Department of Computer Science

Robert Magnusson
Tomas Nilsson

A Proposal of an In-Session Dynamic Encryption Service

D Dissertation 15 ECTS

2005:09

A Proposal of an In-Session Dynamic Encryption Service

Robert Magnusson
Tomas Nilsson

This thesis is submitted in partial fulfillment of the requirements for the Masters degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Robert Magnusson

Tomas Nilsson

Approved, October 14 2005

Opponent: Katarina Asplund

Advisor: Thijs Holleboom

Examiner: Donald F. Ross

Abstract

A large part of today's data exchange is done over the Internet and wireless networks are growing in importance. This makes it easy to make use of small handheld units, such as laptops and PDAs, to exchange data over the Internet. The sender may want to keep the transmitted data secret for a short or a long period, and the security requirements may vary for different data sets. This implies that it would be an advantage if the hosts could be able to decide how long time the data could be kept secret and point out which part of the data to encrypt, to save computer resources. There exist many ways to achieve these goals today. Weak and strong encryption algorithms give the ability to hold the data secure for a short or a long period, and situations where it is possible to point out certain parts of the data exists. In these solutions one has to decide which algorithm and which data parts to encrypt before establishing a connection between the hosts. In this thesis, we introduce the concept of dynamic encryption which offers variable levels of encryption during the session, and the ability to adapt this encryption level to performance and CPU resource availability constraints. A model to accomplish this concept is suggested and tested, and the results show that the dynamic concept could be implemented with a low overhead cost.

Acknowledgements

We would like to thank Randall Stewart for giving us fast SCTP support and for sending us patches to get things working.
We would also like to thank Stefan Lindskog for his enthusiasm and for the idea of the Dynamic Encryption service.

Contents

1 Introduction **1**

2 Background and Motivation **5**

2.1 Encryption Algorithms 5

2.2 Security 7

2.3 Security Services 8

2.4 Encryption Level 9

2.5 Previous Work 10

2.6 Areas of Interest 12

2.6.1 Protection of Layered Information 12

2.6.2 Protection of Partitioned Information 13

2.6.3 Protection of Live Broadcasts 13

2.6.4 Protection of Video Conferences 14

2.6.5 Protection in Different Environments 15

2.7 Possible Implementation Models 16

2.7.1 Periodic Encryption 16

2.7.2 Stepwise Rising 17

2.7.3 Stepwise Falling 17

3 The Design of the Dynamic Encryption Model **19**

3.1 Definitions 20
3.2 Selection Mechanism 20
3.2.1 Operations 21
3.3 Investigation of the Dynamic Encryption Service 21
3.3.1 Possible Solutions 22
3.3.2 Evaluation of Solutions 25
3.4 Transport Layer Protocol 26
3.4.1 The Stream Control Transport Protocol 26
3.4.2 Useful Features for the Dynamic Encryption Service 27
3.5 The Encryption Mask 27
3.5.1 Encrypting the Encryption Mask 28
3.6 Control Channel 29
3.7 The Data Channel 31
3.7.1 The Encryption Mask and the Encryption Mask Length 31
3.7.2 Separating the Data and the Encryption Mask 32
3.7.3 Padding 33
3.8 Summary of the Dynamic Encryption Model 33

4 Test Implementation **35**
4.1 Target of Test Implementation 35
4.2 Test Implementation Overview 36
4.2.1 The Program Modules 36
4.3 Theoretical Gain 42
4.4 Test Cases 45
4.4.1 Test Environment 46
4.4.2 Functionallity Test, Test Case 1 46
4.4.3 Dynamic Encryption Behaviour Without Transmission, Test Case 2 47
4.4.4 Dynamic Encryption Behaviour With Transmission, Test Case 3 . . 48

4.4.5 Network Speed Effect, Test Case 4 48
4.4.6 Encryption Mask Alteration Overhead, Test Case 5 49
4.4.7 Key Initiation, Test Case 6 50
4.5 Test Case Results 50
4.5.1 Functionallity Test, Test Case 1 51
4.5.2 Dynamic Encryption Behaviour Without Transmission, Test Case 2 52
4.5.3 Dynamic Encryption Behaviour With Transmission, Test Case 3 . . 54
4.5.4 Network Speed Effect, Test Case 4 56
4.5.5 Encryption Mask Alteration Overhead, Test Case 5 57
4.5.6 Key Initiation, Test Case 6 58
4.6 Discussion of test results 59

5 Conclusions **65**

6 Future work **67**

References **71**

A Test Results **73**
A.1 Test case 2 results 73

List of Figures

2.1 Encryption Level 9

2.2 Handheld unit changing network area 15

2.3 Periodic encryption broadcasting movie and commercials 16

2.4 Stepwise rising encryption 17

2.5 Stepwise falling encryption 18

3.1 Transmission in one channel 23

3.2 Transmission in two channels 23

3.3 Transmission in three channels 24

3.4 Transmission in two channels, separating mask from control data 26

3.5 Negotiation of the encryption level 30

3.6 Terminology of the data in the data channel. 31

3.7 Transmission of encryption mask length, encryption mask and data. 32

3.8 Padding of a data block 33

3.9 A picture over the DE model 34

4.1 An overview of the dynamic encryption model 37

4.2 Schematic picture of the test environment 46

4.3 Test case 1, where the encryption level is decreasing. 52

4.4 Encryption time as a function of encryption level for different amounts of data, test case 2a and 2b. Also shown is the time for encryption without using the dynamic encryption service. 53
4.5 Encryption and transmission time as a function of encryption level for different amounts of data, test case 3a and 3b. Also shown is the time for encryption and transmission without using the dynamic encryption service. 55
4.6 Figure a) shows the encryption and transmission time as a function of mask change frequency at encryption level 50%. Figure b) shows the transmission of the same file with only one change of the encryption mask. By reading the time needed at encryption level 50% in figure b), it can be compared to the variation of time in figure a), test case 5. 58
4.7 Test case, where the transmission rate is set to 250 Mbps. 63

6.1 Modules above the dynamic encryption service. 67
6.2 Several clients connected to a server. 69

List of Tables

4.1 Assembly instructions 43
4.2 Used hardware and software 47
4.3 Time needed to read data from disc 51
4.4 Time needed for transmitting data with different transmission rates, test case 4a. (Pure AES) 56
4.5 Time needed for transmitting data with different transmission rates at different encryption levels, test case 4b. (dynamic encryption service) 56
4.6 Time needed for transmitting data at different frequency of mask changes, test case 5. 57
4.7 Results from test case 6. Time needed for encryption and decryption key initialization 58
4.8 Overhead per 1024 bytes 62
4.9 Total data to be sent 62

Chapter 1

Introduction

The Internet is still growing tremendously, adding new users and services every day. The effect of this is that both the amount, and the size of information exchanged between hosts also are growing rapidly. The information exchanged between the hosts may, for various reasons, need to be kept secret during a certain period of time. This confidentiality is accomplished by encrypting the information. Depending on the nature of the information the confidentiality demands may vary. For example, a financial report only needs to be kept confidential until it is publicly published, while a medical report usually does not have a "confidentiality expiring date". In [17] the "Principle of Adequate Protection" is stated as: *Computer items must be protected only until they lose their value. They must be protected to a degree consistent with their value.* This implies that the time the encryption need to withstand an attack may vary depending on the data it is protecting. Traditionally all data that is to be sent has been fully encrypted or not encrypted at all. In this case the ways of decreasing the load on the hosts when encrypting, are using a shorter encryption key or using a faster, and possibly less secure, encryption algorithm. In [16] this type of encryption algorithm is reffered to as a lightweight algorithm. Instead of using a lightweight algorithm and encrypting all the data in the transmission to reduce resource consumption, it could be beneficial if a stronger algorithm is applied to a subset

consisting of the sensitive parts of the data. In this way, a more secure encryption of the sensitive data could be achieved and the load on the hosts could be held down.

In some situations, the ability for a host to take care of encryption or decryption of the data may vary over time depending on other processes going on. If it is possible to change the amount of data to be encrypted during the present transmission an adaption to the hosts condition can be achieved. By giving the hosts the ability to communicate with each other during a transmission they could negotiate a suitable level of the amount of the data to be encrypted. Thereby the hosts, on the basis of their capability, could take part in the decision of the amount of data to be encrypted and thus be able to take care of the encryption or decryption without suffering from latency or a too heavy load. Some situations where this ability could be useful are when one or more of the hosts is a handheld unit with limited power resources and a small CPU, or a server with low CPU resources with the intention to send a big data file that has to be encrypted.

Also, more and more portable units, such as laptops and PDAs, are used giving the user the ability to move freely within a wireless network area. These wireless network areas could be classified in different categories, for example trusted, non-trusted and insecure, giving the possibility for individual treatment of the data to be transmitted in the aspect of encryption. By giving the ability to negotiate and adjust the encryption level during the present network session there is no need for establishing a new connection between hosts when changing network area.

This thesis describes an approach that makes it possible to change the subset of data to be encrypted during the present transmission and gives the participating hosts the ability to negotiate the appropriate subset to encrypt.

To test these ideas a series of tests has been performed. The intention of these tests are to show the effects of a variable encryption level. Two main aspects are tested: The effect in performance from a decreased encryption level, and the overhead cost introduced by the dynamic encryption mechanism. The result of the tests show that a decreased encryption

level will result in a corresponding decrease of execution time. However, the mechanism which performs the actual encryption level variation could possibly be optimized.

Chapter 2 contains a general introduction to cryptography and a motivation for our in-session Dynamic Encryption service. In chapter 3 different models are discussed and a motivation for the model we chose to implement and evaluate is presented. Chapter 4 presents the implementation model and shows the architecture behind the dynamic encryption model. The conclusions of this work are presented in chapter 5 and finally in chapter 6 issues for future work in building some models applicable above the model described in this thesis are discussed.

Chapter 2

Background and Motivation

In chapter 2.1 the background to encryption algorithms and their use is described. A definition of the encryption level with an aspect of security is given in sections 2.2, 2.3, and 2.4. Some previous work on scalable encryption is presented in section 2.5, where one solution relies upon encryption with faster, and possibly less secure, algorithms for the insensitive parts of the data and slower and more secure algorithms are used for the sensitive parts of the data. Another solution proposes a selection of the data to be encrypted, before the transmission and thereby lower the encryption and decryption cost. Finally several areas where dynamic encryption could be useful are presented in 2.6.

2.1 Encryption Algorithms

Encryption algorithms have been used by humans for a very long time. We can find encryption back in Caesar's time where the alphabet's letters are shifted one or more steps (i.e, A becomes B, B becomes C and so on...). A tremendous evolution in encryption took place during World War two, where the famous Enigma machine [22] played a central role. The Enigma machine was one of the reasons why the people constructing encryption algorithms and encryption machines became more and more skilled in developing better encryption

algorithms. On the other hand, the code breakers also became more skilled in breaking the encryption algorithms. Althoug the enigma machine reached an almost mythical status, the evolution of encryption algorithms has continued and encryption are becoming more and more used in today's life. In communicating, people sometimes may want to keep some part of their information secret and therefore use encryption. Today a very large part of information exchange is done over the Internet and encryption is commonly used when people want to keep some of the exchanged information secure. However the size of the information which is transmitted over the network grows rapidly since bigger and bigger data files, such as pictures and movies, are exchanged, and the supply and the demand for these files increase. Encryption and decryption of this kind of data is time consuming and requires fast computers and fast encryption algorithms. The evolution of encryption algorithms has lead to faster algorithms and nearly unbreakable solutions, but there are some factors to take into consideration when dealing with encryption over the Internet:

- The hosts might be handheld units with less computing power than a stationary unit. These units might not be able to manage the large amount of data that has to be first received then decrypted and finally delivered.

- The sender of the transmitted data could be loaded with power consuming work that affects the encryption speed in a way such that it may not be able to carry out the encryption work before the desired time limit.

Encryption algorithms can be classified into *strong* and *weak* encryption algorithms [16]. Here follows a short description of these two terms:

Strong encryption algorithms are algorithms that withstand a brute force attack for a long time and have a high level of security [16]. However, since computers get faster and code breakers get more skilled, an algorithm that is considered strong today might be considered weak tomorrow. The security of encryption algorithms depends, among other things, on the length of the encryption key. The longer the key used, the more secure the

encryption algorithm becomes. For example, there are $2^8 = 256$ possible keys in a 8 bit long key and $2^{128} = 3.402823 \cdot 10^{38}$ possible keys in a 128 bit long key. Therefore, it will take a code breaker, using a brute force attack, much longer time to find the correct key in the latter case [21]. The value of the encryption key also affects the strength of the encryption. People tend to use keys that are easy to remember, like common words in the english language. The use of such a key may reduce the time needed to break the code as the number of possible keys is reduced enormously if the code breaker confine the search to words from a dictionary [21]. The time needed for a specific computer to encrypt the data with a stronger algorithm is greater compared to encryption with a weaker algorithm. This will cause a delay in delivering the data.

Lightweight encryption algorithms are designed to constrain the load on computational, storage, and transmission resources, possibly at the cost of a less secure encryption.

Weak encryption algorithms are lightweight algorithms where it is fairly easy to break the encryption code [16]. However, if the time the data is to be kept secret is short, the use of a weak encryption algorithm may be justified. These weak algorithms are less time and resource consuming and reduce the delay time in transmission from the sender to the receiver.

2.2 Security

Traditionally, the following three aspects of security are distinguished: confidentiality, integrity, and availability [15]. A secure system has to fulfill all three aspects, this also applies to systems involving the Internet. Confidentiality concerns the ability to prevent an unauthorized user get hold of secure information. Integrity concerns preventing unauthorized modification of data, and availability concerns unauthorized withholding of data. These

three aspects describe different ways of protecting data and can sometimes contradict each other. For example, when availability is the most important aspect it may have to be withheld at the cost of reduced integrity and confidentiality. In this thesis the confidentiality is the most important aspect, and therefore the security issues further described are done with the intention of maintaining a high confidentiality.

2.3 Security Services

In [14] security is thought of as a tuneable system attribute that allows users to request a specific protection level as a service from the system. Today, the lack of mechanisms by which system owners and users can request a specific level of protection as a service in the system makes it impossible to offer protection based on need. Instead, all users are offered the same services. This results in applications with inadequate data protection and unnecessary costs to users. In [14], a proposal of a taxonomy for dynamic data protection services is presented. A unified terminology is introduced to provide a framework in which the dynamic data protection can be classified and to provide a basis for development. Five different dynamic data protection services are discussed in [14], namely IP security, transport layer security, Authenticast[1], a scalable encryption service[2], and a dynamic encryption service[3]. These dynamic data protection services are classified according to the taxonomy and they each becomes an own class. By introducing this taxonomy, an overview of the security in the network can be achieved and security could be looked upon as a QoS parameter.

[1] A scalable authentication service.

[2] A scalable encryption service allows information to be partially encrypted using a predetermined pattern.

[3] A dynamic encryption service allows the partial encryption pattern to be changed during a session.

2.4 Encryption Level

The level of encryption is a relative term and this section will explain what is meant by this in the context of this work. If data is to be transmitted from a sender to a receiver and all the data is encrypted we say that 100% has been encrypted, and therefore the encryption level is 100%. In the same way, if every second block is encrypted in the transmission, the encryption level is 50%, assuming that all blocks are of the same length. The encryption level can thus be expressed as the number of blocks that is encrypted out of the total number of blocks that are to be sent (see section 3.1). Figure 2.1 shows the case where 4 blocks out of 10 are encrypted, resulting in an encryption level of 40%.

This gives that the more blocks that are encrypted, the higher the encryption level,

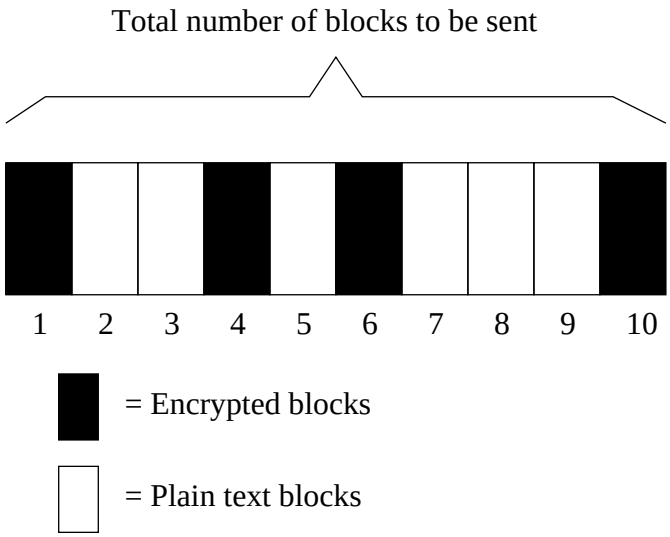


Figure 2.1: Encryption Level

increasing the confidentiality of the transmitted data.

