Object Synchronisation and Security for Mobile Communication Devices

By

Torbjörn Borison

Royal Institute of Technology
Kungliga Tekniska Högskolan

2001-09-20

Examiner and academic Supervisor: Prof. Gerald Maguire, Department of Teleinformatics, Royal Institute of Technology

Corporate Supervisor: Johan Hedin, Ericsson Radio System AB

Abstract

The main objective of this master's thesis project was to investigate and find solutions to the problem of how to combine the SyncML synchronisation specification with object security and thus protection of personal information, such as contacts and calendar entries in mobile devices.

SyncML is a new synchronisation specification agreed upon by major device developers (Ericsson, Palm, Motorola, etc.) and the major synchronisation server developers (Starfish, Puma, fusionOne, etc.). It is independent of transport (HTTP, WSP, or OBEX) platform, operating system, and application and simplifies synchronisation of personal information between dissimilar SyncML supportive devices.

SyncML compliant devices are fully capable of synchronising information with a third party operated Internet based server and a desktop computer. This allows us to access, up-date and maintain information independent of Intranets or geographical position. However, synchronising and storing confidential personal information on an third party operated Internet based server entails weaknesses in our personal information security. Even if transport and storage security are used, how secure is the server where this information is stored since this server has the highest probability of being attacked. Can we really trust that an employee or other person with valid appropriated administrators access to the storage facility with the appropriate knowledge, working together with the third party server operator, won't try to access our stored information? To prevent this, the personal information's confidentiality must be guaranteed before the information leaves the device.

When synchronising and exchanging personal information, the information is often marked according to a specific format. The three de-facto standard PIM formats are: (1) vCard (contact information), (2) vCalendar, and (3) iCalendar (calendar and scheduling information). These formats divide the personal information into properties. Each property is assigned to contain a small piece of the personal information entry (e.g. a telephone number, an e-mail address, the time when the calendar event begins, etc.).

Furthermore to preserve the interoperability between different devices given by SyncML, authorised recipients must automatically be able to reverse the encryption process and decrypt the encrypted property value. Therefore general cryptographic formats are used (e.g. CMS, PGP and the newly developed XML Encryption). They add information needed by the recipients (e.g. algorithm used, padding method used on the plain text, etc.), encrypt the plain text into cipher text, and decrypt the cipher text into plain text given the correct key.

Acknowledgements

To my parents for all your support during my first 26 years!
Without both of you this thesis would literally never have been possible!

To my corporate supervisor, Mr. Johan Hedin and to
my academic supervisor Professor Gerald Maguire
for your valuable comments, inputs and, suggestions!

To my manager, Mrs. Lori Robertsson and the wonderful staff at
WARP lab, Ericsson Radio System AB, Kista
for supporting me with my master's thesis project!

To my former internship managers, Mr. Henry Ösund, Mr. Lars
Andersson, Mr. Robert Palin, Mr. Michael Eslamian,
and Mrs. Fatemeh Valipour and their staff
thank you for giving me the opportunity of exploring the wireless world!

To my former, wonderful co-workers at
the Mobile Applications Initiative, Berkeley
for all your support and encouragement during my internship last year,
I miss you all!

To our dog Whittie
for being a genuine rascal dragging me out on long walks,
giving me lots of fresh air, and new ideas!

Finally, to all of you, not mentioned here, but in some way have
contributed to this master's thesis!

Contents

1. Introduction 6
- 1.1 Background 6
- 1.2 Purpose 7
- 1.3 Disposition 7

2. Background to object synchronisation and security 8
- 2.1 Synchronisation of Personal information 8
  - *2.1.1 Introduction to SyncML* 12
- 2.2 Cryptographic algorithms 14
  - *2.2.1 Symmetric Algorithms* 15
  - *2.2.2 Asymmetric Algorithms* 16
- 2.3 Object Security 18
- 2.4 Transport Security Protocols 19

3. SyncML 20
- 3.1 Overview of SyncML 20
- 3.2 The SyncML Representation Protocol 22
- 3.3 The SyncML Synchronising Protocol 25

4. Cryptographic formats 31
- 4.1 CMS / S/MIME 32
- 4.2 PGP 34
- 4.3 XML Encryption 35

5. Personal Information management formats 37
- 5.1 Introduction to PIM formats 37
- 5.2 Calendar Formats 39
  - *5.2.1 vCalendar Format Version 1.0* 39
  - *5.2.2 iCalendar 1.0* 51
- 5.3 Contact Formats 52
  - *5.3.1 vCard* 52
- 5.4 Future PIM Formats 54
  - *5.4.1 xCard* 54

6. Solutions 56

7. Conclusions and Future Work 59
- 7.1 Conclusions 59
- 7.2 Future Work 60

8. References 61

Appendix A Abbreviations and Acronyms 65

List Of Figures

2.1 Local and Remote Synchronisation…….. 9
2.2 IrMC 1.1 Synchronisation Session……... 11
2.3 Synchronisation information transmission over multiple transport bindings... 13
2.4 Information exchange using symmetric cryptography…… 15
2.5 Information exchange using asymmetric cryptography……17
2.6 Comparison of security levels of ECC and RSA…….. 18

3.1 SyncML Architecture……21

4.1 Cryptographic Envelope……... 33

1. Introduction

1.1 Background

The development of more powerful mobile communication devices, for example the Ericsson R380, has made it easier to access on-line applications designed for wireless devices provided by application service providers, (ASP). In addition this development has resulted in new possibilities to provide information confidentiality using powerful algorithms before information is transmitted and more easily authenticate which user should have access to which information.

Moreover the available on-line calendar and contact applications offers secure storage and the ability to synchronise the content stored on the Internet with a PDA or other mobile communication devices. To make this synchronisation functionality independent of manufacturer and operating system the major mobile communication device manufacturers (Ericsson, Motorola, Palm, Psion, and Nokia) and synchronising server manufacturers (Starfish, Puma, and fusionOne) have made an effort to agree upon a common synchronisation specification called SyncML [Sync. Whitep.]. Unfortunately, Microsoft has so far chosen to be observer and is not a member of the SyncML initiative.

Unfortunately there are disadvantages with these type of on-line Personal Information Management (PIM) applications; in an article in the newspaper Computerworld with the title “Users eye Yahoo-Starfish offering warily”, Peter Mojica, R & D vice president at First Union National Bank’s capital markets division in Charlotte, N.C says, “Contact information is of corporate value that needs to be protected from prying eyes and is a target for corporate espionage, even if you have a secure transmission, you still have your corporate data sitting on someone else’s server. What’s the deal with that?” [Hamblen].

Encrypting the personal information before it is transmitted is another approach for protecting information, but gives rise to new problems since every field of a PIM database entry must be encrypted independently of others if the added protection is not to interfere with the functionality of synchronising information using SyncML. However, this type of protection is not very well supported by the most frequent used personal information formats vCard [Versit 2], iCalendar [Dawson 1], and vCalendar [Versit 1].

However, there have been discussions how to represent contact and calendar entries in XML. Several drafts have been suggested, but no decision has so far been taken, which one is going to be an IETF RFC. Furthermore the SyncML Initiative has also investigated this issue and so far an early version of a contact format xCard [xCard] is found. Since SyncML is mend to provide interoperability for synchronisation of information between dissimilar devices it is of importance of this M.Sc. project to SyncML supported standards or de-facto standard formats. Despite the fact there is no such XML PIM formats standards, combining XML encoded PIM formats of both contact and calendar formats with the W3C XML Encryption format may be a solution to the problems faced in this M.Sc. project.

1.2 Purpose

The main objective of this thesis project is to investigate and solve the problem of how to add security to protect Personal Information Management (PIM) in combination with synchronisation using SyncML. The central questions that will be analysed are:

- Is it possible to encrypt single fields in independently of each other, marked according to one of the PIM formats iCalendar, vCalendar, and vCard without interfering with the SyncML synchronisation protocol.

- If it proves to be impossible to add encryption of the information in the fields, what are the problems and can they be solved. Are there any other formats that can be used without interfering with the synchronisation process?

- Finally if it is too complicated to add protection to the different fields in an entry or if there aren't any suitable formats that can be used will it be possible to support this type of protection in future formats. What is the ideal solution that requires as little as possible to be changed such that the solution is as attractive as possible?

