate a secure session before
IP-datagrams can be sent.
```
-IKE phase 1, main mode:
```
-I initiates connection to R. I sends a UDP-packet
containing a proposed IKE policy to UDP-port 500.
-R replies with a proposed policy.
-I sends random data and key exchange data to R.
-R replies with random data and key exchange data to I.
```
-IKE has established a secure channel.
```
-I sends its certificate and a signature to R.
-R verifies certificate validity.
-R replies with its certificate and a signature to I.
-I verifies certificate validity.
```
-IKE phase 2, quick mode:
```
-I sends key exchange data and a SA containing prosed
protocols and algorithms to R.
-R replies with key exchange data a proposed SA for the
other direction.
-Two IPSec SAs negotiated, one for each direction
```
-IKE completes. IPSec connection is up.
```
-incoming IP-datagrams are authenticated and decrypted
according to the inbound SA
-outgoing IP-datagrams are encrypted and attached with a
MAC according to the outbound SA
```