The encryption level, however, not the only factor that affects the security in the transmission. One also has to take into consideration what kind of encryption algorithm that is used. If the use of an algorithm with a key length of 256 bits is compared with the same algorithm with a key length of 128 bits, the use of a longer key produces a higher security

level (see section 2.1). Different encryption algorithms have different security levels which will affect the total security level.
Other aspects that affect the security level are authentication integrity and key exchange algorithms. However, there already exist good and tested algorithms in these areas, such as Distributed Authentication Security Service (DASS) and Diffie-Hellman key exchange algorithm [21], these will not be further discussed here.

2.5 Previous Work

Goodman and Chandrakasan [9] introduced the concept of scalable encryption. Scalable encryption means that high priority data is encrypted with a strong encryption algorithm and low priority data is encrypted with a weaker encryption algorithm. This implies that the data format and structure are known to the user, to make it possible to point out which data to be encrypted with a stronger algorithm and for which data to use a weaker algorithm. The possibility to point out which parts of the data to be encrypted with a strong algorithm makes it possible to allocate security where it is needed the most within a data stream. In addition, the parts of the data encrypted with a weaker algorithm will decrease the computational burden.

An evaluation of selective encryption techniques of MPEG streams, in the work of Alattar and Al-Regib [1], shows the high efficiency of encrypting only data associated with Intra-coded macro blocks, or the headers of all predicted macro blocks together with the data in all Intra-coded macro blocks. Intra-coded macro blocks are sensitive portions of a MPEG compressed video stream and represent a complete picture while other frames only represents changes and are thereby less sensitive with the aspect of hiding information. It is shown in [1], that the latter method is most effective regarding best security level but

also achieves less savings in processing time.

In the paper by Lindskog, Strandbergh, Hackman, and Jonsson [16], the encryption level is selected before the transmission and is maintained throughout the session. This makes the encryption level dynamic on a per-session basis and it offers a scalable solution at a low cost. The model presents a way to make a trade-off between the security level and the encryption cost that can be applied to any type of data.

IP Security (IPSec) is a protocol that implements security at the IP level as described in [12]. IPSec is provided as an integrated part of IP version 6 (IPv6) [7] and can be added as an extension to IP version 4 (IPv4) [18]. Security services in IPSec are implemented through extension headers that follow the main IP header attached to each packet. Two types of such headers exist: Encapsulating Security Payload (ESP) headers and Authentication Headers (AHs). An ESP header is attached when data confidentiality is requested, while an AH is attached when data authenticity and/or data integrity is requested. When an ESP header is used, an optional AH can be attached as well.

Transport Layer Security (TLS) [3] [8] is an Internet protocol that implements security features above the Transmission Control Protocol (TCP) or the Stream Control Transmission Protocol (SCTP) [20], which implies that a reliable end-to-end security service is provided. The first version of TLS was derived from the Secure Socket Layer (SSL) protocol, introduced originally by the Netscape Corporation. Version 3.1 of SSL and TLS 1.0 are essentially the same protocol. Two types of services are offered by TLS: confidentiality and message integrity. Confidentiality is achieved through the use of conventional encryption algorithms such as DES, 3DES, IDEA, etc., and message integrity is guaranteed through the use of a hash algorithm (SHA-1 or MD5). In addition, a compression algorithm can be used to compress data before adding the Message Authentication Code (MAC) produced

by the hash algorithm, which in turn is added before encryption of data is performed. In TLS, both the application data and the MAC are encrypted.

2.6 Areas of Interest

In this section a number of cases where the user could benefit from a more detailed level of control over the encryption process are presented.

2.6.1 Protection of Layered Information

In some cases the information consists of several parts, called layers, that are combined to reconstruct the original information. Some of these layers may contain information that is publicly known, while other layers may contain sensitive data.
This model we will call the Layered Information Model.

Protection of Maps

One example would be a military map constructed from one layer containing information about the location of strategic targets and troop movements, and one layer containing the actual geographical map. This idea is derived from the paper [10], by Iyengar and Samtani, where the map is partitioned into smaller pieces and treated differentially depending on their priority. In this case, the geographical map can be considered public information but the strictly military information is to be kept secret. An example of non-military use is a map containing information about the whereabouts of protected wildlife species such as eagles' nests or current locations of wolf packs. Again the geographical map will be publicly known but the information about the wildlife is to be kept secret. The point of dynamic encryption in this area is that since some of the layers contain information that

is publicly known, it is sufficient to encrypt the layers containing non-public information.

In the examples given above there are significant differences in the density of data. The layers containing the geographical information are likely to contain much more data than the layers containing the secret information, and thus the number of resource-consuming encryption operations will be reduced if only the layers containing secret information is encrypted, compared to the alternative where all data (i.e, all layers) is encrypted. This will make it possible to decrease the workload on the involved hosts, which might be handheld devices with limited capacity, while still being able to transmit all data throughout one session.

2.6.2 Protection of Partitioned Information

When the data to be transmitted consists of several files and one specific file is needed to be able to make use of the data set, there could be beneficial to encrypting only this important file. The benefit could be reduced overhead time to encrypt and decrypt, and also in lower power consumption. An example of this kind of transmissions is when a host wants to download an application which could consist of an executable file and several configuration files. The user of the application needs access to the executable file to be able to run the application, otherwise the application will be useless. Therefore, only encrypting the executable file could be enough to keep the application secure from unauthorized downloaders and, at the same time, gain computational resources.
This model we will call the Partitioned Information Model.

2.6.3 Protection of Live Broadcasts

In this model focus is on the performance of the involved hosts, the level of encryption is adapted to the currently available resources. Consider a server broadcasting a live concert and handling a large amount of clients (viewers). The viewers paying to view the broadcast

demand high quality broadcast and the nature of live broadcasting demands just-in-time encryption to be performed by the broadcasting server. The encryption process puts a high load on the server and may cause the quality of the broadcast to deteriorate. A possible solution would be to encrypt only a fraction of the data such that the load on the server reaches an acceptable level. If the server load decreases, a larger amount of data can be encrypted and thus the server may adapt to get the optimal level of quality and encryption. In addition, the client could also be able to request a temporary reduction of the encryption level in the same way as the server. The fluctuation of the level of encryption will be limited to a pre-determined range. The lower limit will in this case be set by the supplier of data, effectively the server. The upper limit could be negotiated at initialization, or simply set to 100 percent, and the level of encryption would be negotiated during initialization and re-negotiated during the session. However, both the server and the client should strive to keep the level of encryption as high as possible at all times. As an addition to this model it might be possible to introduce a minimal average rate of encryption per session. The average rate will allow temporary drops in encryption level, but hinder the possibility of the encryption level to be kept at the lowest allowed rate throughout the entire session. This model we will call the Live Broadcast Model.

2.6.4 Protection of Video Conferences

The idea behind this model is to be able to adjust the level of encryption manually and absolute. The typical target of this model would be a video conference where some part of the conference has to be kept confidential. The part of the conference that deals with publicly known, or non-confidential, information may be shared between participants in plain text. This will, as the Live Broadcast Model described in section 2.6.3, save resources and focus on the quality of the media. Since a conference occurs in real time, it may not be possible to determine in advance when the confidential information is to be treated and thus it must be possible for the participants to change the encryption level manually in

real time. This model can be viewed as an encryption on/off switch, which implies that any raise of encryption level, corresponding to encryption switch on, must be carried out without negotiation. However switching off the encryption may be a matter of negotiation. This model we will call the Video Conference Model.

2.6.5 Protection in Different Environments

Business networks tend to become more and more wireless and situations where a handheld unit, such as a PDA or a laptop, is transported between networks occur more frequently nowadays. This may imply that a change, in real time, of encryption level and IP address for the new network area is desirable. When a handheld unit (see figure 2.2) is in a network area with a certain kind of security level (security level 1) and the user wants to move to another network area, possibly with another security level (security level 2), a transparent change of IP address and security level needs to be done.

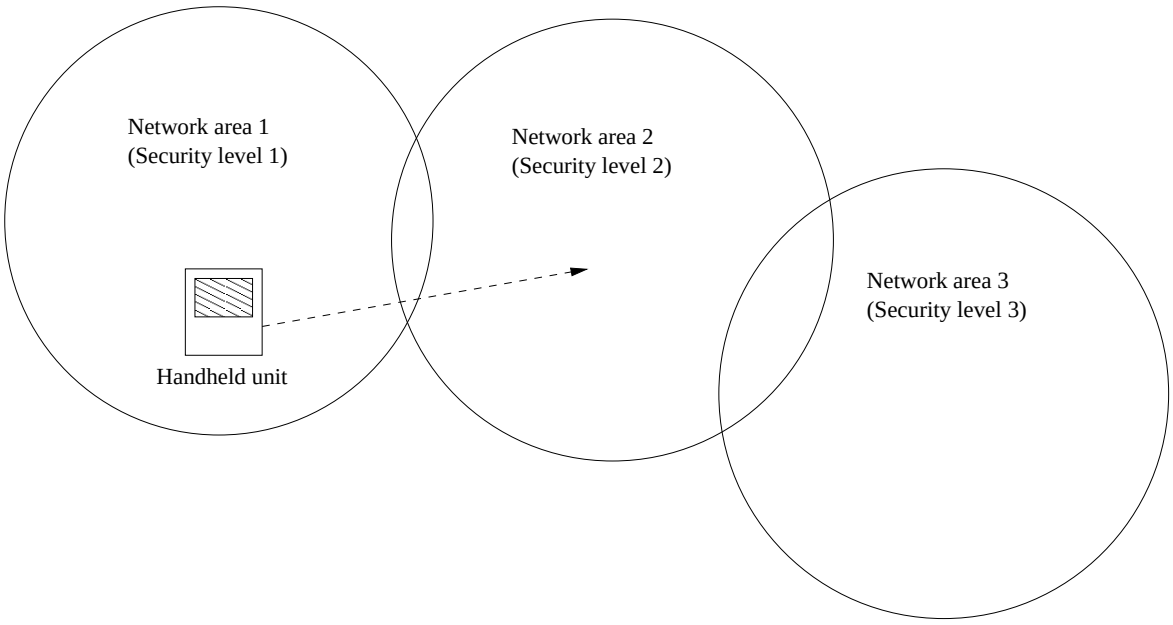


Figure 2.2: Handheld unit changing network area

A similar transparent change of IP addresses has been discussed in [5] but here the

focus is on the change of encryption level. Dynamic encryption would allow for adapting the security level to the requirements of the new network area without resetting the session. This model we will call the Different Environments Model.

2.7 Possible Implementation Models

The following models describe different ways to make use of the encryption level when transmitting data over a network.

2.7.1 Periodic Encryption

The Video Conference Model 2.6.4 may be extended to a periodic encryption-model where the alterations of the encryption level are triggered by time or type of data. This model may be used to broadcast plain text commercials and previews of movies, again with the intention of saving resources, while the actual movie broadcast is encrypted (see figure 2.3). During the encryption period it could be desirable to use negotiation as presented in the live broadcast model.

This model we will call the Periodic Encryption Model.

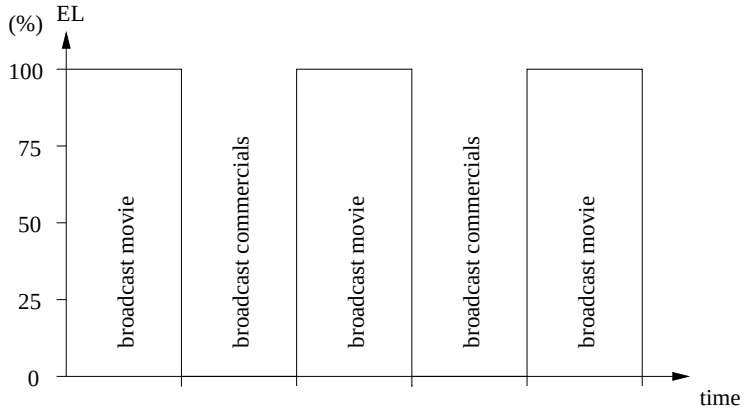


Figure 2.3: Periodic encryption broadcasting movie and commercials

2.7.2 Stepwise Rising

The model displays another aspect of the live broadcast example 2.6.3. If we consider an ice hockey broadcast, the most interesting part from the viewers perspective is the end of the game, there is no use in watching the game if you don't get to know the result. This implies that it might be a waste of resources to maintain a high level of encryption throughout the whole match. A more resource-effective way to handle the broadcast would be to increase the level of encryption stepwise from little or no encryption at all from the start, to a high level of encryption towards the end of the match (see figure 2.4). This stepwise raise of encryption could also be done over each separate period of the match.

This model we will call Stepwise Rising Model.

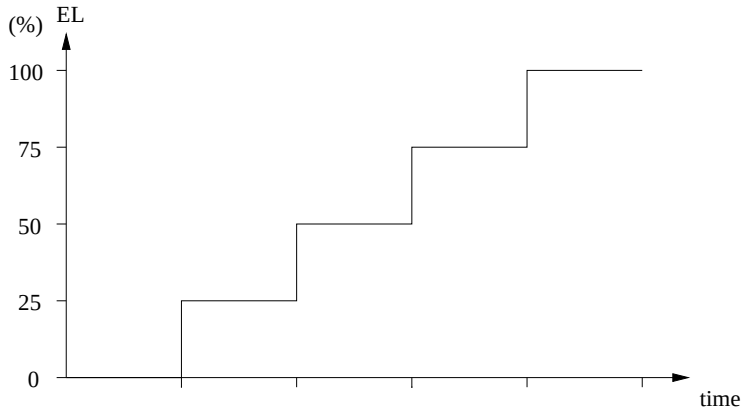


Figure 2.4: Stepwise rising encryption

2.7.3 Stepwise Falling

When information to be kept secret lies in the beginning of the data to be transmitted there could be a gain in lowering the encryption level during the transmission (see figure 2.5). Possible situations where this could take place could be TV-games, where the conditions for the game are given in the beginning of the broadcast. To be able to participate in the game one has to know the rules given in the beginning. Everybody is able to view the

program, but only authorized persons or entities are able to decrypt the broadcast and thereby get the rules.

This model we will call the Stepwise Falling Model.

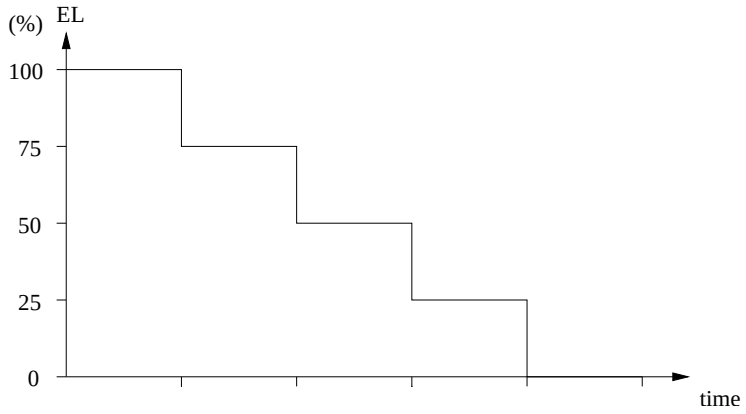


Figure 2.5: Stepwise falling encryption

Chapter 3

The Design of the Dynamic Encryption Model

The intention with the dynamic encryption model is to offer the ability to change the encryption level during an ongoing transmission, and thereby offer a service that could be used to develop the "Areas of Interest" presented in section 2.6. In section 3.1 the definitions needed to describe the model are given. In section 3.2 a mechanism to point out specific parts of the data to be encrypted is described. Three possible solutions: one channel, two channels and three channels are investigated and evaluated in section 3.3, and in section 3.4 we give a short introduction to the SCTP protocol and point out features in the protocol that could be useful in the dynamic encryption model. A central part in the dynamic encryption model is the encryption mask, which is described in section 3.5 and the two channels used, on the basis of the evaluation in 3.3, are described in section 3.6 and section 3.7. In section 3.8 a summary of the choosen model is given.