1.3 Disposition

- Chapter 2 gives a brief introduction of object synchronisation and security.
- Chapter 3 contains a description of the synchronisation protocol SyncML and its functionality.
- Chapter 4 focuses on the cryptographic formats CMS / S/MIME, PGP, and XML Encryption
- Chapter 5 will elucidate the most frequent Personal Information Management (PIM) formats
- Chapter 6 explains my proposed solutions.
- Chapter 7 exams some conclusions resulting from these solutions and information about future work.
- References and Appendix A (which lists abbreviations).

2. Background to object synchronisation and security

2.1 Synchronisation of Personal Information

Since the release of the Palm PDA in 1996 with its build-in connectivity in combination with a serial cable connected to a stationary terminal, synchronisation[1] has become wide spread. Stationary terminals include: laptops, desktops, and servers running different operating systems – from now on we refer to this terminal as "the server". A synchronisation application for PIM ensures that contacts, calendar, and scheduling information, which may be constantly changing is correct and up to date on multiple devices and are identical.

The actual synchronisation of personal information is basically nothing more than a very selective, intelligent data copy between two separate databases that take a number of parameters into consideration before the old information is overwritten with new information. This implies that synchronisation applications for PIM usage between a handheld device and server must be "content – aware" and be able to map and compare different fields in the different database entries.

In the discussion regarding synchronisation of personal information, e-mails, notes, tasks and files between mobile devices, mobile communication devices, and a server -- an agreement must be made according to the format that the information is going to be exchanged in. Often an intermediate format is used, for example the vCalendar format for calendar and scheduling information and the vCard format for contact information. Often the handheld devices only support a limited number of options and information fields and before synchronisation can take place the different information fields in the client's database must be matched against the corresponding field in the server's database. This is called information mapping and once this is done the information in the database on the handheld device can be synchronised with the server.

One of the first pioneering synchronisation applications for powerful PIM personal information up dating of separate devices was the Hotsync. application delivered with the Palm PDA. HotSync. enabled the possibility for PIM, synchronising files, and making a backup copy of the stored information on the PDA. Copying between the PDA and a desktop computer is called local synchronisation. Based on the Hotsync. application, companies like Starfish and PUMA technologies began to develop more advanced PIM synchronising applications for PDA's that rather than just offering local synchronisation also offered remote synchronisation. Remote synchronisation occurs when the PDA communicates with a remote server connected to the Internet or to an Intranet via its own network connection or through a locally connected PC's network connection.

[1] Synchronisation is an abbreviation for the technology of transmitting and copying information for the purpose of having the same set of information on two separate devices.

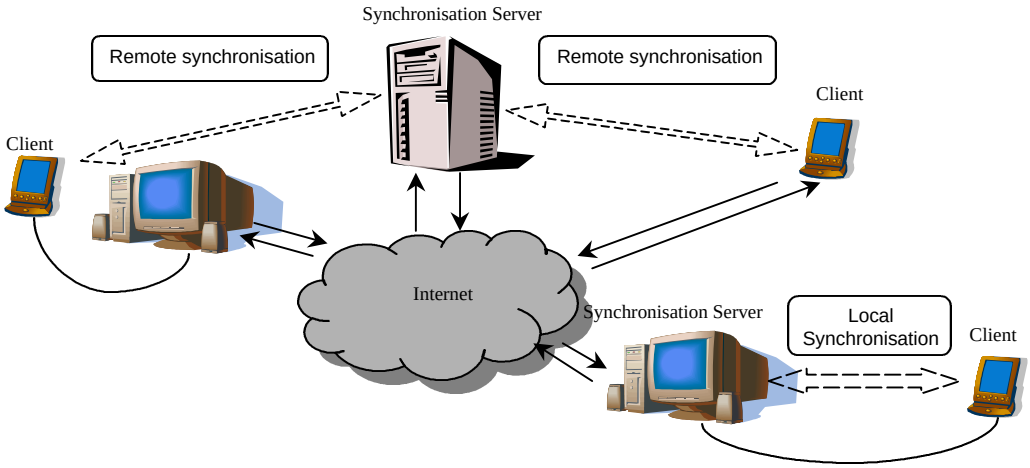


Figure 2.1 Local and Remote Synchronisation

In mobile communication devices such as the Ericsson R520, Ericsson T39, and the Nokia Communicator 9210 it is possible to store calendar info as well as contact information. Moreover these devices can synchronise information, wirelessly with a server located on the Internet without assistance from a stationary terminal and a serial cable.

The problem with Hotsync. was that it only worked with PDA's running the Palm OS and with the development of other operating systems like Microsoft Pocket PC, Symbian Epoc, and mobile phone manufacturer specific OSs like Ericsson's and the Nokia Geo, there was a need for a general, open synchronisation technology. The first general, manufacturer independent synchronisation specification and the predecessor to SyncML are a part of the IrMC 1.1 specification [IrMC]. However, the synchronisation is only a small piece of the IrMC 1.1 specification that contains information about how to exchange information between two devices over the infrared port, which is found on most modern portable terminals.

The general synchronisation protocol found in the IrMC 1.1 standard for local and remote synchronisation consists of 6 parts:

- Device change log
- Synchronisation agent
- Synchronisation client application
- Synchronisation Engine
- Synchronisation server application
- Mapping table

Modern mobile communication devices, from now on called client's, store personal information as separate entries in the corresponding database type, one for each type of personal information, contacts, calendar, notes etc. Similarly the server stores the information received from the client during the synchronisation in the same type of databases. The client stores information regarding changes to database entries as records in a database log, called Change log. The Change log can be both stored on the client, called a Stored log or it can be dynamically created upon request when a synchronisation session is initiated, called Transmitted log [IrMC]. The advantage of a Stored log is that no further computation is necessary by the device to examine which entries have been changed since the last synchronisation. The disadvantage is that the Stored log must be large enough to contain all changes since the last synchronisation, something which is not an issue if a dynamically created Transmitted log is used.

IrMC 1.1 only supports client initiated synchronisation sessions. When the client has initiated a synchronisation session the server based synchronisation engine is started. The synchronisation engine is responsible for managing requests and replies to the client and sees that the correct action is performed on the correct entry.

In addition to the information regarding changes in database entries, the device Change log may also include information regarding the device's serial number, database ID, maximum amount of entries that the database can contain along with the total amount of used entries, information for database authentication and management of available memory space.

A record in the Change log consists in general of the three parameters:

- Action
- Synchronisation anchor (Change counter / Timestamp)
- Locally Unique Identifier (LUID)

The action parameter indicates if an entry in the client's database is added, modified, or deleted. The synchronisation engine uses this information in order to properly manipulate the entry. Upon synchronisation a copy of added and modified entries are requested from the client while deleted entries are removed from the server's database without any further manipulation.

The synchronisation anchor is a parameter that indicates when in terms of order or time a database entry was changed. A synchronisation anchor is either an identification string with a sufficient bit length[2] to be specific, called the Change counter that records a sequence number, or a Timestamp indicating time and date for the change. The information in a synchronisation anchor is extremely useful to the synchronisation engine, when it determines which entries have been changed since the last synchronisation. As a reference the synchronisation engine uses a stored copy of the most recent synchronisation anchor from the last synchronisation. With this information the synchronisation engine can easily determine which records the synchronisation engine should request since these records' synchronisation anchors will have a higher Change counter number or a later Timestamp than the stored reference anchor. This increases the synchronising speed and decreases the time it takes to synchronise the client with the server, since only the modified records since the last synchronisation are transmitted and treated by the synchronisation engine.

The Change counter is a sequence number, which is used to indicate when an alteration of a record has occurred. The number is represented by a long integer value consisting of *n* bits. The oldest alternation, i.e. the first alternation that is made to an entry in the database will receive the highest sequence number, 2^n. Consequently the most recent change will receive the lowest sequence number. If a Stored Change log is used, then *n* must be large enough to contain all changes since the last synchronisation, which is not an issue if a dynamically created log is used. Nevertheless independent of which log is used the client may be synchronised with multiple servers and each server uses its stored reference anchor from the last synchronisation to request only those entries with a lower sequence number from the client, i.e. those records that have been added to the Change log since the last synchronisation.

A Timestamp represents the time and date, when information in a database was changed. The problem is that time can be presented in many different ways and unfortunately there exists no commonly agreed upon standards, only de-facto standards. The most widespread interpretation of Timestamps are the ones present in "common-date" format defined in the International Standard ISO 8601 specifying numeric representations, UTC of date and time.

[2] The IrMC 1.1 specification sets this length to 32 bits

UTC interpretates time and date as “year month day hour minutes seconds”, for example a user, located in London the timestamp, May 6. 1999 3:05:20 PM is translated into the UTC format 19990506T150520Z. The Z at the end stands for zero meridian" and indicates that the time has been specified in Greenwich Mean Time. If Timestamps are used the synchronisation engine determines which entries to request by the Timestamp when the change was made to the database entry.

The last parameter of a Change log record is the Locally Unique Identifier (LUID), which is an identification number for a specific database entry on the client. Since the LUID is only locally unique this identification number might exist on other clients. However an important thing is that the identification number on the **server,** GUID must be globally unique. GUID is an abbreviation for Globally Unique ID and such identification number can never be reused as long as it exists in a mapping table. However, if a database entry in the device is removed from both the client and the server, then the GUID mapped to the LUID of this specific entry could be reused once again. However often the numeric representation of the GUID is long enough that the probability of GUID shortage is low, which makes reuse unnecessary.

The IrMC 1.1 specification only supports client initiated synchronisation. After the client has initiated a synchronisation session and the synchronisation engine is started, the server transmits the stored reference synchronisation anchor from the last synchronisation to the client and requests the portion of the Change log that contains all records since the last synchronisation. Then the client transmits all records that have a lower Change counter sequence number or a newer Timestamp in chronological order with the oldest record first, to the server.

1. The server processes the first record, the oldest, in the received Change log.
2. If it contains information regarding an added or modified record the server requests a copy of the whole database entry from the client and temporarily stores it in a register.

These two steps are repeated until all new records from the Change log have been processed. Then the server accesses the stored copy of the database entries and performs a synchronising analysis.

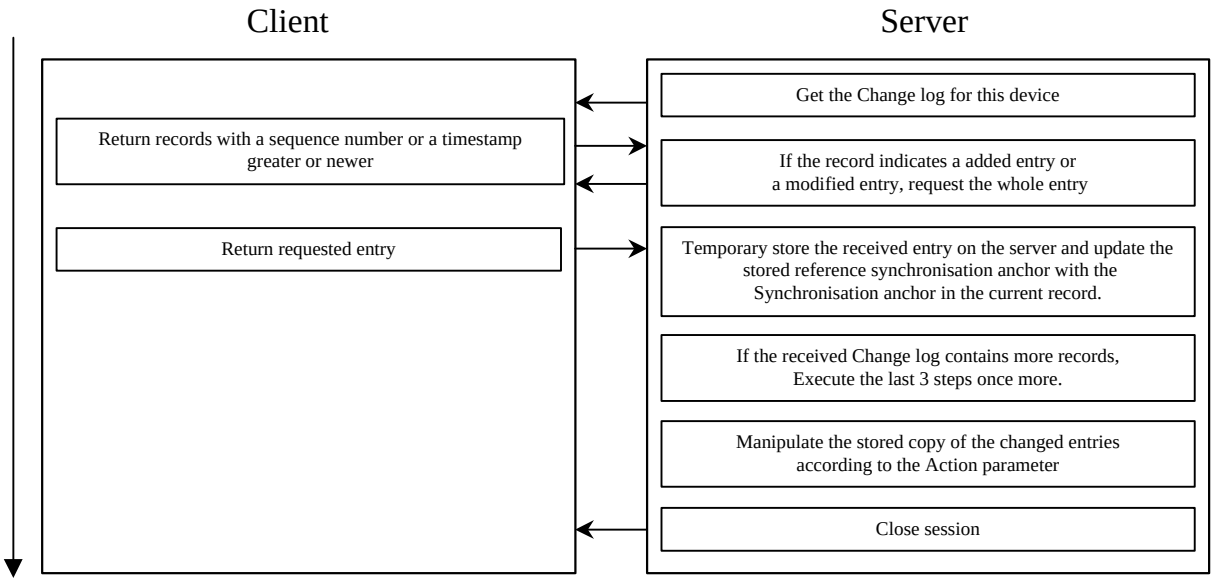


Figure 2.2 IrMC 1.1 Synchronisation session

The LUIDs of the added entries in the client are assigned a mapping to a GUID and added to the mapping table. The changed records are forwarded to the synchronisation application where they are processed and the old information is replaced with the new information.

2.1.1 Introduction to SyncML

One of the features of a mobile communication device is the possibility to access and update information anytime, anywhere. Updating and maintaining information on multiple devices is an assignment well suited for synchronisation, however up to now the only general synchronisation specification is IrMC 1.1, which only supports local synchronisation. For remote synchronisation the users must use manufacturer dependent solutions. This has the disadvantage that they use different network communication protocols, each uses only selected transports, implemented on a few devices from a limited number of manufacturers [Sync. Whitep.]. This proliferation of non-interoperable synchronising technologies restricts the user's ability to access data from any device, anywhere and limits the delivery of mobile data services using synchronisation.

These limitations and device dependencies can be disregarded with a general and wide spread synchronisation technology. Therefore some of the major device and application manufactures founded the SyncML initiative and has developed a new synchronisation specification. The main idea behind this new specification is to make it easier for all parties involved (the users, device manufacturers and application developers) to take full advantage of the mobility offered by mobile communication devices. With such a specification devices are interoperable with a broader range of applications services and transmission technologies are available to the users. The service provider will be able to provide connectivity to a wider selection of applications rather than only one, such as Hotsync. from Palm. Application developers will be able to develop applications that can connect to a more diverse set of devices and networked data.

Hence, the main goal of a common synchronising specification such as SyncML; is to make it possible to:

- Synchronise networked data with any mobile device
- Synchronise any mobile device with any networked data

This enables access and manipulation of data from different devices without loosing important information. For example, a user will be able to change and update a telephone number or entry code in a contact entry on one device and then synchronise it with all his or her SyncML compliant devices. The user now avoids using the old number or entry code, stored on another device by mistake.

Since data traffic from different devices travels over different networks, SyncML supports different session protocols [Sync. Whitep]:

- HTTP for Internet transmissions[3]
- WSP for wireless transmissions, using the WAP standard
- OBEX for short range communications (Bluetooth, IrDA, or a serial cable connection)

[3] Often HTTP uses a TCP connection and can therefore rely on the underlying TCP/IP connectivity. Nevertheless in virtual private networks (VPN), HTTP is the only protocol that is allowed through the company firewall. For this reason, support for HTTP is necessary for synchronising information in all-possible environments.

In addition SyncML also supports [Sync. Whitep]:

- SMTP, POP3, and IMAP
- Pure TCP / IP networks
- Proprietary wireless communication protocols

Compared with the IrMC 1.1 synchronisation described earlier in this chapter the functionality of SyncML has similarities. However, the SyncML specification contains two additional components besides the actual synchronisation protocol, an XML-based representation protocol and transport bindings for the synchronisation protocol. Consequently, the SyncML specification contains three following parts [Sync. Arch.]:

- An XML-based representation protocol
- Transport bindings for the synchronisation protocol
- A synchronisation protocol

The representation protocol specifies an XML Document Type Declaration (DTD) that allows representation of all the information required to perform synchronisation, including data, metadata, and commands.

The synchronisation protocol describes how the client and the server exchange information conforming to the DTD regarding changes in database entries in order to correctly synchronise data. The SyncML synchronisation protocol is based on instructions found in the SyncML representation protocol.

The transport bindings specifies how a specific protocol can be used to exchange messages and responses. Therefore, both the SyncML representation and the synchronisation protocols are independent of transport since every SyncML package is “self-contained” and can theoretically be carried by any transport mechanism. This enables multiple transport protocols between the client and the server. For example OBEX can be used to transfer synchronising information over a Bluetooth connection from a PDA to a mobile communication device, which in turn passes the information further to a WAP gateway over WSP. From the WAP gateway to the synchronisation server located somewhere on the Internet the information is transported over HTTP.