3.1 Definitions

To be able to give a clear and concise description of our dynamic encryption service we introduce the following definitions:

Definition 1. An in-session dynamic encryption service is a service that has been explicitly designed to offer various encryption levels that can be selected, and varied, during the session.

Encryption levels are in this thesis defined as follows:

Definition 2. Let X denote the total amount of data, and let Y denote the encrypted data subset. The encyption level will then express the relation Y/X.

An encryption service is considered dynamic provided:

Let EL denote the encryption level.

Let EL_{t_m} and EL_{t_n} denote two arbitrary points in the transmission.

$$EL_{t_m} \neq EL_{t_n}$$

is valid, and

$$EL \epsilon [0\%, 100\%]$$

.

3.2 Selection Mechanism

In [16] a m-out-of-n selection mechanism is introduced to specify the granularity of the encryption level. To take this one step further and make it possible for an application to specify exactly which blocks to encrypt, we introduce the encryption mask. The encryption mask is a sequence of bits specifying which blocks to encrypt and which blocks to transmit as plain text. The length of the sequence is n and the total number of bits set to 1 (1 denotes

the blocks to encrypt) in the sequence is m. Combining the m-out-of-n mechanism with the encryption mask provides the ability for the application to take the specific structure of the data into consideration in order to make the partial encryption as effective as possible, i.e. to encrypt specific parts of the data. In the general case where the user application does not wish to control the encryption process in detail, the encryption mask may be randomly generated or composed of the first m bits set to 1 and the remaining $n-m$ bits set to 0. The latter composition directly corresponds to the m-out-of-n selection mechanism proposed in [16].

3.2.1 Operations

In addition to ordinary user applications for transmitting data, the dynamic encryption service application should be able to:

- Get the encryption-mask.
- Get the encryption-mask length.
- Set the encryption-mask and its length together.
- Retrieve the block size used to provide the possibility to specify encryption-mask in the above set operation
- Set and get encryption algorithms and keys

3.3 Investigation of the Dynamic Encryption Service

In this section, different solution strategies that support dynamic encryption as described above, are investigated. First we describe the different solutions and their advantages and drawbacks, then we evaluate the solutions and justify our choice. The signalling can be done either *in-band*, which means that the control data is delivered before the application

data and in the same channel, or *out-of-band*, which means that the control data is delivered before the application data but in a different channel. In the latter case, a synchronization has to be done between the control channel and the data channel.

3.3.1 Possible Solutions

We have evaluated three possible solutions to obtain an dynamic encryption service. This is done in order to discover all advantages and drawbacks for each of them. The main difference between these solutions is the number of channels used for the transmission.
For the packets that are being sent we want to transmit:

- Application data which consists of encrypted data or plain text data.
- Control data which consists of the mask, negotiation of the key or negotiation of algorithms.

Below three different models will be described; containing a one, two, and a three channel solution respectively.

One Channel

This solution, using in-band signaling, uses a single channel for both data and control (see figure 3.1). The control data and the application data need to be separated by a flag indicating the data type of the packet. Control data, like the encryption mask, is prepended to the application data.

The advantages of this solution are:

- The independence of a specific underlying transport protocol
- The control data and the application data are automatically synchronized.

The drawbacks are:

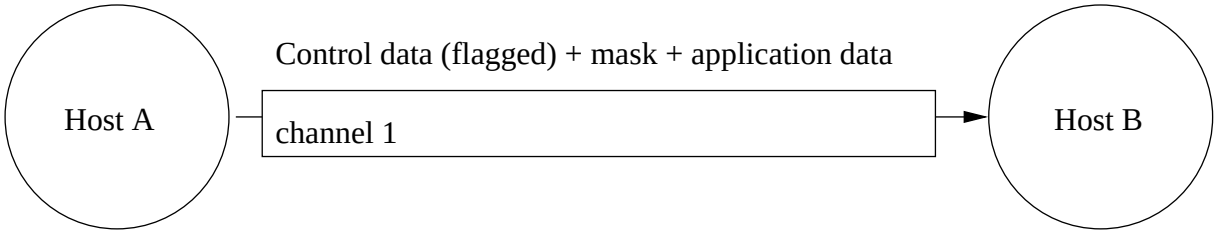


Figure 3.1: Transmission in one channel

- The application data and the control data get unwanted dependence regarding congestion control. If the network is congested, it may be possible to increase the encryption level without affecting the performance. In one channel solution this will not be possible.
- Separation of the control data from the application data must be done at the receiver.

Two Channels

This solution is an out-of-band signaling using one channel for data transport and one channel for control (see figure 3.2). The control data and the application data are separated in two different channels. The information about whether the application data is encrypted or not is transmitted in the control channel as an encryption mask.

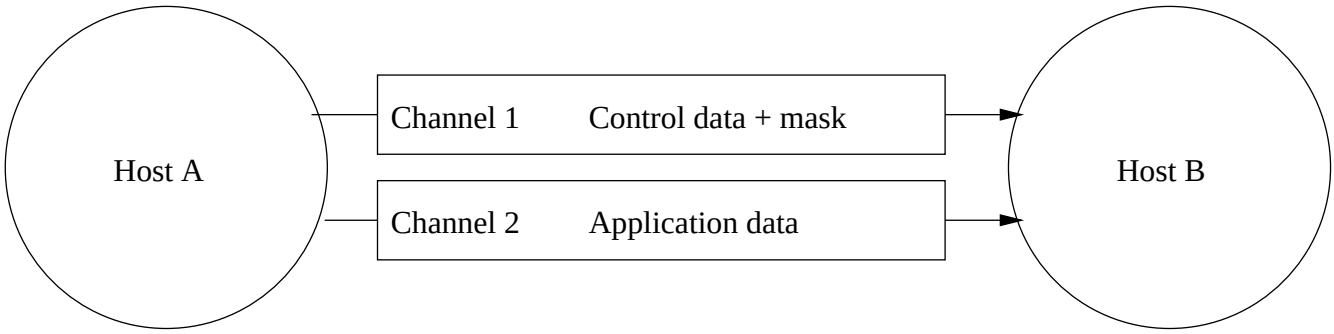


Figure 3.2: Transmission in two channels

The advantages of this solution are:

- Logical separation between the control data and the application data, which is also valid for the congestion control.
- Delay in the retransmission of application data does not necessarily affect the transmission of control data.

The drawbacks are:

- A synchronization between the control data and the application data is needed.
- An implementation will be restricted to either the SCTP protocol or the use of several TCP connections.

Three Channels

This solution is an out-of-band signaling using one channel for control and two channels for data, one fully encrypted, one plain-text only (see figure 3.3). Thus the different data will be separated in three different channels.

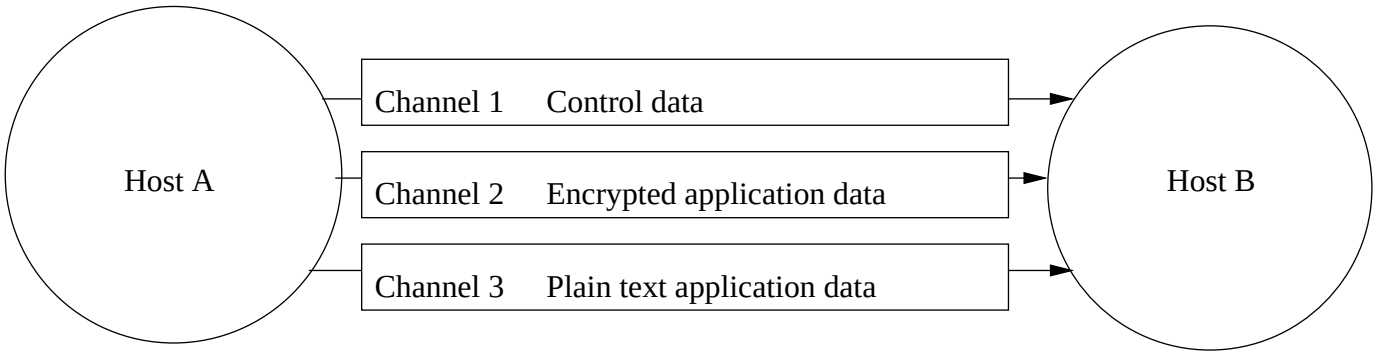


Figure 3.3: Transmission in three channels

The advantages of this solution are:

- It is possible to determine from the channel on which the data is received if the data is encrypted or not, making the encryption mask superflous.

- Logical separation between the control data, the application data encrypted and the application data in plain text. This is also valid for the congestion control.

The drawbacks are:

- Creates dependency between the channels.
- An implementation will be restricted to either the SCTP protocol or the use of several TCP connections.
- The application data may be delivered out of order.
- Claims a buffer or polling.
- An attacker may also be able to determine which parts of the data that are transmitted in plain text.

3.3.2 Evaluation of Solutions

The three channel solution described in section 3.3.1 could be done using the SCTP protocol, either with the ability to specify which stream to read from or by buffering. However, it turns out that there is no possibility to specify which stream to read from. Therefore, buffering is needed above the SCTP protocol if we want to use three channels. This buffering is resource consuming and one of the purposes of this work is to minimize the load on the participating hosts. Due to this, the three channel solution was dropped. With two channels we have the advantage, when separating control data from application data, of getting no dependence between them but even in this solution buffering might be nessecary when changing encryption level. However, buffering can be avoided by using a combination between solution one and solution two. By transmitting control data that is strongly connected to the application data, such as the encryption mask, together with the application data, the problem with encryption level change can be solved. The control data

with weak connection to the application data, such as encryption level negotiation, will be transmitted in the control channel (see figure 3.4). Thus we get the advantage of the ability to transmit control data, such as negotiating encryption level or changing encryption key, even if the application data is delayed. In addition, there is no need for buffering when data per stream is in order and the synchronization between the mask and the application data is achieved. This is the model we apply for our further work in this thesis.

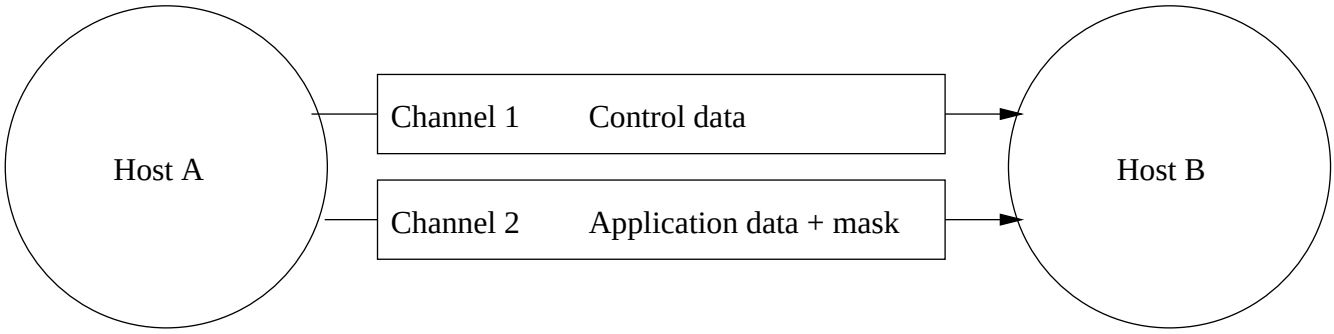


Figure 3.4: Transmission in two channels, separating mask from control data

3.4 Transport Layer Protocol

Although the solution described in section 3.3.2 could be implemented using TCP, the SCTP-protocol will be used in this implementation for two reasons: SCTP contains built-in features to handle multiple streams and multi homing as described in 3.4.2, and for research and educational purposes.

3.4.1 The Stream Control Transport Protocol

The Stream Control Transport Protocol (SCTP) is an evolving protocol and a proposed IETF standard (RFC2960) [20]. This stream control transmission protocol is "designed to expand the scope beyond TCP and UDP" [11] and has several benefits for transmitting

streaming media, such as offering multiple streams in the same association, association establishment that prevents SYN attacks, message-oriented data delivery service and multi homing, the ability to transparently switch between different IP-adresses on a host. The protocol has proved to reduce the latency of multiple file transfers [13] and deliver more consistent, higher quality real time streams [2].

3.4.2 Useful Features for the Dynamic Encryption Service

Some of the features in SCTP can be useful in different situations in the dynamic encryption service. The multi homing feature can be useful in the case where a host travels between two different networks and the security level changes between the networks (see section 2.6.5). Multiple streams give the advantage of separating control data and application data, which makes the control channel independent of the data channel and thereby it is possible to send control data regardless of the state of the data channel. The protocol also provides an option to send unordered messages, which can be valuable in cases when retransmission affects the time-duration too much in a time critical system and small data loss can be disregarded.

Many of the areas presented in section 2.6 involve streaming media which often consist of a large amount of data and therefore are computationally intensive and require much bandwidth. In [2] the authors demonstrate several advantages of SCTP when transmitting streaming media which gives us further arguments in using the SCTP protocol.

3.5 The Encryption Mask

The encryption mask must be shared between the sender and the receiver in order to be able to carry out the partial encryption process. Depending on how often the encryption mask is to be updated, the exchange of the encryption mask may be done in different ways. If the user application wishes to specify an encryption mask, and thereby the encryption

level, based on the content of the data rather than the structure of the data, the mask may be changed quite often. In this case, the length of the mask should be kept constant, and transmitted in a separate control message prepending the data, to allow the mask to be bundled with the data to minimize the overhead generated from changing the mask. If the user application wishes to specify an encryption mask based on the structure of the data to be sent, for example to ensure that I-frames are encrypted during transmisson of an MPEG file, the sensitive part of the data may be found in a repeated pattern. In this case, the same encryption mask could be valid for a large part of the transfer, which leads to a low frequency change of the encryption mask, and the encryption mask together with the length of the mask should be sent in a control message prepending the data. The same argument is applicable when the mask only is to be changed when the encryption level is changed, the case where the application does not want to specify the encryption mask but only the encryption level. In general, all control messages concerning the encryption mask should preferably be sent over the data channel to obtain automatic syncronization between the control message and the data it concerns.

3.5.1 Encrypting the Encryption Mask

As mentioned in section 3.2, the encryption mask specifies which blocks to encrypt. Since the receiver needs the encryption mask to determine which blocks to decrypt, the encryption mask may be viewed as a key. Thus the combination of the data encryption key and the encryption mask constructs the complete key needed to successfully decrypt the message. This viewpoint implies that the encryption mask too needs to be kept secret, and thus encrypted, during transmission. By encrypting the encryption mask using the same block cipher that is used to encrypt the control data, no further complexity is added. The use of a block cipher will require the mask to be padded to reach the length of an even multiple of the block size. In order to avoid introducing further complexity by adding another encryption algorithm into the service, the mask will be encrypted using the same

block cipher as the one used to encrypt data.

3.6 Control Channel

To control the transmission of the application data, a control channel is established where several types of messages can be sent. Both a control channel from sender to receiver and a control channel from receiver to sender are established, since the hosts need to communicate with each other. The messages can be of the following types:

1. The encryption key that is going to be used by the sender and the receiver through the session is exchanged before the encryption can begin.

2. Negotiation of the encryption level which could, for example, occur as follows (see figure 3.5): The sender is able to send the data with an encryption level of 80% and informs the receiver about this. The receiver can only manage to handle the data with an encryption level of 60% and send a response to the sender with this information. The sender accepts this encryption level and sends an acknowledgment to the receiver. Now the sender starts to transmit application data with an encryption level of 60%.

3. If an error occurs a message will be sent in the control channel. This error could be a request of changing encryption level when, for example, the sender becomes overloaded with processes with no ability to continue with the negotiated encryption level. The error could also be a request to close the session where either the sender or the receiver wants to end the transmission before completeness.