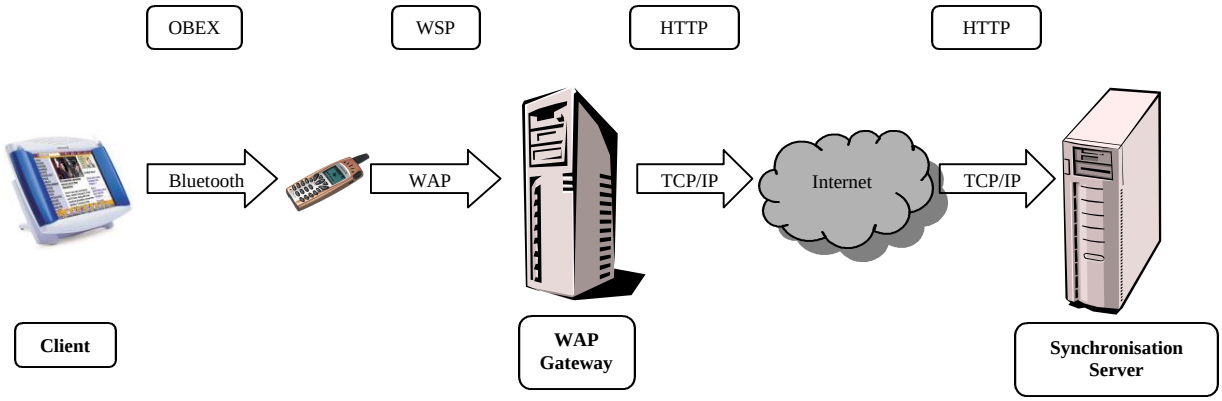


Figure 2.3 Synchronisation information transmission over multiple transport bindings

2.2 Cryptographic Algorithms

There is a growing demand for open networks; therefore protection of information has grown in importance. As recent as only a couple a years ago, cryptographic algorithms developed in the U.S were considered to be of such importance, and in the nation's interest that this technology was treated for export purposes the same as weapons. One of the most important regulations was that only symmetric algorithms with a key length less than or equal to 64 bits could be exported to other countries.

In the discussion of information protection there are basically four different aspects:

- Confidentiality — Confidentiality means protecting information against unauthorised access, using different cryptographic encryption methods. This is implemented by encryption and decryption of data information.
- Authentication — Authentication deals with the question of how users can prove their identity to each other. Knowing the true identity of the receiver prevents transferring sensitive information to the wrong person. For authentication purposes we use special certificates [Borison].
- Data integrity — Data integrity involves protection against unauthorised changes of information. Digital signatures provide data integrity. These prevent an unauthorised person from undetectable altering the protected information.
- Non - repudiation — Non-repudiation prevents a user from denying that they are the author of the protected information. To provide non-repudiation we use digital signatures.

When a user wants to protect information against unauthorised access the best option is to encrypt the information using an algorithm designed for providing information confidentiality. This can be done with either symmetric or asymmetric cryptographic algorithms. The main difference is that a symmetric algorithm uses exactly the same key for encryption and decryption, while asymmetric cryptographic algorithms uses one key for encryption and a different key for decryption. Information that hasn't been encrypted is called plain text. After encryption the information has been transformed to cipher text. A key is a string of bits that are general much shorter than the encrypted information (the plain text) but long enough to be difficult to guess and expensive enough with respect to time and money to discourage an attacker from trying all possible combinations (this is called a brute force attack).

A common notation for communicating users in the cryptographic literature is Alice (A) and Bob (B). The evil user that tries to get hold of and decrypt the information exchanged between Alice and Bob is called Edgar (E).

2.2.1 Symmetric Cryptographic Algorithms

A symmetric cryptographic algorithm uses the same secret key for encryption and decryption of the plain text. The advantage with this cryptographic algorithm design is that the encryption speed is considerable higher, compared with an equal strong asymmetric algorithm offering the same level of security. Symmetric cryptographic algorithms are especially suitable for individual encryption, i.e. when the same user both encrypts and decrypt the given information. The disadvantage of a symmetric algorithm is that the secret key must be shared between the users *before* the receiver can extract the plain text from the cipher. However, if a third person, Edgar is eavesdropping when the key is exchanged, then Edgar gains knowledge of the secret key and thus can encrypt and decrypt all cipher text sent between Alice and Bob, since the same key is used for encryption and decryption.

Symmetric cryptographic algorithms can be divided into two groups, stream ciphers and block ciphers. The difference between them is that stream ciphers operate on one bit or byte at the time, while blocks ciphers operates on a block of bites, often 64 bits.

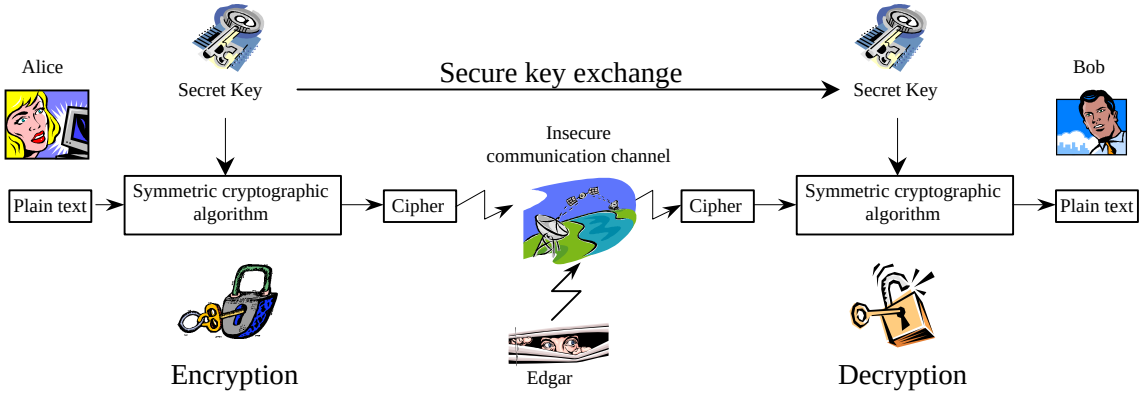


Figure 2.4 Information exchange using symmetric cryptography

2.2.1.1 IDEA

IDEA is an abbreviation for International Data Encryption Algorithm and was developed by Lai and Massey at ETH in Zürich between 1990 and 1992 [Menezes]. The algorithm is a block cipher and operates on blocks with a length of 64 bits [Menezes]. It has been widely adopted thanks to the software library and application Pretty Good Privacy, (PGP) [Schmeck]. The strength of IDEA is based on both the mathematical functions used and a key length of 128 bits. IDEA's designing concept and cryptographic strength mix mathematical operations from three different algebraic groups of 2^n elements to introduce non-linearity. Further IDEA was primary designed for software implementation and not for hardware implementation (such as in an application specific integrated circuit). The security of IDEA currently appears bounded only by the length of the blocks, 64 bits, which gives rise to a weakness compared to its key length. A more thorough description of IDEA can be found in [Menezes].

2.2.1.2 RC5

RC5 was designed by Ronald Rivest and was released in 1994 [RSA FAQ]. RC5 is a block cipher with a variable block size, a variable key size, and a variable number of rounds. The block size can be 16, 32, 64, or 128 bit lengths, although 16 and 32 bits are only for experimentation and evaluation purposes. For real usage a block length of 64 bits or 128 bits must be used to protect the information with a sufficient level of security.

The number of rounds can range from 0 to 255, while the key can range from 0 bits to 2040 bits in size. With these possibilities of varying block length, key size, and the number of encryption rounds provides flexibility to choose the level of security and efficiency. Sometimes the need for a fast algorithm is greater than the need for a high level of security. In this case a shorter key size, fewer rounds, and smaller blocks is used. RC5 consists of three functions: key expansion, encryption, and decryption. In the key-expansion function, the secret key, provided by the user is expanded to fill a key table whose size depends on the number of chosen rounds. The key table is then used in both encryption and decryption. The encryption function consists of three primitive mathematical operations:

1. Integer addition
2. Bit wise XOR
3. Variable rotation

The compact description and simplicity of RC5 makes it suitable for implementation in devices with limited computation and memory -capacity, and is easy to analyse. The security of RC5 depends on the heavy use of data-dependent rotations that makes the cipher resistant to differential and linear cryptanalysis. A through description of RC5 can be found in [Menezes].

2.2.2 Asymmetric Cryptographic Algorithms

Unlike a symmetric cryptographic algorithm, an asymmetric cryptographic algorithm uses two independent keys, one key for encryption, often called the public key and one key for decryption, often called the secret key. The public key is, as the name implies public and can be spread without restrictions to other users. However the secret key, which is also known as the private key, must be kept secret since this key decrypts the cipher text, encrypted with the public key. The main advantage of an asymmetric cryptographic algorithm is that it avoids the hazardous key exchange between a pair of communicating users. When Alice sends sensitive plain text to Bob, she simply encrypts the plain text with Bob’s public key, which she for instance finds on the Internet and sends the cipher text to Bob. The only person that can decrypt the cipher text is Bob since he is the only person that has knowledge of the secret key. The main disadvantage with an asymmetric cryptographic algorithm is that the encryption / decryption speed is significant lower than for an equal symmetric algorithm. Often the asymmetric algorithm is 100 to 1000 percent slower than an equal strong symmetric algorithm.