4. A negotiation mode can be sent to determine which hosts are able to participate in the negotiation. For example:

    - Mode 1: The sender decides the encryption level and the receiver has to accept or

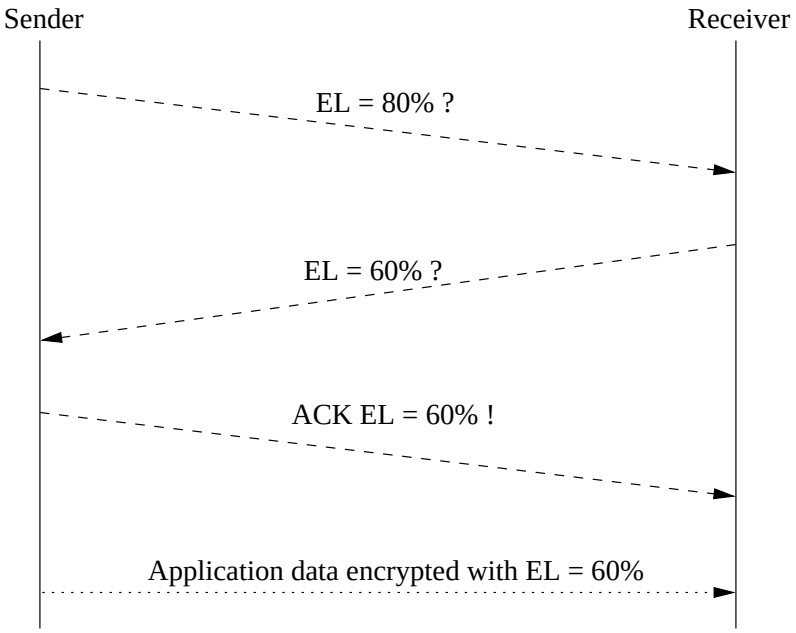


Figure 3.5: Negotiation of the encryption level

reject (and thereby close the connection). The encryption level can be changed, only by the sender, during the session.

- Mode 2: The sender and the receiver are able to negotiate the encryption level before or during the session, where the sender has the control over the decided encryption level.

- Mode 3: The receiver decides the encryption level and the sender has to accept or reject. The encryption level can be changed, only by the receiver, during the session.

The test implementation described in chapter four will focus on Mode 1.

3.7 The Data Channel

The data channel is used to transmit both the application data and the encryption mask (see section 3.5). The application data is transmitted from sender to receiver in *data units* (see figure 3.6) with the same size as the block size of the used encryption algorithm. The encryption mask and the length of the encryption mask are transmitted in *control units* in front of the data units or a *data sequence* (i.e. several data units). All the data composed of a control unit and following data sequence or a data unit is denoted as an *encryption sequence*.

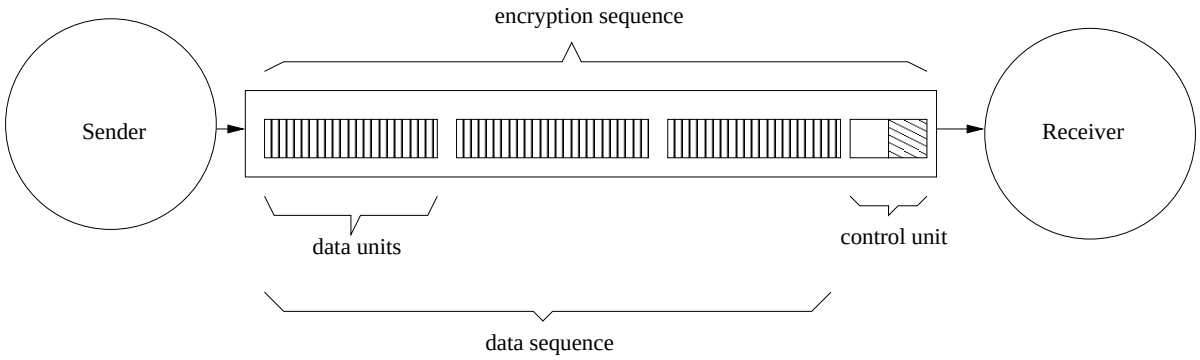


Figure 3.6: Terminology of the data in the data channel.

3.7.1 The Encryption Mask and the Encryption Mask Length

The very first unit in a transmission contains the encryption mask and the encryption mask's length. The length has to be transmitted when the mask's length can vary from one encryption sequence to another encryption sequence. If, for example, as seen in figure 3.7, a mask of length 16 is transmitted in front of 16 blocks of application data, the receiver knows, out of the mask's length, that at least 16 data blocks follow the mask and when the receiver reads the mask it knows which blocks to decrypt and which blocks to leave as plain text. The next encryption sequence, (see figure 3.7) has the mask length 32 and

thereby the receiver knows that at least 32 blocks follow the mask and the receiver then can decrypt these blocks or forward them as plain text depending on the content of the encryption mask.

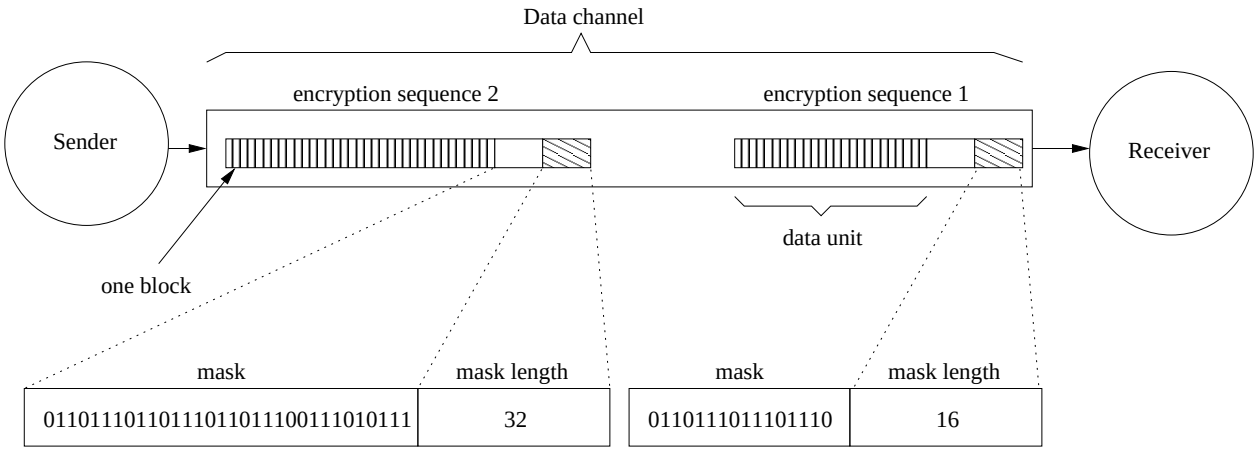


Figure 3.7: Transmission of encryption mask length, encryption mask and data.

The encryption mask can, however, be valid for several data sequences and last until the next mask is transmitted. Thereby the overhead can be minimized to a minimum. The mask may be changed at any given point of the session. The change will take effect immediately, disqualifying the currently used mask even if it is not fully traversed. If the end of the encryption mask is reached, and no new mask has arrived, the mask is reused.

3.7.2 Separating the Data and the Encryption Mask

When the application data is sent in the same channel as the encryption mask, specifying the application data to follow, and the padding, specifying how much of the received block is padded, a solution for separating the three different kinds of data is needed. By using a flag, in the message header, indicating if the transmitted data is application data, an encryption mask, or padding information, the receiver can determine what kind of data that is received.

3.7.3 Padding

If an encryption algorithm with a 128 bit block size is used and if the last bits of the application data or the control data do not correspond to the 128 bit block it has to be padded. If for example (see figure 3.8) the last data block to be transmitted only consists of 84 bits, 44 bits of padding have to be added.

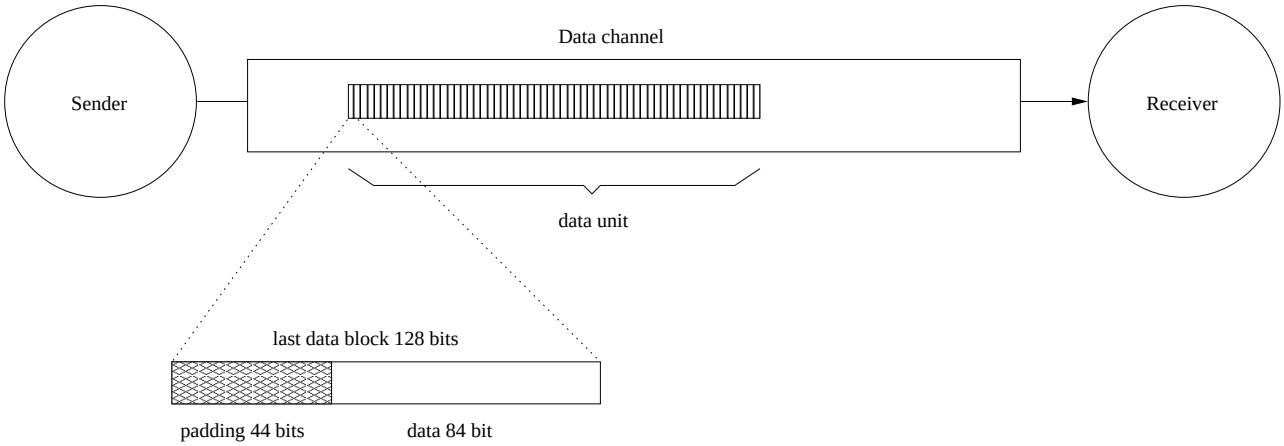


Figure 3.8: Padding of a data block

The padding information has to be sent as a control unit in front of the padded data block which gives the receiver information of how many bits are padded and thereby how many bits to read as relevant data.

3.8 Summary of the Dynamic Encryption Model

The Dynamic Encryption model consists of a sender A and a receiver B (see figure 3.9). Between A and B three channels are established where one control channel and one data channel transmit data from A to B. The third channel is a control channel transmitting control data from B to A. Control data is data that has no direct relation to the data transmitted in the data channel. The messages in the control channel are of the types:

end-of-session, encryption key, negotiation of encryption level, or encryption algorithm, and error messages. The first message in the data channel is the encryption mask showing which of the following blocks of data that are encrypted or sent as plain text. If the pattern of encrypted blocks changes during the session, a new encryption mask has to be sent over the data channel.

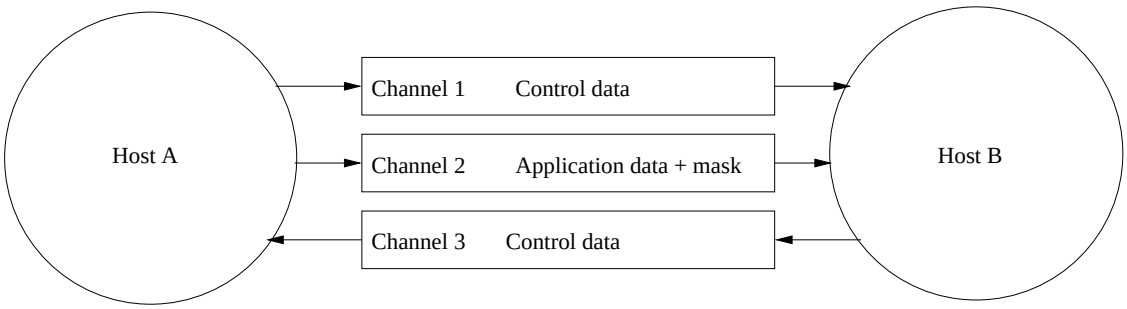


Figure 3.9: A picture over the DE model

Chapter 4

Test Implementation

This chapter describes the test implementation. The target of the test implementation is presented in section 4.1. An overview of the test implementation model is given in section 4.2. In section 4.3 an analytical evaluation of the computational gain using the dynamic encryption service, is presented. In section 4.4, the test cases, used to evaluate and test the dynamic encryption service are presented, and finally the result of these test cases are presented in section 4.5.

4.1 Target of Test Implementation

In the test implementation, the encryption level can only be changed through the encryption mask. Thereby it is possible to both specify the encryption level and which blocks out of the application data that are going to be encrypted. On top of this implementation, several modules can be implemented which give the possibility to offer an application interface that simplifies the user application's interaction with the dynamic encryption service (see section 6).
The dynamic encryption test implementation gives the ability to:

- Measure the overhead time of exchanging mask, compared to encryption and decryp-

tion in the traditional way with no encryption mask.

- Estimate and decide the granularity of the encryption level. If it is possible to see a difference in the measurement decreasing the encryption level in steps of 1% the granularity should be held at 1/100, but if it requires a decrease in steps of 5% to see a difference, the granularity should be 5/100.

- Show the savings achieved by using the dynamic encryption service.

4.2 Test Implementation Overview

An overview of the dynamic encryption model is presented in figure 4.1. The data to be transmitted is sent together with the encryption mask to the dynamic encryption module, notation (1) in figure 4.1. The encryption mask points out which data to be encrypted, notation (2.*a*), which include the encryption mask itself, and which data to be sent as plain text, notation (2.*b*). The encryption algorithm produces encrypted data, notation (3), which consists of encrypted application data and encrypted encryption masks. This encrypted data together with the plain text data are transmitted, notation (4), to the receiver via the transport layer. At the receiver the transport layer delivers the data, notation (5), to the dynamic encryption module which, on the basis of the decrypted encryption mask, notation (7), decides whether the data is encrypted and therefore should be decrypted, notation (6), or pass as plain text, notation (8). The plain text data and the decrypted data, notation (9), represents the complete data to be delivered at the receiver, notation (10).

4.2.1 The Program Modules

The test implementation of the dynamic encryption model consists of four modules of which two modules comprise the actual service (encryption algorithm and dynamic encryption),

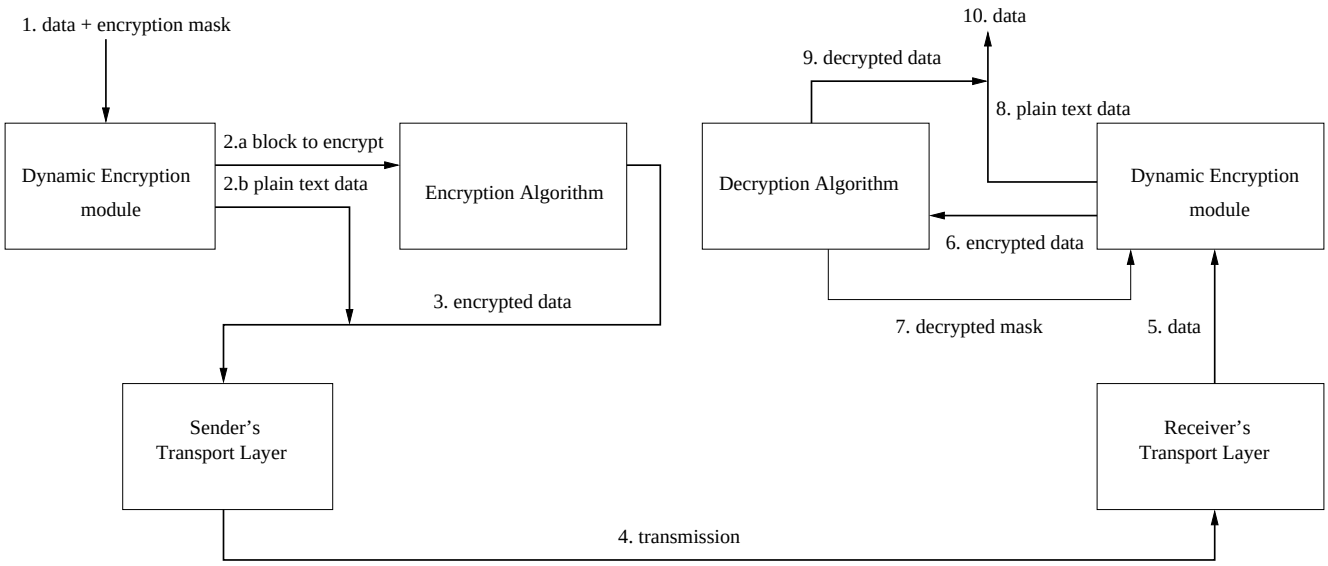


Figure 4.1: An overview of the dynamic encryption model

the remaining two modules act as server and client through the tests. To accomplish the actual encryption, an encryption module has been implemented using the AES-functions from the openSSL-library included in FreeBSD. The openSSL-functions used by this module are:

`AES_set_encrypt_key`

and

`AES_set_decrypt_key`

which both take the user key and the length, in bits, of the user key as parameters and produce the encryption key schedule and the decryption key schedule respectively in their third parameter, and

`AES_ecb_encrypt`

which takes four parameters:

- A buffer containing the data to process.
- A buffer to hold the processed data.
- The key schedule to use in the process.
- A flag determining whether to perform an encryption or decryption process.

The encryption module simply encapsulates the open-SSL functions and adds the possibility to retrieve the name and block length of the algorithm. The dynamic encryption module contains the actual service with functions to set up and tear down a connection between two hosts, send and retreive data, and manipulate the encryption mask. The functions used to set up and tear down a connection encapsulate the functions specified in [19]. The actual transmission is done in read and write functions, containing the selection mechanism, via sctp_sendmsg [19] and sctp_recvmsg [19]. The write function continously checks the size of the data to transmit and adds padding to the data and notifies the receiving end if nessecery. When the mask is changed the dynamic encryption module records the change but the new mask is not transmitted until a following call to the write-function is done. A typical transmission will be done in the following steps:

server module:

```
create DE
setup
await connection
```

which will initiate the dynamic encryption service, create a socket and wait for incoming requests

```
negotiate algorithm and exchange key
```

in the test implemetation the negotiation will be done implicitly

```
create algorithm
algorithm.setkey(key)
DE.setAlgorithm
```

theese steps will setup the dynamic encryption service with the negotiated encryption algorithm and key

```
DE.setmask(mask)
```

when the encryption mask is set the transmission can start

```
DE.write(data) (repeated)
DE.setmask(newMask)
```

the encryption mask may be altered at any point during the transmission

```
DE.write(data) (repeated)
...
DE.sendEOF
```

this step will tear down the connection and free allocated resources.
The client module operates in a similar manner:

```
create DE
setup
connect to server
```

which will initiate the dynamic encryption service, create a socket and connect to the server

```
negotiate algorithm and exchange key (Implicit)
create algorithm
algorithm.setkey(key)
DE.setAlgorithm
```

the negotiation and setup of encryption algorithm and key corresponds directly to the same steps in the server

```
DE.read(data) (repeated)
...
DE.close
```

when the client gets notification from the server the connection is closed and and allocated resources are freed.

Encryption

This implemetation is using three functions from the open-SSl/AES library. The key initialization is done via AES_set_encrypt_key(key, 128, encryptionKey) and AES_set_decrypt_key(key, 128, decryptionKey) in the algorithm.setkey(key) call in the example on page 41. The encryption is done by calling AES_ecb_encrypt(data, result, encryptionKey, AES_ENCRYPT). The AES_ecb_encrypt also performs decryption when the decryptionKey is passed and the last parameter is set to AES_DECRYPT.

Selection Mechanism

At this level, the data will be viewed as a stream which the selection is to be applied to. The selection is performed by an if-else statement with the condition containing a logical AND-operation given the current mask position, obtained with a shift operation, and TRUE as arguments. If the condition validates as TRUE the current block in the source data stream will be encrypted/decrypted and placed in the result data stream, else the current block of data will be copied from the source stream to the result data stream without modification. To keep track of the current position in the mask and in the data streams two counters are used, a mask position counter and a stream position counter. The mask position counter, named position in the code sample (page 41), is stored between

subsequent calls and will be resetted when the end of the mask is reached. The stream position counter, named i in the code sample (page 41), is also used to determine whether padding is needed to obtain a full block at the end of the sender side source stream, in which case the receiver will be notified of the size of the padding to be able to remove the padding from its result data stream.

Selection code sample[1]:

```
10: if((mask >> position) & 1)
20: {
30:    alg.encrypt(result[i],source[i]);
40: }
50: else
60: {
70:    bcopy(source[i],result[i],blockSize);
80: }
90: i++;
100: position++;
110: if(position >= maskLength)
120: {
130:    position = 0;
140: }
```

These steps are repeated in a loop until the end of the stream is reached.

[1]The last if-statement could be written as position %= masklength, but the if-statement allows for a smaller set of assembly code instructions, providing a more simple calculation of the required machine instructions

4.3 Theoretical Gain

An analytical evaluation of the computational gains that can be achieved by using the dynamic encryption service is discussed in this section. In [4], the major AES candidates are compared and evaluated with respect to performance. After that paper was published, the Rijndael algorithm [6] has been elected as the AES algorithm. By using the results from [4], it is possible to calculate a theoretical measure of the gain with our dynamic encryption service at different encryption levels (ELs). We base our further calculations on the results reported for the Rijndael algorithm only. In the following, the AES algorithm is a synonym for the Rijndael algorithm. Furthermore, we use the figures reported for a C code implementation of the algorithm using a 128-bit key on a 32-bit Pentium Pro processor.

According to [4], the time it takes to encrypt one data block is 440 clock cycles using the AES algorithm. A block containing 128 bits corresponds to 16 bytes. Consider, for example, an MP3 file of size 3 678 040 bytes. The file consists of 229 878 blocks in total. The last block must, however, be padded with 8 extra bytes, since $3\,678\,040 \, mod \, 16 = 8$. The computational time it takes to encrypt the complete file, denoted C, will then be:

$$C = 229\,878\,blocks * 440\,clock\,cycles/block = 101\,146\,320\,clock\,cycles$$

If not all of the blocks are encrypted, computational gains could be achieved. In the worst case scenario, the time required to only copy a block of plaintext data from a source to a target destination require no more then 32 clock cycles per block. Two MOV instructions are used to first copy a byte from memory to a register and then from the register to memory. Each MOV instruction is specified to execute within 1 clock cycle. Hence, to copy a block, which contain 128 bits = 16 bytes, 2 * 16 = 32 clock cycles are needed.

The selection mechanism to specify whether a block should be encrypted or not could be implemented in the C programming language as described in the code sample on page

Instruction	WCET† (clock cycles)	Occurences rowline
SHIFT	1	10
AND	2	10
CMP	4	10, 110
JNZ or JMP	1	20, 60, 120
INC	4	90, 100
JLE	1	10, 110
MOV	8	70, 130

Table 4.1: Assembly instructions

† WCET: Worst-Case Execution Time.

41.

In the code sample, `mask` is the bit mask and `i` is used as an index. Furthermore, `result[i]` and `source[i]` refers to ciphertext block i and plaintext block i respectively, `position` refers to the current position in the mask, and `maskLength` denotes the length of the bit vector.

The selection mechanism described above could thus be implemented using the following minimal set of machine code instructions on a Pentium Pro processor:

The combination of the code sample on page 41 and table 4.1 implies that for each block, the selection mechanism produces an overhead of 21 clock cycles in the case where the `position` is not resetted. In the case where `position` is resetted the overhead produced will be 28 clock cycles. The formula for calculating the estimated number of clock cycles needed to encrypt a file using the dynamic encryption service and a mask length equal to the number of blocks + 1 can then be written as follows:

$$C = 21n + 440n_e + 32(n - n_e) \tag{4.1}$$

Where n is the total number of blocks in a data transfer, and n_e is the number of blocks that are encrypted.

The formula for calculating the estimated number of clock cycles needed to encrypt a file using the dynamic encryption service and a mask length equal to 1 can then be written as follows:

$$C = 24n + 440n_e + 32(n - n_e) \tag{4.2}$$

The general formula for calculating the estimated number of clock cycles needed to encrypt a file using the dynamic encryption service can then be written as follows:

$$C = 21n + 3n/m + 440n_e + 32(n - n_e) \tag{4.3}$$

Where n is the total number of blocks in a data transfer, n_e is the number of blocks that are encrypted, and m is the length of the mask used. By using equation 4.3, we can calculate at which encryption level the overhead for the dynamic encryption service is greater than or equal to the overhead for encrypting all the blocks without the selection mechanism.

$$24n + 440n_e + 32(n - n_e) \leq 440n \tag{4.4}$$

Rewriting the expressions in (4.4) and (4.2) gives us:

$$\frac{n_e}{n} \leq \frac{387}{411} \approx 0.94 \tag{4.5}$$

for the case when the mask length is set to 1, and

$$\frac{n_e}{n} \leq \frac{387}{408} \approx 0.95 \tag{4.6}$$

for the case when the mask length is set to number of blocks + 1.

This implies that as long as the amount of encrypted blocks in a data transfer is equal to or less than 95 %, the dynamic encryption service produces less overhead than encrypt-

ing everything. This estimate is, however, expected to be optimistic. This is due to the fact that the given selection mechanism contains an absolute minimal set of machine code instructions. Hence, additional instructions might be needed in a real implementation. Remember also that this evaluation is based on measures on a quite old processor. Newer processor architectures are faster, mostly because of increased clock frequency, but also partly because of more efficient machine code instructions. The analytical evaluation described in this section is completely based on computational time measured in clock cycles, which means that the clock frequency will not affect the estimate. More efficient machine code instructions, on the other hand, will affect the estimate. The increased speed of newer processors is, however, mostly due to increased clock frequency.

4.4 Test Cases

In order to obtain empirical results which support our thesis, we perform several test cases. The purpose with these test cases are to verify that:

1. The time needed to encrypt the data with a low encryption level, is less than the time needed to encrypt the data with a high encryption level.

2. The time needed to encrypt the data with a low encryption level, including the overhead given by the dynamic encryption module, is less than full encryption without using the dynamic encryption module.

3. The correct amount of data, and the correct blocks of the data, is encrypted according to the specification in the encryption mask.

4. The overhead cost using an encryption mask is relatively small.

5. The granularity to be used for the encryption level.

6. The time needed to encrypt and transmit the data with a low transmission rate is less than the time needed with a higher transmission rate.

All test cases are done with the AES-ECB-128 encryption algorithm and are performed between two computers connected on an isolated network, 1GB Ethernet.
Each test is repeated 40 times, giving the ability to see the variation and be able to calculate the mean.

The figure 4.2 shows an overview of the test environment where A represents the sender,

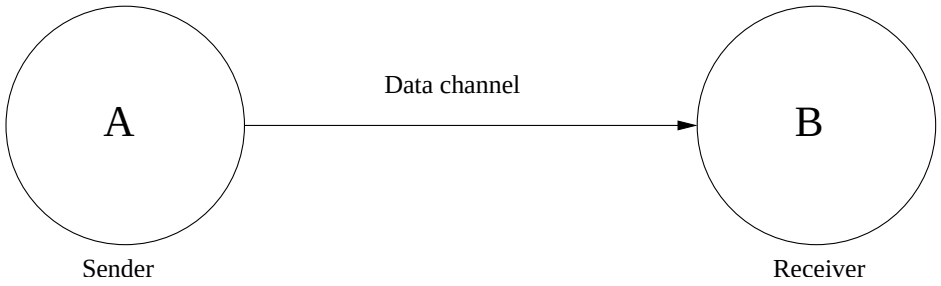


Figure 4.2: Schematic picture of the test environment

and B represents the receiver. Since the network used in the tests is isolated and without additional load, time measurements are only done at the sender side. Furthermore the encryption algorithm is symmetric which means the encryption time and the decryption time will be equal.

4.4.1 Test Environment

The hardware and software presented in table 4.2 were used in all the test cases.

4.4.2 Functionallity Test, Test Case 1

In this test case we send data from A to B and count the amount of data that has been encrypted. We compare the received data blocks pattern at B with the pattern of the

Computer	Dell OptiPlex GX270
Processor	P4 2.8 GHz
RAM	512 MB
Disc Drive	Western Digital 80 GB model: WD800BB-75FRA0
OS	FreeBSD 4.9
SCTP version	KAME 2004-04-05
Compiler	GCC-2.95.4
Compiler flags	-Wall -pedantic
AES Implementation	AES-ecb-128 OpenSSL 0.9.7c

Table 4.2: Used hardware and software

encryption mask. This test case corresponds to purpose 3 in section 4.4. The transmitted data contains 32 bytes of each letter in the alphabet from a to z (26 letters $\times$ 32 bytes = 832 bytes). The length of the encryption mask varies in the following tests (**a-e**):

a) EL = 100%, mask length = 1, pattern = 1.

b) EL = 75%, mask length = 4, pattern = 1110.

c) EL = 50%, mask length = 2, pattern = 10.

d) EL = 25%, mask length = 4, pattern = 1000.

e) EL = 0%, mask length = 1, pattern = 0.

Goal: To show that the data is encrypted as specified by the encryption mask.

4.4.3 Dynamic Encryption Behaviour Without Transmission, Test Case 2

In this test case we encrypt data and measure the time needed for different file sizes. This test case corresponds to purposes 1 and 5 in section 4.4. Files of 3 different sizes: 1 MByte, 10 MByte, and 100MByte are encrypted. No transmission of the data is done in this test

case.

a) Using pure AES

b) Using our dynamic encryption service.

The encryption level is increased in steps of 12.5% from 0 to 100%.

Goal: To show that the needed time to encrypt the data increases for each encryption level going from 0 to 100% using the dynamic encryption service, and also estimate the granularity that should be used by reading where the graphs, for using pure AES and our dynamic encryption service, intersect each other.

4.4.4 Dynamic Encryption Behaviour With Transmission, Test Case 3

In this test case, we encrypt the data, send it from A to B and measure the needed time at A. This test case corresponds to purposes 1 and 2 in section 4.4. Files of 3 different sizes are transmitted: 1 MByte, 10 MByte, and 100MByte.

a) Using pure AES

b) Using our dynamic encryption service.

The encryption level is increased in steps of 12.5% from 0 to 100%.

Goal: To show that the needed time, to encrypt and send the data, increases for each encryption level going from 0 to 100% using our dynamic encryption service, and also show the overhead cost for the dynamic encryption service, at 100% encryption level, compared with pure AES. Compare the time needed, in this test case, with test case 2 (4.4.3).

4.4.5 Network Speed Effect, Test Case 4

In this test case, we transmit 10 MByte data from A to B and measure the time needed at A for four different transmission rates, which corresponds to purpose 6 in section 4.4:

a) Using pure AES

1000Mbps Gigabit Ethernet

100Mbps Fast Ethernet

54Mbps IEEE 802.11g

11Mbps IEEE 802.11b

b) Using our dynamic encryption service.

The encryption level is increased in steps of 12.5% from 0 to 100%.

1000Mbps Gigabit Ethernet

100Mbps Fast Ethernet

54Mbps IEEE 802.11g

11Mbps IEEE 802.11b

Goal: To investigate if the time needed to encrypt and send the data increases for each lower transmission rate. To investigate if the lower transmission rate used, the less influence the needed timed to encrypt the data has.

4.4.6 Encryption Mask Alteration Overhead, Test Case 5

In this test case, we transmit data from A to B and measure the time needed. This test case corresponds to purpose 4 in section 4.4. The size of the data to be sent is 10 MByte and the encryption mask is changed with different frequency. The encryption level of the encryption mask is held constant at 50% in all tests (**a-f**).

a) The same encryption mask is used throughout the session.

b) The pattern of the mask is changed 1999 times during the lifetime of the session.

c) The pattern of the mask is changed 2499 times during the lifetime of the session.

d) The pattern of the mask is changed 3333 times during the lifetime of the session.

e) The pattern of the mask is changed 4999 times during the lifetime of the session.

f) The pattern of the mask is changed 9999 times during the lifetime of the session.

Goal: To show that the only added cost is the overhead when changing the mask's pattern. Show the relation between the time needed to change the encryption mask, and the total time needed for encryption and transmission of the data.

4.4.7 Key Initiation, Test Case 6

Before performing the actual encryption the key to be used needs to be initiated.This initiation will expand the user key into round keys for encryption and decryption. This process is a part of the AES encryption process and is not specific to the dynamic encryption service. In this test case, we measure the time needed for the sender to initiate the encryption key schedule and the decryption key schedule for the encryption algorithm, AES-ECB-128.

Goal: This is done with the purpose to see how much the key initialization affects the total time for the dynamic encryption service, as this time is not included in the other test cases. Both the decryption key schedule initiation and the encryption key schedule initiation are measured 20 times and the mean is calculated from these measurements.

4.5 Test Case Results

All the results in the test cases are calculated with a confidence interval of 95% out of the mean. The confidence interval showed to have a very small range and are therefore withdrawn from the diagrams though they are not visible.