To take advantage of the encryption / decryption speed of a symmetric algorithm and the simplicity of plain text exchange between a pair of users a method called a cryptographic envelope can be used. Alice, the sender wishes to send sensitive plain text to Bob the receiver. She encrypts the plain text using a symmetric algorithm and sends the cipher text to Bob. She then puts the secret key in a cryptographic envelope, which is nothing more than an encryption of the secret key with Bob’s public key using an asymmetric algorithm. When Bob receives the cryptographic envelope he opens it by using his private key. He has now got the secret symmetric key and can decrypt the earlier received cipher text as plain text.

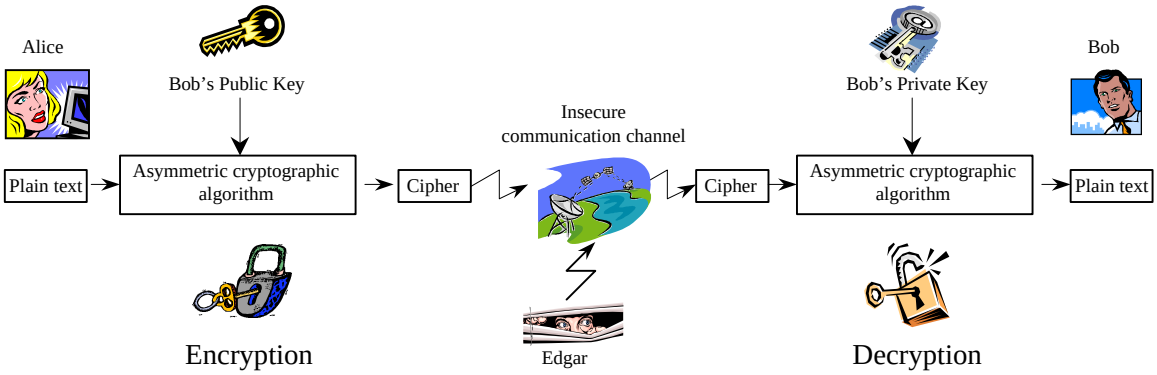


Figure 2.5 Information exchange using asymmetric cryptography

The design of asymmetric algorithms relies upon the known difficulty of solving a specific mathematical problem without knowing some piece of essential information. However, if you know this piece of information the problem becomes very easy to solve. Today there are only three mathematical problems (the factorisation problem, the discrete logarithm problem, and the elliptic curve discrete logarithm problem) that are considered to be hard enough to compute that algorithms based on these three problems are considered to secure. A thorough description of these problems can be found in [Carrara] and [Menezes].

The encryption and decryption speed of an asymmetric cryptographic algorithm depends on the key length. The fastest asymmetric algorithm is the Elliptic Curve Cryptosystem that only needs a key of length 160 bits to achieve the same level of security as RSA using a 1024 bit key length. This advantage over RSA makes the algorithm very well suited for small devices with limited CPU power, memory and battery capacity. The advantage with RSA compared with ECC is that RSA is older and has been thoroughly investigated and tested. Moreover it is used in more applications and has had a greater distribution than the younger ECC.

2.2.2.1 RSA

RSA is an abbreviation for the initials of its inventors: Rivest, Shamir and Adleman and was presented in 1977 [RSA FAQ]. The security of RSA is based upon the mathematical problem of factor large prime numbers. Factor a number that are composed by two large primes is extremely difficult even with a very powerful computer and the costs for cryptanalysis are very high. However, the discovery of an easy method of factor such a number into two primes would break RSA. Fortunately such method is not known to exist and despite intensive research such a method hasn't been found (or at least is not widely known).

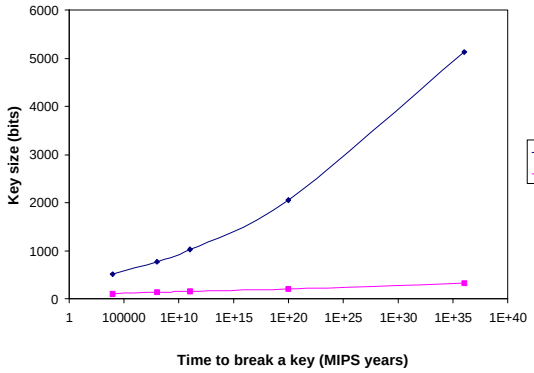
There are actually only three known ways to break a RSA cipher:

1. Steal or somehow gain knowledge of the secret key
2. To somehow factor the public key from a cipher and calculate the secret key.
3. Build a trapdoor into the hardware that generates the RSA modulus $n = pq$ in such a way that the hardware manufacturer can easily factor n, but factoring n remains difficult for all other parties. However, such a trapdoor is easily detected [Menezes].

RSA can be used for both encryption and for creation of digital signatures and has been implemented in the Ericsson R380. The algorithm is very easy to understand and the steps can be written on the backside of an envelope. A more thorough description of RSA can be found in [Menezes].

2.2.2.2 Elliptic Curve Cryptosystem (ECC)

Neil Koblitz and Victor Miller both presented this algorithm in 1985, independently of each other and hence are the inventors of ECC [Certicom]. The ECC algorithm is based on the third mathematical problem – it depends upon the difficulty of calculating the discrete logarithm problem on an elliptic curve. This problem is regarded as more complex and more difficult to compute than the integer factor problem used in RSA. Therefore ECC can use a shorter key and only requires a 160 bit key to achieve the same level of security as RSA with a key length of 1024 bit. The shorter key length is as also the major advantage of ECC since a shorter key increases encryption speed and reduces the time it takes to encrypt the plain text into a cipher text. Another advantage of ECC compared with RSA is that the cryptographic strength of ECC increases more rapid with an increased key length than RSA does. A complete description of the elliptic curve discrete logarithm problem can be found in [Mattila] and [Menezes].



RSA Key Size	ECC Key size	Time to Break a key (in MIPS years)
512	106	10^4
768	132	10^8
1024	160	10^{12}
2048	210	10^{20}
5120	320	10^{36}

Figure 2.6 Comparison of security levels of ECC and RSA [Certicom]

2.3 Object Security

Object security is the most widespread application of information security. It embraces techniques for creating a lasting protection over time of stored electronic information objects such as text, images, sound -files, email, etc. Thus the goal of object security is to preserve the confidentiality of data stored in data objects. With the rapidly increasing amount of sensitive and private information stored on different servers on the Internet, the danger that unauthorised persons will access this information has increased dramatically. There is now increased awareness of the importance of preventing information theft and unauthorised access of sensitive information, therefore there is increased interest in object security. This is often implemented by applications, for example the widely spread and used cryptographic application PGP, which implements versions of different cryptographic algorithms (IDEA, RSA, and RC5 among others). Such an application then functions as a filter, between storage media and the user; it can encrypt and decrypt information automatically or on demand by the user. If the storage media is exposed to a unauthorised intrusion, lost, stolen, the user and the user's organisation don't have to worry about the fact that the information can "fall into the wrong hands" if an algorithm with a suitable level of security has been used.

Compared with transport security, which must only protect the information while in transit, the protection of an object must persist until the user wishes to access it. This means that the scope of the protection lives beyond the transmission and hence a protected object, theoretically can be stored on a third parties storage facility **without** compromising its confidentiality.

2.4 Transport Security Protocols

A transport security protocol provides confidentiality; data integrity, and authentication for information exchanged between a client and a server from the session layer and above in the OSI model and in the WAP stack. This transport security can be visualised as a tunnel with an entrance at the client and an exit at the server. This tunnel provides information confidentiality and transparency to the higher layers. This means that the functionality of the protocols and applications running above the transport layer are not affected by the added security protocol and only minor changes needs to be made for them to take advantage of the added protection. Transport security protocols are used when sensitive information is transmitted over an insecure network. For example, they should be used during the transmission of personal information between the PIM client and the PIM server when the data passes over the Internet.