The time for reading data from disc is included in all test cases and we have measured

File size	**Time needed** (seconds)
1 MByte	0.00084874
10 MByte	0.00850187
100 MByte	0.24878186

Table 4.3: Time needed to read data from disc

this time separately for each file size. The results presented in table 4.3 are the average of 100 repetitions.

4.5.1 Functionallity Test, Test Case 1

The input file contains 32 bytes of each letter in the alphabet from a to z, and is the file sent from sender side A. The output file is the created file out of the received data at B. The input file and the output file are presented in Appendix A and show the result after transmitting data from A to B and where an output file is created at the receiver before the data is decrypted. A comparison between the input file and the output file shows how much data is encrypted, where one line (16 characters) corresponds to 25% of the data amount, 2 lines corresponds to 50% etc.. The input file is transmitted repeatedly until 2560 bytes are transmitted (512 bytes per encryption level), where the encryption decreases from 100% down to 0% in steps of 25% (see figure 4.3).

Result: The files in Appendix A shows that the intended functionality agrees with the result.

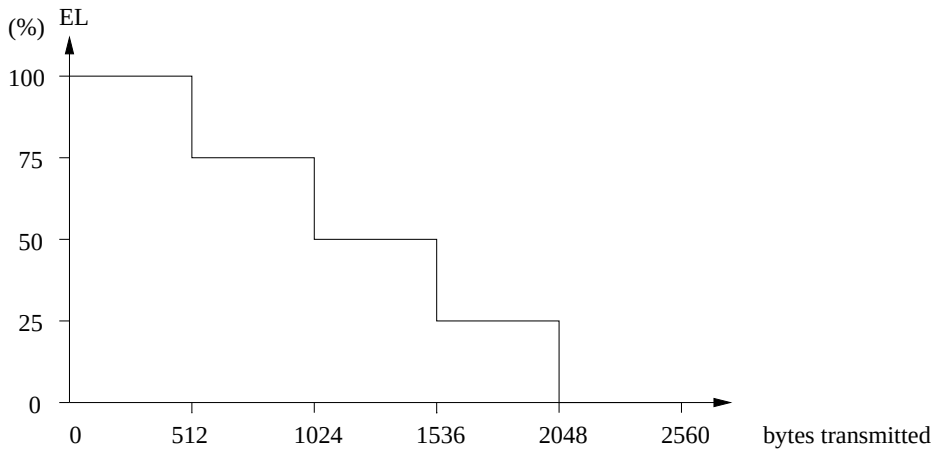
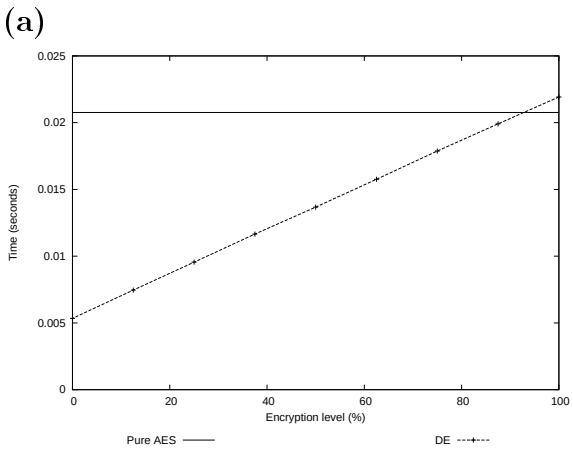


Figure 4.3: Test case 1, where the encryption level is decreasing.

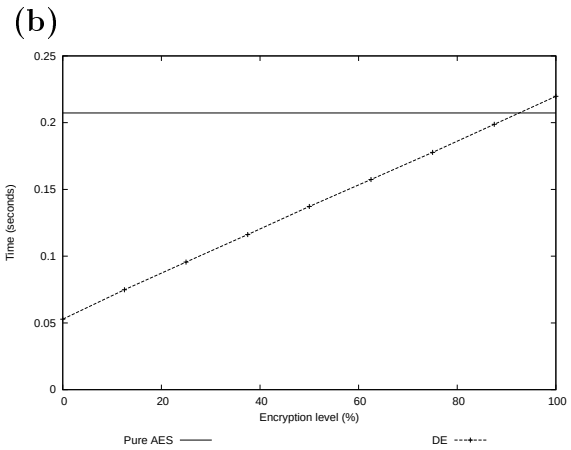
4.5.2 Dynamic Encryption Behaviour Without Transmission, Test Case 2

Figure 4.4 shows the results when only encrypting with the dynamic encryption service, without transmission of the data. As long as the needed time for the dynamic encryption service is less than the needed time for pure AES, there is a benefit in using our dynamic encryption service. These diagrams give a picture of how the dynamic encryption behaves when the computational time is not influenced by the network. The figure shows that encryption time is an almost linear function of the encryption level for all tested file sizes. An adjustment in encryption level will give a corresponding reaction in encryption time.

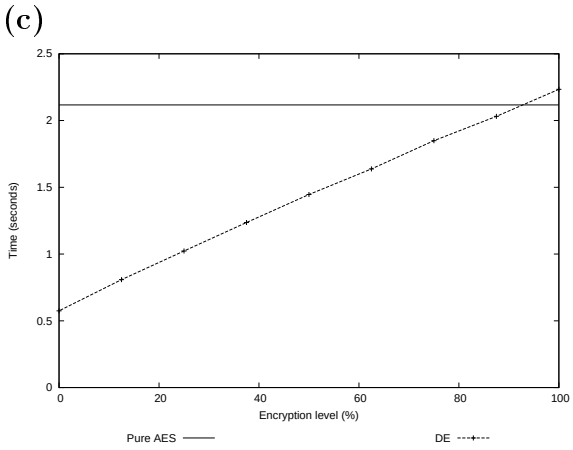
Result: Hereby we conclude that test case 2 shows that the time needed to encrypt and send the data decreases as the EL is lowered.
Reading the diagrams in figure 4.4, we can see where the graphs intersect each other, which is at approximately 95% encryption level in accordance with the analogy in section 4.3.



Test case 2, 1MB file



Test case 2, 10MB file



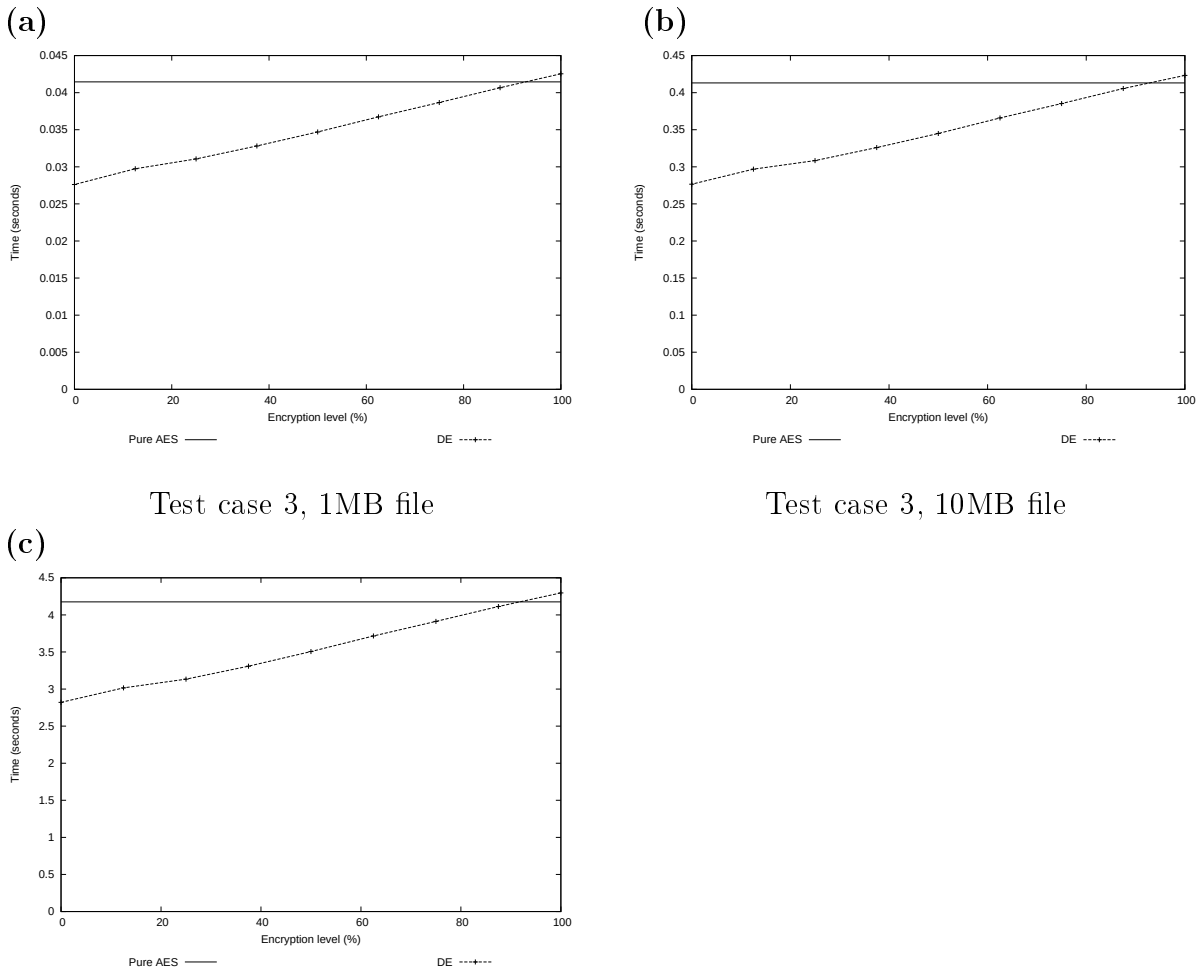
Test case 2, 100MB file

Figure 4.4: Encryption time as a function of encryption level for different amounts of data, test case 2a and 2b. Also shown is the time for encryption without using the dynamic encryption service.

4.5.3 Dynamic Encryption Behaviour With Transmission, Test Case 3

Figure 4.5 shows the results when encrypting and transmitting the data over SCTP. As long as the time needed by the dynamic encryption service is less than with pure AES, there is a benefit in using our dynamic encryption service.

Result: Hereby we conclude that test case 3 also, as in test case 2, shows that the time needed, to encrypt and send the data, decreases as the EL is lowered.
By reading the diagrams in figure 4.5 we can see that the overhead cost for the dynamic encryption service at 100% encryption level is approximately 5%.
When comparing the diagrams in figure 4.5 with the diagrams in figure 4.4 we can see that the transmission adds a time of approximately 2 seconds for a 100MB file, 0.2 seconds for a 10MB file and 0.02 seconds for a 1MB file.



Test case 3, 1MB file

Test case 3, 10MB file

Test case 3, 100MB file

Figure 4.5: Encryption and transmission time as a function of encryption level for different amounts of data, test case 3a and 3b. Also shown is the time for encryption and transmission without using the dynamic encryption service.

4.5.4 Network Speed Effect, Test Case 4

Table 4.4 shows the results from transmitting 10 MB data with four different transmission rates using pure AES.

Note: Values of Time are rounded to four decimals.

Transmission rate (Mbps)	1000	100	54	11
Time (seconds)	0.4131	0.8573	1.5878	7.7950

Table 4.4: Time needed for transmitting data with different transmission rates, test case 4a. (Pure AES)

Table 4.5 shows the results from transmitting 10 MB data with four different transmission rates using our dynamic encryption service.

EL (%)	**Transmission rate** (Mbps)			
	1000	100	54	11
	Time (seconds)			
0	0.2765	0.8573	1.5878	7.7950
12.5	0.2968	0.8573	1.5878	7.7950
25	0.3083	0.8573	1.5878	7.7950
37.5	0.3259	0.8573	1.5878	7.7950
50	0.3449	0.8573	1.5878	7.7950
62.5	0.3659	0.8573	1.5878	7.7950
75	0.3853	0.8573	1.5878	7.7950
87.5	0.4056	0.8573	1.5878	7.7950
100	0.4232	0.8573	1.5878	7.7950

Table 4.5: Time needed for transmitting data with different transmission rates at different encryption levels, test case 4b. (dynamic encryption service)

Result: For transmission rates lower than 100 Mbps the time needed for encryption does not affect the total processing time. The overhead is almost independent of the encryption level.

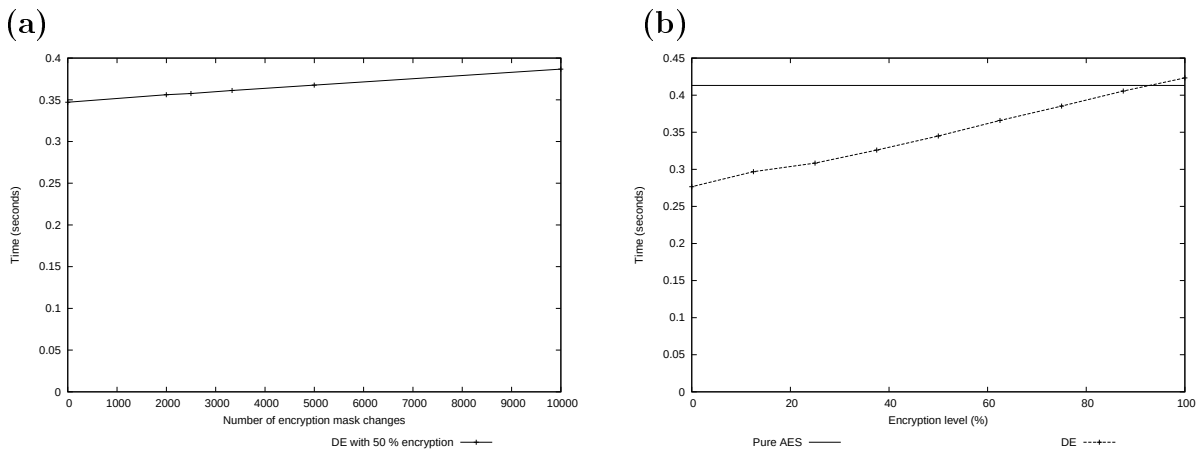
4.5.5 Encryption Mask Alteration Overhead, Test Case 5

Test case 5 is presented in figure 4.6 where the file of size 10 MByte is used and transmitted in six different transmissions with EL = 50%. For each transmission the encryption mask is changed different number of times, namely 0, 1999, 2499, 3333, 4999 and 9999 times.

Number of mask changes	**Time needed** (seconds)
0	0.3471
1999	0.3561
2499	0.3575
3333	0.3611
4999	0.3676
9999	0.3867

Table 4.6: Time needed for transmitting data at different frequency of mask changes, test case 5.

Result: The time needed (taken from table 4.6) to transmit with no mask changes is 0.3471 seconds , and the time needed for transmission with 9999 mask changes is 0.3867 seconds. This gives us the difference $0.3867 - 0.3471 = 0.0396$ seconds, which is the overhead cost for changing the encryption mask 9999 times. The time needed for no changes of the mask, compared to the time needed to change the encryption mask 9999 times gives us, $\frac{0.0396}{0.3471} = 0.1141 \approx 11.4\%$ overhead. The influence of changing the mask can also be seen by comparing the diagrams in figure 4.6 a) and b), where the gradient of the curve in figure 4.6 a) is much smaller than the gradient in figure 4.6 b).



Test case 5, 10MB file

Test case 3, 10MB file

Figure 4.6: Figure a) shows the encryption and transmission time as a function of mask change frequency at encryption level 50%. Figure b) shows the transmission of the same file with only one change of the encryption mask. By reading the time needed at encryption level 50% in figure b), it can be compared to the variation of time in figure a), test case 5.

4.5.6 Key Initiation, Test Case 6

The results from measuring the time for encryption key schedule initialization and decryption key schedule initialization in the AES implementation derived as in table 4.7:

-	**Time needed** (seconds)
Encryption key initialization	0.00000475
Decryption key initialization	0.0000051

Table 4.7: Results from test case 6. Time needed for encryption and decryption key initialization

The time for encryption and decryption key schedule initialization is not included in

the other test cases, but since this time always is required we choose to present this time separately to be able to give an understanding of the size of the time needed.
From the table 4.7 it follows that the time needed for encryption and decryption key schedule initialization is very small compared to the time needed for encrypting and transmitting the data (see for example Figure 4.5). If, for example, 1 MB data is transmitted with a low EL (25%) the time needed is approximately 0.0311 seconds, taken from graph (a) in Figure 4.5. The time for key initialization (encryption and decryption) is $0.00000475 + 0.0000051 = 0.00000985$ seconds, which gives that the encryption key initialization is $\frac{0.00000985}{0.0311} \approx 0.032\%$ added to the total needed time. For bigger files the key schedule initialization part, added to the total time, is much smaller, which can be seen in Figure 4.5 (b-d).