The most frequent used protocol for securing plain text transmissions for wired usage is the Secure Socket Layer (SSL), developed by Netscape and released in 1994. The development was then taken over by IETF and the name of the protocol was changed to Transport Layer Security (TLS) [Linder]. However TLS is merely SSL version 3.1, while SSLv3, developed by Netscape uses SSL version 3.0. Worth noting is that SSLv3 and TLS only supports TCP and not UDP. For wireless usage the WAP forum developed the optional Wireless Transport Layer Security (WTLS) protocol that operates below the WDP and WTP protocols. WTLS is based upon TLS with the modification that WTLS also supports data gram transportation; this has lead to security flaws in the protocol [Saarinen].

Worth noting is that WTLS only supports connectionless transmissions, however this is not a problem since connection orientated transmissions are handled by the WTP protocol that resides above the WTLS layer.

The design of the SSL / TLS and the WTLS protocol are in general similar. They are composed of two layers, the record layer protocol and the three higher-level protocols: the handshake protocol, the change cipher spec protocol, and the alert protocol.

The record protocol operates on top of some reliable transport protocol, e.g. TCP for SSL / TLS and WDP or UDP for WTLS. It encapsulates the higher-level protocols, such as the Handshake protocol and the transmitted data. Further, the handshake protocol allows the server and the client to authenticate each other and establish a session. The change cipher protocol is used for choosing cryptographic algorithm and key exchange. The confidentiality of the information is guaranteed by use of symmetric cryptography. Sharing the symmetric key is described in the change cipher spec. Finally the alert specification is used to discover errors and to guarantee the reliability of the connection by checking the message integrity using a secure hash function.

3. SyncML

3.1 Overview of SyncML

This general, open data synchronisation specification is intended to be a standard for information update and maintenance, interoperable between multiple devices, different platforms and applications running on different operating systems (Palm, Epoc, etc.). The SyncML synchronisation specification consists of the SyncML Representation protocol, the SyncML Sync Protocol, the three transport bindings: (1) SyncML HTTP binding, (2) SyncML over WSP binding, and (3) the SyncML OBEX binding and Document Type Declarations (DTD), the Meta information DTD and the Device information DTD according to the XML mark-up language.

To simplify and describe how the different parts in the SyncML specification interact with each other, a conceptual model called the SyncML framework can be created. The SyncML framework basically consists of two parts: (1) the SyncML I/F and (2) the SyncML adapter. A third part, the SyncML Agent is outside the SyncML framework and generally cannot be described since it is manufacturer dependent, but its functionality is important for both understanding and designing SyncML specification conformant solutions. Both the client and the server have their own, platform and manufacturer specific agent. The SyncML agent is responsible for the communication between the device's application and the SyncML framework and for invocating data synchronisation operations, creating SyncML objects (messages) used for communicating both synchronisation instructions and information to the SyncML adapter, through the SyncML I/F. The SyncML I/F is an Application Program Interface (API), between the SyncML agent and the SyncML adapter. The SyncML I/F, i.e. the instructions that the SyncML agent can use to create messages, and how to mark-up the synchronisation information is described in the SyncML Representation protocol [Sync. Rep.]. The SyncML adapter is also responsible for interfacing with the different network transport protocols (OBEX, WAP, HTTP) between the server and the client, creating and maintaining a connection between the server and the client and transmitting and receiving SyncML messages.

In reality the server and client implementations are seldom implemented in these discrete framework components.

The Synchronisation protocol describes how to use the SyncML Representation protocol to create SyncML messages with instructions for initiation, synchronisation, and termination of a SyncML synchronisation session. To accomplish this, a set of common attributes and requirements [Sync. Protocol] were developed to support the seven different synchronisation types. There are different types of synchronisation procedures depending on whether the client or the server initiates the synchronisation and whether all the information in a database is transmitted, a so-called slow sync or only the modified database entries, fast sync [Sync. Protocol.].

The problem with synchronising information between different devices is that these devices often have different capabilities. Therefore the SyncML specification contains a Device Information DTD that is used for exchanging device capabilities between the server and the client. Device capabilities are information, concerning supported PIM formats, software versions, and other device specific parameters, which must be negotiated before the synchronisation of information can be initiated. In addition some smaller devices i.e. mobile communication devices often only supports parts of these PIM formats to save storage space. Therefore before information can be synchronised, i.e. during the initiation of the absolutely first synchronisation session between the communicating devices, information regarding device capabilities formatted according to the SyncML device information DTD must be exchanged to prevent interoperability problems.

This information is then used by the Sync Engine, which sends only the required amount of data information that is useful for the client and thereby saves bandwidth.

The framework and the interaction between the client and the server in SyncML are shown in the figure below:

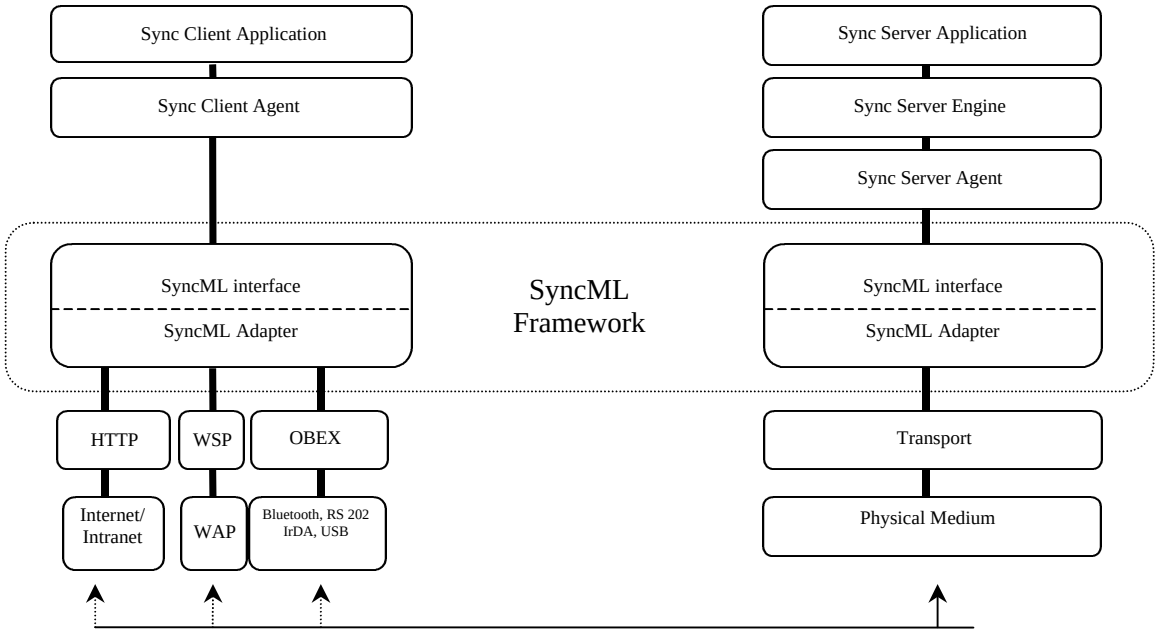


Figure 3.1 SyncML Architecture

The SyncML Meta-information DTD description, describes how the different meta elements shall be used, for example the two sync anchors `<Last>` and `<Next>`, the `<Format>` element specifying the encoding format and the `<Type>` element that specifies the media type –of the content information in the `<Data>` element. Considering the objective in this project, elements not mentioned above described [Meta Inf.], don't contribute to this document. These additional elements and their description are found in [Meta Inf.].

Using SyncML to synchronise encrypted information is well supported with the option to specify both the media type and the encoding of the submitted data information between the `<Data>` and the `</Data>` tag. There are actually only two elements that might cause problems if confidentiality is added to the synchronised data information, i.e. to parts of the database entries. These elements are the Protocol Command Element `<Search>` and a combination of the two Common Use Elements `<LocUri>` and `<Target>` used for Database Addressing. Both these elements are involved in search instructions to a database on the server. The problem is that searching or filtering a database consisting of encrypted information with a plain text string will never be successful since there exists no string in the database that will match the query string.

Searching an encrypted database using a plain text string is a problem that has received attention by different research groups around the world. One of these is the ISAAC[4] research group, at the University of California at Berkeley.

[4] Internet Security, Applications, Authentication and Cryptography (ISAAC) [http://www.isaac.cs.berkeley.edu/]

This group has developed a technique that was presented at the 2000 IEEE symposium on Security and Privacy in Oakland, California, U.S [http://www.bell-labs.com/user/reiter/sp2000/]. The technique uses a plain text string as input and returns a list of possible values ranked according to the probability that the plain text string occurs in each encrypted value [Xiaodong].

3.2 The SyncML Representation Protocol

The Representation Protocol describes a XML language based document mark-up format that specifies and describes a set of elements, intended to function both as synchronisation instructions used in the synchronisation session and as a container for the synchronised data information. The Representation protocol is used as a template how to assimilate and produce individual, well formed, but not necessarily valid SyncML messages. Because SyncML supports different transport protocols the size of these messages are only limited by the limits of the transport protocol used (OBEX, WSP, HTTP) [Sync. Rep.].

SyncML messages can be sent both as plain text XML and / or encoded in a tokenised binary format defined by the WAP Binary XML content format specification (WBXML) [Sync. Rep.]. For the purpose of transporting SyncML messages with different transport and session protocols supporting Multimedia Mail Extensions (MIME) the content types *"application/vnd.syncml+xml"* and *"application/vnd.syncml.wbxml"* have been registered at the IANA organisation for plain text respectively binary SyncML messages [Sync. Rep.].

The format specifies a set of elements, divided in five types depending on their functionality, which are used to divide and mark-up the synchronisation instructions and the data information into easy parsable and human readable parts.

- Message Container Elements
- Protocol Command Elements
- Protocol Management Elements
- Common Use Elements
- Data Description Elements

The Message Container Elements consists of three base elements, the `<SyncML>` element and its two children container element, the `<SyncHdr>` and the `<SyncBody>`. The `<SyncML>` element indicates the beginning and end of a SyncML message and has a conceptual function as container for the `<SyncHdr>` & `<SyncBody>` elements. The other two Message Container Elements functions as a container for header information, `<SyncHdr>` and as a container for body information, `<SyncBody>.`

Both Data Description Elements and Common Use Elements can occur within both the `<SyncHdr>` and the `<SyncBody>` element while Protocol Command Elements and Protocol Management Elements only can occur within the `<SyncBody>` container element according to the [Sync. Rep.].

3.2.1 SyncHdr

The SyncML Message Container element `<SyncHdr>` specifies a container for header information such as revisioning and routing information. The `<SyncHdr>` element encapsulates ten children elements from the two types Common Use Elements and Data Description Elements. Six of these ten elements are mandatory and must be supported by SyncML compliant devices and applications.

These six mandatory elements are:

VerDTD
VerProto
SessionID
MsgID
Target
Source

The additional four optional elements are:

RespURI
NoResp
Cred
Meta

The first two elements are used for specifying the version identifier of the Representation Protocol used to represent the SyncML message <VerDTD> and the version identifier used with the SyncML message <VerProto>.

The <SessionID> and <MsgID> element are used for identification of the current session <SessionID> and for the current SyncML message <MsgID>.

The two elements <Target> and <Source> can have different meanings depending upon within which element they are used. When found within the <SyncHdr> tags they specify source routing information for the networked device that originated the data synchronisation request <Source> and source routing information for the networked device that is receiving the information <Target>.

Since none of these elements in the header has any influence on the objectives of this project they won't be described any further. For those interested in a thorough description regarding their functionality and usage see chapter 5 in [Sync. Rep.].

For the same reason none of the Common Use Elements are described in this thesis except for one, the <LocUri> element, since this element might cause problems if used for Database Addressing.

3.2.2 SyncBody

The SyncML message container body <SyncBody> element specifies a container, encapsulating elements used for both synchronisation instructions and the actual information to be synchronised [Sync. Rep.]. All the Protocol Management Elements and almost all Protocol Command Elements are children of the <SyncBody> container element. These are used to mark-up the instructions that the recipient shall execute on the submitted data in combination with suitable Common Use Elements. The actual data is marked-up with the Data Description Elements also in combination with the Meta Data Description Elements.

The Protocol Management and the Protocol Command –Elements can be found below, a detailed description of them can be found in [Sync. Rep].

The Protocol Management and the Protocol Command Elements are:

Add

Alert

Atomic

Copy

Delete

Exec

Get

Put

Replace

Search

Sequence

Sync

Results

Fortunately only the `<Search>` element and the Common Use Element `<LocUri>` (as mentioned earlier) are likely to cause problems if the information is encrypted, and therefore they will be more thoroughly described.

3.2.2.1 Search

The `<Search>` Protocol Management Element enables the originator to send a search request to the recipient's databases with a query string for information in a specific entry or distinguished field of an entry in a database. This is an optional element and need not be supported by SyncML compatible devices.

The problem is if the client submits a plain text PIM format property value as the search string to the search request to a database, consisting of encrypted entries. Since the server where the database is located is considered as distrusted it has no knowledge of the secret key and therefore can't decrypt the stored information and hence the server will therefore always reply with "not found" since the searched string doesn't exists.

Fortunately as mentioned in the beginning of this chapter the problem of how to search an encrypted database with a "plain text" search string has received attention in different research groups around the world. One solution can be found in [Xiaodong].

3.2.2.2 LocUri

The `<LocUri>` Common use element combined with CGI scripting can be used for target address filtering. This search method is considered to be easier to implement than the more robust `<Search>` element and can be used to filter or restrict the number of selected database entries before synchronisation.
The SyncML Representation Protocol has defined a special set of CGI script tags for the vCard contacts format and the iCalendar calendar and scheduling formats and its predecessor vCalendar.

The <LocUri> Common use element specifies a target or source specific absolute or relative URI or a well-known URN address. This parameter can be a child of either the <Source> or the <Target> parent element. These two elements are themselves, children of the Protocol Management Elements <Item>, <Map>, <MapItem>, <Search>, <Sync>, and <SyncHdr>.

Fortunately neither of these two elements, <Search> and <LocUri> are likely to cause any functionality problems and affect the performance of SyncML specification compliant applications.

3.3 The SyncML Synchronisation Protocol

The Synchronisation protocol describes how to initiate, synchronise, and terminate a synchronisation session and exchange synchronisation data using SyncML messages, created from the SyncML Representation Protocol. To accomplish this seven different synchronisation types are specified together with a set of common features and requirements [Sync. Protocol] supporting all seven synchronisation types. The synchronisation types differ from each other based on whether the client or the server initiates the synchronisation session and if all information or only the modified entries in a database are transmitted.

The set of common features and requirements supporting the different synchronisation types are:

- Change Log information
- Sync Anchors
- ID Mapping of Data Items
- Conflict Resolution
- Security
- Addressing
- Exchange of device capabilities
- Sync without Separate Initialisation
- Device Memory Management
- Multiple Messages in Package
- Busy Signalling

3.3.1 Change Log Information

Similar to the synchronisation procedure described in the IrMC 1.1 specification, SyncML compliant terminals must use a database Change log on both the server and the client to be able to trace changes that have occurred since the last synchronisation in a specific personal information (calendar, contact, etc.) database. This makes it easier to distinguish, up-date, and exchange the information that has been modified since the last synchronisation.

However the SyncML Sync Protocol doesn’t specify the format in which the information is stored in the Change log, but it demands that when a synchronisation session is initiated the client and the server must be able to specify which entries have been modified, i.e. replaced, added, and soft or hard deleted in the database since the last synchronisation session. Using unique ids (LUID and GUID) together with the type of modification (e.g. the different elements Replace, Add, Delete) does this.

The SyncML Synchronisation protocol specification allows synchronisation between multiple devices. This extension gives rise to new possibilities and additionally increases the requirements on the Change log. The Change log in SyncML compliant devices must be able to keep track of all changes related to previous synchronisations sessions with each device.

3.3.2 Sync Anchors

A Sync Anchor is a parameter that indicates the last successful synchronisation session and makes it easier to distinguish the records in the different database's Change log that has been modified since the last successful synchronisation session. A Sync Anchor can be either a monotonically increasing number transferred as an integer string indicating how many successful synchronisation sessions that has been made since the first successful synchronisation session or a UTC based ISO 8601 time and date stamp indicating the last time a successful synchronisation session was performed.

The SyncML Sync Protocol, version 1.0 uses two Sync Anchors, `Last` and `Next`. The anchors are interpreted in the SyncML specification as children elements to the `<Meta>` element. The `<Meta>` element and all its children can be found in the SyncML Meta-information DTD, version 1.0. The `<Last>` element value specifies the last event (time) the client / server was successfully synchronised with the server / client and the `<Next>` element value specifies the current synchronisation session [Sync. Protocol] according to the originator. Moreover both the server and the client exchange the `<Last>` and `<Next>` Sync Anchor with each other during the initiation of a synchronising session.