Result: The added time, needed for key schedule initialization, is very small compared to the needed time for encryption and transmission of the data.

4.6 Discussion of test results

The purpose with the test cases was to show that:

1. The time needed to encrypt the data with a low encryption level, is less than to encrypt the data with a high encryption level.

2. The time needed to encrypt the data with a low encryption level, including the overhead given by the dynamic encryption module, is less than full encryption without using the dynamic encryption module.

3. The right amount of data is encrypted according to the specification in the encryption mask.

4. The overhead cost using an encryption mask is relatively small.

5. The granularity to be used for the encryption level.

6. The time needed to encrypt and transmit the data with a low transmission rate is less than the time needed with a higher transmission rate.

We can now draw the conclusions of our test cases corresponding to these purposes.

1. The diagrams in figure 4.4 and 4.5, which is the outcome from test case 2 and 3, shows that the needed time encrypting the data with a low encryption level is less than with a higher encryption level. This was the intended result and gives our dynamic encryption service the right behaviour in adjusting the encryption level over an Internet connection.

2. The diagram in figure 4.5, which shows the results from test case 3, proves that as long as the encryption level is kept below 90%, the time needed for using our dynamic encryption service is less than full encryption using pure AES. This gives our dynamic encryption service a valuable benefit compared to pure AES because of the gain in time and that there is no need for re-establishing the connection to get a new encryption level.

3. The files presented in Appendix A, which are the result of test case 1, show the input file (before encryption on the sender side) and the output file (before decryption at the receiver side). These files show that using a mask with encryption level (EL)=100% gives the result in the output file from row 0000 to row 0760. EL=75% gives the result from row 1000 to 1760, EL=50% gives the result from row 2000 to row 2760, EL=25% gives the result from row 3000 to row 3760 and finally EL=0% gives the result from row 4000 to row 4760 which is the message in plain text. This result proves that the dynamic encryption service transmitts the data in an appropriate way.

4. Test case 5 (see section 4.5.5) shows an overhead cost of approximately 11.4% using 9999 mask changes in transmitting 10 MB data. This seems to be a rather high value, but if we look back in section "Areas of Interest" 2.6 the majority of these proposed cases only need a few (1 to 10) mask changes. The diagrams in figure 4.6 also shows that even if the mask is changed 9999 times, there is still a small benefit in using the dynamic encryption service compared with using pure AES.

5. The granularity in test case 2 (section 4.5.2) that corresponds to the analogy in section 4.3, is a suitable level of the adjusting steps to be taken to change the level of encryption. From the graph in figure 4.5 c) we can estimate a gain in time, when lowering the EL with 5%, from 50% to 45%, at about 0.125 seconds. This is a gain of approximately $\frac{0.125}{3.5} \approx 3.6\%$ of the total time to transmit 100 MB data. From the use cases discussed in section "Areas of Interest" 2.6 we do not think there is a need for a finer granularity. With a granularity of 5% our use cases can be carried out with the intended result. This result also corresponds to the theoretical calculations in section 4.3 where the expected overhead is calculated to be around 5%.

6. In the results for test case 4 (see section 4.5.4), the time needed for transmission rate 100, 54, and 11 Mbps is the same using pure AES, or using our dynamic encryption service (see table 4.4 and 4.5). The only difference in time needed, using the dynamic encryption service, is when using a transmission rate of 1000 Mbps. From this we can draw the conclusion that the time needed to transmit data with 100, 54, and 11 Mbps affects the total time needed much more than the dynamic encryption service does. To estimate a threshold value for the maximum transmission rate achieved using the specified hardware and software presented in table 4.2, we listed the overhead in table 4.8. In table 4.9 we calculated the total number of bytes sent when the original data to be sent was 10 MB.

   The "Sum of total data" in table 4.9 is $11,020,000$ bytes which is equal to $8 *$

Header	**Bytes**
Ethernet	26
IP	24
SCTP common header	12
Chunk header	16
Sum of overhead	78

Table 4.8: Overhead per 1024 bytes

Data	**Bytes**
The data to be sent	$10,240,000$
Number of packets	$10,000$
Overhead	$780,000$
Sum of total data	$11,020,000$

Table 4.9: Total data to be sent

$11,020,000$ bits $= 88,160,000$ bits. The time to send 10 MB data with an EL of 50% from table 4.5 where the transmission rate is set to 1000 Mbps is 0.3449 seconds. This gives a maximum transmission rate of: $\frac{88,160,000}{0.3449} \approx 256$ Mbps. From this calculation we did a new test case where the transmission rate was set to 250 Mbps and 10 MB data was transmitted, to be able to see if we could show this threshold value in a graph. Figure 4.7 shows the threshold value when the transmitting rate is 250 Mbps. From this figure we can see that the threshold value of the maximum transmission rate is located at approximately 250 Mbps where there is a knee on the graph. When the encryption level (EL) is lower than 50%, the time to encrypt the data does not influence the total needed time to encrypt and transmit the data, and when the EL is higher than 50% the dynamic encryption service influences the time to transmit the data.

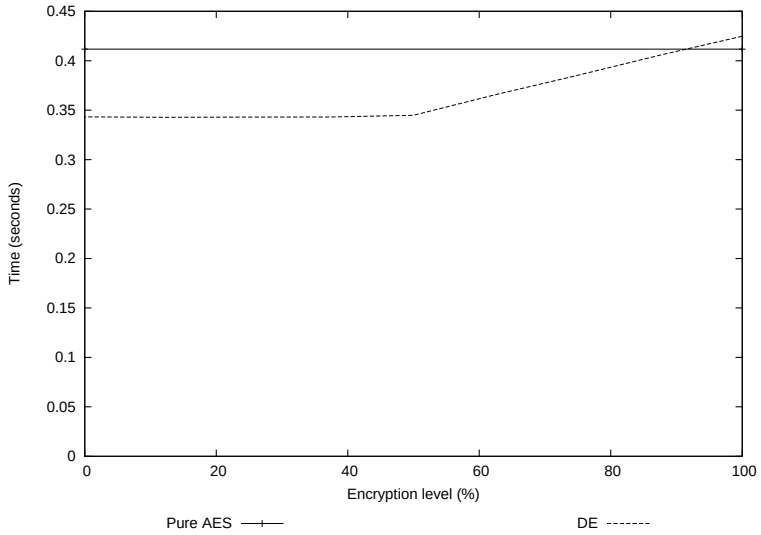


Figure 4.7: Test case, where the transmission rate is set to 250 Mbps.

We can see that the calculations described in section "Theoretical Gain" 4.3 corresponds to the achieved results in our test cases. The calculation in equation 4.5 gives a value of 96% for the amount of encrypted blocks in a data transfer that is encrypted with the dynamic encryption service, to give a less overhead than encrypting everything. From figure 4.5 we can see that this is the value we achived in our test cases.

Test case 6 (see section 4.5.6) shows that encryption key initialization and decryption key initialization have a very small impact on the time needed for transmitting the data (see table 4.7). This is also the case for reading data from disc which is presented in table 4.3. These tests were done with the purpose of eliminating undesired impact from these processes.

Chapter 5

Conclusions

The solution of today's transmission of encrypted data is relatively static in the way that no changeble parameters are offered during the present transmission. This implies that there are no possibilities to adapt the encryption to changes in the network or changes in the computer load.

The traditional way to encrypt data has been to either encrypt all the data or to encrypt no data at all. The eligible parameters used in these cases are key, key length, and encryption algorithm.

The idea behind this thesis is to offer additional changeble parameters to be able to increase the adaptiveness. By offering the ability of only encrypting a subset of the data, which also can be varied during the present transmission, the communicating parties can adapt the encryption level regarding different changes that occurs.

This thesis describes a model to accomplish this flexibility. The performed tests show that without transmission a linear ratio is achieved between the subset of encypted data and the process time. This indicates that the encryption level is a useful parameter to increase the adaptiveness regarding the computer load.

The results also show that the overhead cost for the suggested selection mechanism is relatively small, about 5%, and the results also show that the overhead cost for changing

the encryption level is negligible. Furthermore the tests show, when transmitting the data with low loaded and fast computers, that the effect is only visible when using a high bandwidth network. In our case more than 250 Mbps.

The model described in this thesis is able to give further advantages regarding energy consumption and limited processor power in handheld devices.

Chapter 6

Future work

Some of the possibilities that could be offered with the Dynamic Encryption service we choose to introduce in this chapter of future work.
Above the Dynamic Encryption implementation, described in this thesis, several modules could be applied that simplifies the API for the user application (see figure 6.1):

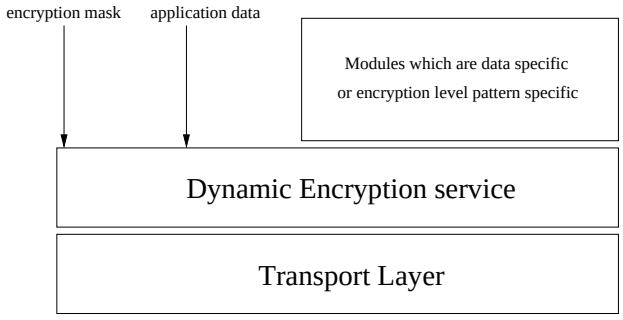


Figure 6.1: Modules above the dynamic encryption service.

- Different data could require different encryption levels to keep the data secret and at the same time save resources.

  It would be interesting to investigate a module, added above the dynamic encryption module, with access to a list of file types and the encryption level they require. The

user could thereby be able to specify the file type as an argument to the module. The module looks up the required encryption level for the specified file type and calls the dynamic encryption module with the encryption level as an argument.

- The possibility to alternate the pattern of the encryption level, as discussed in section 2.7, could be added as a module above the dynamic encryption module. The added module receives the pattern as an argument and then handles the communication with the underlying dynamic encryption module to achieve the desired pattern.

- The security could be changed by using different encryption algorithms together with the encryption mask in the dynamic encryption service. A module, added above the dynamic encryption module, giving the ability to alter the algorithm during the session would be interesting to investigate.

- As mentioned in chapter 3.5, the user may want to specify a mask pattern that corresponds to the sensitive parts in the data structure (e.g. I-Frames in MPEG files). A study in the behaviour and the outcome from this point of view would be interesting.

Another possibility is the multi homing feature in SCTP that could be used to offer increased security by transmitting the data in two physical separated channels between the hosts.

The load on the server could be tested by connecting several clients to the server and register how the load increases and how this load could be lowered by decreasing the encryption level for the data to be sent by the server (see figure 6.2). This could be a study case for the future.

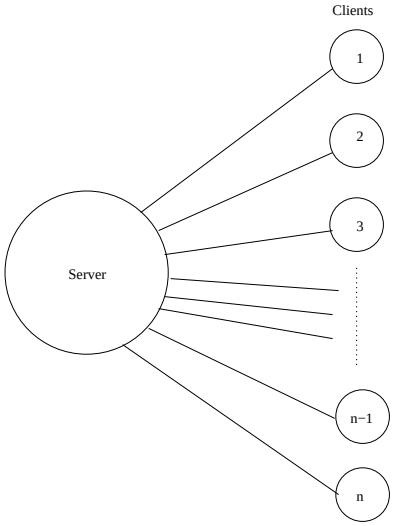


Figure 6.2: Several clients connected to a server.

Another study case, for the future, is to decrease the processor speed. Then it would be possible to observe at which processor speed the dynamic encryption service is affected when using a constant load.

References

[1] Adnan M. Alattar and Ghassan I. Al-Regib. Evaluation of Selective Encryption Techniques for Secure Transmission of MPEG-compressed Bit-streams. Technical report, Digimarc Corporation Lake Oswego Oregon, Electrical Engineering-Department King Fahd University of Petroleum and Minerals Dahran Saudi Arabia, 1999.

[2] Alex Balk, Marc Siegler, Mario Gerla, and M.Y. Sanadidi. Investigation of MPEG-4 Video Streaming Over SCTP. Technical report, Network Research Laboratory Computer Science Department UCLA Los Angeles, 2002.

[3] Simon Blake-Wilson, Magnus Nystrom, David Hopwood, Jan Mikkelsen, and Tim Wright. RFC 3546: Transport Layer Security (TLS) Extensions, 2003.

[4] Bruce Schneier and John Kelsey and Doug Whiting and David Wagner and Chris Hall and Niels Ferguson. Performance comparison of the aes submissions, 1999.

[5] Phillip T. Conrad, Gerard J. Heinz, Armando L. Caro Jr., Paul D. Amer, and John Fiore. SCTP in Battlefield Networks. Technical report, Computer and Information Science Department Temple University Philadelphia, Computer and Information Science Department University of Delaware Newark, School of Engineering and Applied Science University of Pennsylvania Philadelphia, 2001.

[6] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.

[7] Stephen E. Deering and Robert M. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) Specification, 1998.

[8] Tim Dierks and Christpher Allen. RFC 2246: The TLS Protocol Version 1.0, 1999.

[9] James Goodman and Anantha P Chandrakasan. *Low Power Scalable Encryption for Wireless Systems*, volume 4, Issue 1, pages 55–70. Kluwer Academic Publishers, Hingham, MA, USA, 1998.

[10] Janardhan R. Iyengar and Sunil Samtani. SCTP Multistreaming: Design Ideas on Preferential Treatment Among Streams, 2002.

[11] Armando L Caro JR, Janardhan R Iyengar, Paul D Amer, Sourabh Ladha, Gerard J Heinz II, and Keyur C Shah. SCTP: A Proposed Standard for Robust Internet Data Transport. Technical report, University of Delaware, 2003.

[12] Stephen Kent and Randall Atkinson. RFC 2401: Security Architecture for the Internet Protocol, 1998.

[13] Sourabh Ladha and Paul D Amer. Improving Multiple File Transfers Using SCTP Multistreaming. Technical report, Computer and Information Sciences University of Delaware, 2003.

[14] Stefan Lindskog, Anna Brunstrom, and Erland Jonsson. Dynamic Data Protection Services for Network Transfers: Concepts and Taxonomy. Technical report, Department of Computer Engineering Chalmers University of Technology Gothenburg Sweden, Department of Computer Science Karlstad University Sweden, 2004.

[15] Stefan Lindskog and Erland Jonsson. Dynamic Data Protection Services for Network Transfers: Concepts and Taxonomy. Technical report, Department of Computer Engineering, Chalmers University of Technology, Gothenburg, Sweden, 2004.

[16] Stefan Lindskog, Johan Strandbergh, Mikael Hackman, and Erland Jonsson. A Content-Independent Scalable Encryption Model. Technical report, Department of Computer Engineering Chalmers University of Technology, Department of Computer Science Karlstad University, May 14-17 2004. In Proceedings of the 2004 International Conference on Computational Science and its Applications (ICCSA 2004), Perugia, Italy.

[17] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing.* Prentice Hall PTR, 3th edition, 2003.

[18] Jon Postel. RFC 791: Internet Protocol, 1981.

[19] R. Stewart and Q. Xie and L. Yarroll and J. Wood and K. Poon and K. Fujita and M. Tuexen. Sockets API Extensions for Stream Control Transmission Protocal (SCTP) draft-ietf-tsvwg-sctpsocket-08.txt. http://www.ietf.org/proceedings/04aug/I-D/draft-ietf-tsvwg-sctpsocket-08.txt, 2004.

[20] R Stewart, Q Xie et al. Stream control transmission protocol, RFC 2960, 2000.

[21] Bruce Schneier. *Applied Cryptography.* Cloth. Wiley, UK, 2nd edition, 1996.

[22] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* . Anchor, 1th edition, 2001.

Appendix A

Test Results

A.1 Test case 2 results

The files are created with the command:

```
od -c {file}
```

where the line numbers in the print-outs are written in octal numbers.

The input file (input test data):

```
0000000    a   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a
0000020    a   a   a   a   a   a   a   a   a   a   a   a   a   a   a  \n
0000040    b   b   b   b   b   b   b   b   b   b   b   b   b   b   b   b
0000060    b   b   b   b   b   b   b   b   b   b   b   b   b   b   b  \n
0000100    c   c   c   c   c   c   c   c   c   c   c   c   c   c   c   c
0000120    c   c   c   c   c   c   c   c   c   c   c   c   c   c   c  \n
0000140    d   d   d   d   d   d   d   d   d   d   d   d   d   d   d   d
0000160    d   d   d   d   d   d   d   d   d   d   d   d   d   d   d  \n
0000200    e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e
0000220    e   e   e   e   e   e   e   e   e   e   e   e   e   e   e  \n
```

```
0000240  f  f  f  f  f  f  f  f  f  f  f  f  f  f  f  f
0000260  f  f  f  f  f  f  f  f  f  f  f  f  f  f  f  \n
0000300  g  g  g  g  g  g  g  g  g  g  g  g  g  g  g  g
0000320  g  g  g  g  g  g  g  g  g  g  g  g  g  g  g  \n
0000340  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h
0000360  h  h  h  h  h  h  h  h  h  h  h  h  h  h  h  \n
0000400  i  i  i  i  i  i  i  i  i  i  i  i  i  i  i  i
0000420  i  i  i  i  i  i  i  i  i  i  i  i  i  i  i  \n
0000440  j  j  j  j  j  j  j  j  j  j  j  j  j  j  j  j
0000460  j  j  j  j  j  j  j  j  j  j  j  j  j  j  j  \n
0000500  k  k  k  k  k  k  k  k  k  k  k  k  k  k  k  k
0000520  k  k  k  k  k  k  k  k  k  k  k  k  k  k  k  \n
0000540  l  l  l  l  l  l  l  l  l  l  l  l  l  l  l  l
0000560  l  l  l  l  l  l  l  l  l  l  l  l  l  l  l  \n
0000600  m  m  m  m  m  m  m  m  m  m  m  m  m  m  m  m
0000620  m  m  m  m  m  m  m  m  m  m  m  m  m  m  m  \n
0000640  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n
0000660  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  \n
0000700  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o
0000720  o  o  o  o  o  o  o  o  o  o  o  o  o  o  o  \n
0000740  p  p  p  p  p  p  p  p  p  p  p  p  p  p  p  p
0000760  p  p  p  p  p  p  p  p  p  p  p  p  p  p  p  \n
0001000  q  q  q  q  q  q  q  q  q  q  q  q  q  q  q  q
0001020  q  q  q  q  q  q  q  q  q  q  q  q  q  q  q  \n
0001040  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r
0001060  r  r  r  r  r  r  r  r  r  r  r  r  r  r  r  \n
0001100  s  s  s  s  s  s  s  s  s  s  s  s  s  s  s  s
```

```
0001120   s   s   s   s   s   s   s   s   s   s   s   s   s   s   s  \n
0001140   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t
0001160   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t  \n
0001200   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u
0001220   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u  \n
0001240   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v
0001260   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v  \n
0001300   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w
0001320   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w  \n
0001340   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x
0001360   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x  \n
0001400   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y
0001420   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y  \n
0001440   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z
0001460   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z  \n
0001500
```

The output file (encrypted data, depends on the encryption level):

```
0000000  270   @ 373 240 305   S   K   | 023   ; 354 373 033   g   M 363
0000020  217 344   9 373 250 201  \0   n 276   | 327   #   ; 340 322 363
0000040  222   \ 307 260 375 301 243 256   N   u 311 362 026 324 357 353
0000060  234 002   d   K 311 233 227 227 335   g   2 264 200   4  \n 022
0000100  237 256   l 340 376 016 263   M   O 266 215 033   _   )   \ 234
0000120    0   q   ~ 276 302   ! 036 003 214   B 242 276   ~ 220   ! 266
0000140    # 370 325   6 345   S 226   5 305 204 376   a 207   `   8 243
0000160  243 006 027 275  \0 024   E 213   }   H  \r   5 033   9 207 231
0000200    U 361 006   - 271   4 277   8   U 024   O   1   g 241   m 356
0000220  274 304 247 277 357   |   x 266   K 272 367 023 350   A   s 342
```

```
0000240    F 006 341 223 370   %   F 023 354 360 262  \f   . 377   =  \f
0000260  254 326 304   0 361 334   b 215 367   J   / 315   0  \n 216   0
0000300    F  \a 271 230   b 333   ; 001   } 257   w   &   [ 314   r   S
0000320    ? 215   w 316   [ 333   9 247 216   A 037 262 350 370   X 242
0000340  237 313 227 361 362 276 271   n   p   / 340   G 266 262   / 205
0000360  375   . 231 276   # 037   J   [   N   } 223 026 216 264 247 245
0000400  275 371 374  \f   R   Y 332   % 217 341   z   w 260   3   R   t
0000420  263   f 206   S 353 362 317 244 200 215 025   ; 307  \t   / 307
0000440  032   v   S 242   + 005   ' 341   = 232 006 330   m   ~   !   d
0000460  336 016 264   2   I 332   F 211   f 346   # 317   0 362  \b 377
0000500    t 216 312   M   [ 305 030   X 252 210 306 230 264 371   H 277
0000520    K 251   Z   V   c  \v   D 177   S   $ 033 200 362   Y 025   0
0000540  240   .  \b 030 016 275   I   0   G   '   , 233   b   8   ? 002
0000560    4 266 342  \b   X 336 231   `   q 216 230 250   /   | 313   A
0000600  265 336   m   R 030   \ 361  \r 332   (   E   [ 360   U 254 365
0000620  375 271 217   r   y   " 275   g   A 343  \t   G 001   & 016   f
0000640    F   3 216 376   v 230 246   ) 227 241   A 331 250 257 374   s
0000660    ^   4   d 306 372 337   * 347 017   / 212   ] 277   *  \a 350
0000700  251   ) 375 254   q   3 373   ] 034 031 243 032   8 341 321 244
0000720    (   n   V   4 237   X 361 334 360 362   y   r 370   K   E   K
0000740    L 330 272   y 251 361 343     252   Y 304   C   4   ` 032   q
0000760  016 357 321   w 350 343 021 006   <   ` 222 021 216 227 210   C
0001000    q   q   q   q   q   q   q   q   q   q   q   q   q   q   q   q
0001020  201 333 344 177 216 031   \   ,   F   C   9 270  \f  \r   X   6
0001040  367   ^   U 344   D 301 360  \a 373   j 022   4 350 345 300 327
0001060    g 256   [ 333 020   d   Y 033   @ 375 317   9   T  \a  \f   k
0001100    s   s   s   s   s   s   s   s   s   s   s   s   s   s   s   s
```

```
0001120    x 216   1   # 226 351 020 036 265   g 207 247 035   @   +   ‘
0001140  320   X   B  \a  \f 231 243 317 252 313 314   u 035 032   e   g
0001160    Y 332 317 201   P 270 214 222   g   J 355 361 202 210 350 370
0001200    u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u
0001220    V 242 204 002 226  \f   {   e 330 271   v   T   J 212 017 272
0001240      246   3   ‘ 020 231 326 006 222 002 263  \f 017  \t 246   o
0001260    A 232 211 233   J   O 273 032 371 346 206 027 216 370 322 364
0001300    w   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w
0001320  027 255 301 033 243   X 305 305   9   8 305 035 204 316   q 230
0001340  032   g   :   N   T   T   9 206 204 373 240 356 220 305   t 316
0001360    ^ 232   r 267   { 016 350 317 216   d   8 275   A   a   u  \r
0001400    y   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y
0001420    I 375  \f 370 220   a   1 037   7 375 243   8 251   3 205   .
0001440  312   S 352 347 036 215 245 016 252   F 024 246   $   / 271   0
0001460  036   W   W 033   { 272   t 334 221   J 321   0 313 362   ; 255
0001500    a   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a
0001520  217 344   9 373 250 201  \0   n 276   | 327   #   ; 340 322 363
0001540  222   \ 307 260 375 301 243 256   N   u 311 362 026 324 357 353
0001560  234 002   d   K 311 233 227 227 335   g   2 264 200   4  \n 022
0001600    c   c   c   c   c   c   c   c   c   c   c   c   c   c   c   c
0001620    0   q   ~ 276 302   ! 036 003 214   B 242 276   ~ 220   ! 266
0001640    # 370 325   6 345   S 226   5 305 204 376   a 207   ‘   8 243
0001660  243 006 027 275  \0 024   E 213   }   H  \r   5 033   9 207 231
0001700    e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e
0001720  274 304 247 277 357   |   x 266   K 272 367 023 350   A   s 342
0001740    F 006 341 223 370   %   F 023 354 360 262  \f   . 377   =  \f
0001760  254 326 304   O 361 334   b 215 367   J   / 315   O  \n 216   O
```

```
0002000   g   g   g   g   g   g   g   g   g   g   g   g   g   g   g   g
0002020   ? 215   w 316   [ 333   9 247 216   A 037 262 350 370   X 242
0002040   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h
0002060 375   . 231 276   # 037   J   [   N   } 223 026 216 264 247 245
0002100   i   i   i   i   i   i   i   i   i   i   i   i   i   i   i   i
0002120 263   f 206   S 353 362 317 244 200 215 025   ; 307  \t   / 307
0002140   j   j   j   j   j   j   j   j   j   j   j   j   j   j   j   j
0002160 336 016 264   2   I 332   F 211   f 346   # 317   0 362  \b 377
0002200   k   k   k   k   k   k   k   k   k   k   k   k   k   k   k   k
0002220   K 251   Z   V   c  \v   D 177   S   $ 033 200 362   Y 025   0
0002240   l   l   l   l   l   l   l   l   l   l   l   l   l   l   l   l
0002260   4 266 342  \b   X 336 231   `   q 216 230 250   /   | 313   A
0002300   m   m   m   m   m   m   m   m   m   m   m   m   m   m   m   m
0002320 375 271 217   r   y   " 275   g   A 343  \t   G 001   & 016   f
0002340   n   n   n   n   n   n   n   n   n   n   n   n   n   n   n   n
0002360   ^   4   d 306 372 337   * 347 017   / 212   ] 277   *  \a 350
0002400   o   o   o   o   o   o   o   o   o   o   o   o   o   o   o   o
0002420   (   n   V   4 237   X 361 334 360 362   y   r 370   K   E   K
0002440   p   p   p   p   p   p   p   p   p   p   p   p   p   p   p   p
0002460 016 357 321   w 350 343 021 006   <   ` 222 021 216 227 210   C
0002500   q   q   q   q   q   q   q   q   q   q   q   q   q   q   q   q
0002520 201 333 344 177 216 031   \   ,   F   C   9 270  \f  \r   X   6
0002540   r   r   r   r   r   r   r   r   r   r   r   r   r   r   r   r
0002560   g 256   [ 333 020   d   Y 033   @ 375 317   9   T  \a  \f   k
0002600   s   s   s   s   s   s   s   s   s   s   s   s   s   s   s   s
0002620   x 216   1   # 226 351 020 036 265   g 207 247 035   @   +   `
0002640   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t
```

```
0002660    Y 332 317 201   P 270 214 222   g   J 355 361 202 210 350 370
0002700    u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u
0002720    V 242 204 002 226  \f   {   e 330 271   v   T   J 212 017 272
0002740    v   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v
0002760    A 232 211 233   J   O 273 032 371 346 206 027 216 370 322 364
0003000    w   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w
0003020    w   w   w   w   w   w   w   w   w   w   w   w   w   w   w  \n
0003040    x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x
0003060    ^ 232   r 267   { 016 350 317 216   d   8 275   A   a   u  \r
0003100    y   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y
0003120    y   y   y   y   y   y   y   y   y   y   y   y   y   y   y  \n
0003140    z   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z
0003160  036   W   W 033   { 272   t 334 221   J 321   0 313 362   ; 255
0003200    a   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a
0003220    a   a   a   a   a   a   a   a   a   a   a   a   a   a   a  \n
0003240    b   b   b   b   b   b   b   b   b   b   b   b   b   b   b   b
0003260  234 002   d   K 311 233 227 227 335   g   2 264 200   4  \n 022
0003300    c   c   c   c   c   c   c   c   c   c   c   c   c   c   c   c
0003320    c   c   c   c   c   c   c   c   c   c   c   c   c   c   c  \n
0003340    d   d   d   d   d   d   d   d   d   d   d   d   d   d   d   d
0003360  243 006 027 275  \0 024   E 213   }   H  \r   5 033   9 207 231
0003400    e   e   e   e   e   e   e   e   e   e   e   e   e   e   e   e
0003420    e   e   e   e   e   e   e   e   e   e   e   e   e   e   e  \n
0003440    f   f   f   f   f   f   f   f   f   f   f   f   f   f   f   f
0003460  254 326 304   O 361 334   b 215 367   J   / 315   O  \n 216   O
0003500    g   g   g   g   g   g   g   g   g   g   g   g   g   g   g   g
0003520    g   g   g   g   g   g   g   g   g   g   g   g   g   g   g  \n
```

```
0003540    h   h   h   h   h   h   h   h   h   h   h   h   h   h   h   h
0003560  375   .  231  276   #  037   J   [   N   }  223  026  216  264  247  245
0003600    i   i   i   i   i   i   i   i   i   i   i   i   i   i   i   i
0003620    i   i   i   i   i   i   i   i   i   i   i   i   i   i   i  \n
0003640    j   j   j   j   j   j   j   j   j   j   j   j   j   j   j   j
0003660  336  016  264   2   I  332   F  211   f  346   #  317   O  362  \b  377
0003700    k   k   k   k   k   k   k   k   k   k   k   k   k   k   k   k
0003720    k   k   k   k   k   k   k   k   k   k   k   k   k   k   k  \n
0003740    l   l   l   l   l   l   l   l   l   l   l   l   l   l   l   l
0003760    4  266  342  \b   X  336  231   ‘   q  216  230  250   /   |  313   A
0004000    m   m   m   m   m   m   m   m   m   m   m   m   m   m   m   m
0004020    m   m   m   m   m   m   m   m   m   m   m   m   m   m   m  \n
0004040    n   n   n   n   n   n   n   n   n   n   n   n   n   n   n   n
0004060    n   n   n   n   n   n   n   n   n   n   n   n   n   n   n  \n
0004100    o   o   o   o   o   o   o   o   o   o   o   o   o   o   o   o
0004120    o   o   o   o   o   o   o   o   o   o   o   o   o   o   o  \n
0004140    p   p   p   p   p   p   p   p   p   p   p   p   p   p   p   p
0004160    p   p   p   p   p   p   p   p   p   p   p   p   p   p   p  \n
0004200    q   q   q   q   q   q   q   q   q   q   q   q   q   q   q   q
0004220    q   q   q   q   q   q   q   q   q   q   q   q   q   q   q  \n
0004240    r   r   r   r   r   r   r   r   r   r   r   r   r   r   r   r
0004260    r   r   r   r   r   r   r   r   r   r   r   r   r   r   r  \n
0004300    s   s   s   s   s   s   s   s   s   s   s   s   s   s   s   s
0004320    s   s   s   s   s   s   s   s   s   s   s   s   s   s   s  \n
0004340    t   t   t   t   t   t   t   t   t   t   t   t   t   t   t   t
0004360    t   t   t   t   t   t   t   t   t   t   t   t   t   t   t  \n
0004400    u   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u
```

```
0004420   u   u   u   u   u   u   u   u   u   u   u   u   u   u   u  \n
0004440   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v
0004460   v   v   v   v   v   v   v   v   v   v   v   v   v   v   v  \n
0004500   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w
0004520   w   w   w   w   w   w   w   w   w   w   w   w   w   w   w  \n
0004540   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x
0004560   x   x   x   x   x   x   x   x   x   x   x   x   x   x   x  \n
0004600   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y
0004620   y   y   y   y   y   y   y   y   y   y   y   y   y   y   y  \n
0004640   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z
0004660   z   z   z   z   z   z   z   z   z   z   z   z   z   z   z  \n
0004700   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a
0004720   a   a   a   a   a   a   a   a   a   a   a   a   a   a   a  \n
0004740   b   b   b   b   b   b   b   b   b   b   b   b   b   b   b   b
0004760   b   b   b   b   b   b   b   b   b   b   b   b   b   b   b  \n
```