Since multiple devices can be synchronised with each other, the `<Next>` Sync Anchor on the synchronised clients (devices) must be stored until the next synchronisation session in order for the devices to maximum utilize the anchors functionality [Sync. Protocol].

3.3.3 ID Mapping of Data Items

SyncML supports the possibility for both the client and the server to have their own identifier format for entries in their databases. This is a desired feature since often the identifiers on small clients such as mobile communication devices have less memory than full size terminals (laptops, desktops or servers) hence the IDs are much smaller than on such terminals. Considering these ids size there is a conflict with the condition that the ids must be "unique forever" [IrMC 1.1]. Lower the requirement of the device's id of being only locally unique solves this conflict. However, the database entry id on the server must be big enough to be unique forever, i.e. globally[5] unique (GUID) from all other database ids. Since the identifier format for the same database entry on both the server and the client might be different the server must contain a so-called mapping table connecting the entry identifiers on the client with the entry identifiers on the server. If the mapping table were located on the client, i.e. the mobile communication device it would soon be obsolete, when another client updated the server. In addition if the client simply used the server's considerably longer Globally Unique ID GUID no memory gain would be made.

3.3.4 Conflict Resolution

The Sync Protocol also discusses the issue of how to resolve conflicts, called Conflict resolution. A conflict arises when the Sync Engine can't decide which information from

[5] Globally unique means that the identifier, i.e. the character sequence are of such length that the probability that the server will randomly choose the same identifier for another entry is close to zero.

which set of data to up-date or exchange. This can happen when the same database entry on both the server and the client was modified simultaneously and independently of each other. However, the Sync Protocol doesn't specify hints how these conflicts should be solved and leaves this issue to the Sync Engine developers. The different PIM formats vCalendar, iCalendar, and vCard provide some help by providing mark-up of information, indicating when the last change was made to a database entry, how many changes have been made to the same database entry since it was created, and the date when the database entry was created. Besides the support from these PIM interoperability formats the SyncML Representation Protocol defines a set of status codes that shall be used by the Sync Engine to notify the client of how the conflict was solved.

3.3.5 Security

Considerations have been made in the SyncML specification concerning authentication of clients and servers and transmission security during the transmission between the originator and the recipient. For authentication purposes, the SyncML specification contains both elements and a procedure for both the server and the client to challenge the other party. By authenticating themselves both the server and the client can ensure themselves of the true identity of the other party and prevent an impostor from pretend to be either one of them.

Authentication can either be done by the originator of the synchronisation session, who includes its credentials during the initiation using the common element `<Cred>` in the `<Alert>` and `<Sync>` element as well as using the `<Status>` element. The other possibility is if the response code to a request sent without credentials is 401 (Unauthorised) or 407 (Authentication required). The response must them contain the `<Status>` element, which must include the `<Chal>` child element. The `<Chal>` element value contains challenge applicable to the requested resource [Sync. Rep.], [Sync. Protocol].

SyncML supports both basic authentication and the more complex MD5 digest authentication. Furthermore SyncML conformant devices must support both these authentication procedures [Sync. Protocol].

To protect the information during transmission SyncML relies upon transport security services provided by lower transport layers such as WTLS, TLS, or SSL.

As mentioned in the beginning of this chapter the security features offered by the SyncML specification neither influence nor affect the SyncML specification as to the of this project encryption of single database entry field values.

3.3.6 Addressing

The SyncML Sync protocol, version 1.0, specifies three different types of addressing:

- Device and Service Addressing
- Database Addressing
- Addressing of Data Items

Neither Device and Service Addressing or Addressing of Data Items uses information found in the PIM databases (contact, calendar, etc.) entry field values. Since these entry field values are the information that are intended to be secured by using encryption algorithms, the functionality of both these addressing types are not affected whether the values are in plain text or not.

However as mentioned earlier the information found in these database entry field values can be used by the Database Addressing method that uses the `<LocUri>` element in combination with some of the PIM CGI script tags found in the [Sync. Rep.]. Unfortunately this type of addressing is difficult if the fields in these entries are encrypted. The reason is because addressing must use the same cipher-string that is stored in the database on the server. This implies that the device must either store the information in encrypted form or it must encrypt the plain text database entry field values using exactly the same algorithm and key that was used to create the cipher entry values, stored in the server database when a Database Address request is created.

3.3.7 Exchange of Device Capabilities

During the initialisation phase in the first synchronisation session between a client and a server the client must send its capabilities to the server prevents the server from sending information that is not supported by the client and therefore useless. By not sending unsupported information, the amount of information transmitted and therefore the bandwidth used is minimised. Moreover the device capabilities are considered to be static since this type of information is seldom changed. Obviously if there is a change in them, for example due to a software upgrade, the new device capabilities must be exchanged between the server and the client. For an average user, changes in the device's capabilities are not a frequent event. Therefore the device capabilities is not likely to be transmitted more than one time, during the initialisation of the first synchronisation session. This minimises the amount of information transmitted, lowers the workload of the wireless networks and lowers the costs for the end consumers for each subsequent synchronisation session.

3.3.8 Sync without Separate Initialisation

The synchronisation protocol offers the possibility to start both the initialisation and the operation phase of a session simultaneously. This will lower the amount of information transmitted, since fewer packages will be sent, but it will also lowers the security. The problem is that the server can't send it's credentials back to the client before it starts sending data and therefore the client cannot be certain that it is communicating with the correct server before any information is exchanged. This makes it possible to set up a fake server and gain access to the innocent user's personal information. However, using the methods of this thesis, to encrypt the personal information before it is transmitted from the mobile communication device, can prevent this. Besides the security aspects, considerations must be given to robustness since the client sends its modifications before the synchronisation anchors are exchanged. If the server or the client requests a slow-sync all the information exchanged before the synchronisation anchors were received must be retransmitted [Sync. Protocol].

3.3.9 Device Memory Management, Multiple Messages in a Package, Busy Signalling

These last three features and requirements are not influenced by whether the synchronised information is encrypted or not, nor do they contribute to the understanding of SyncML, or raise/lower the security for the synchronised information or the personal privacy. Therefore I have chosen not to describe them any further, a description can be found in [Sync. Protocol].

3.3.10 The seven Supported Sync Types

The above common attributes and requirements support seven different synchronisation types, which differ from each other depending on whether the server or the client initiates the synchronisation and whether both the server or the client sends it's changes to the other party or if information is only transmitted in one direction a one way synchronisation session: from the client to the server or only from the server to the client.

- Slow sync
- Two-way Sync
- One-way Sync From Client Only
- Refresh Sync From Client Only
- One-way Sync From Server Only
- Refresh Sync From Server Only
- Server Alerted Sync

3.3.10.1 Slow Sync

Slow sync is a synchronisation type where the two sets of data are compared with each other. Practically, this means that the client sends its whole database to the server, which performs the synchronisation analysis. This type of synchronisation is always used the first time a new client is synchronised with a server or when the client and the server are badly out of sync (e.g. when the sync. anchors doesn't match).

3.3.10.2 Two-way Sync

This is the normal and the most common type of synchronisation procedure in which only the modifications, found in the database's Change log are exchanged and up-dated on both the server and the client. Independent of whether the server or the client was the originator, i.e. initiated the synchronisation session, the client always sends its modifications first.

3.3.10.3 One-way Sync From Client Only

When this type of synchronisation is used the client sends all its modifications, as per the database Change log, to the server where these entries are synchronised, but the server doesn't send its modifications back to the client.

3.3.10.4 Refresh Sync From Client Only

This is a synchronisation where the client sends all information, i.e., both modified and unmodified entries, in a database to the server (i.e. exports all information) and the server replaces the entries in the target database with the received information.

3.3.10.5 One-way Sync From Server Only

When this type of synchronisation is used the server sends all its modifications, as per the Change log, to the client where these entries are synchronised, but the client doesn't send its modifications back to the server.

3.3.10.6 Refresh Sync From Server Only

This is a synchronisation where the server sends all information, i.e., both modified and unmodified entries, in a database to the client (i.e. exports all information) and the client replaces the entries in the target database with the received information.

3.3.10.7 Server Alerted Sync

This type of synchronisation can only be initiated from the server and instructs the client to initiate one of the synchronisation types specified above.

4. Cryptographic formats

Cryptographic formats are specifications how to develop interoperable applications for protecting, i.e. encrypting and decrypting confidential information exchanged between two communicating items (users and devices) during transmission, temporary, and permanent storage. This type of format is mainly used when confidential information needs more permanent end-to-end protection than provided by transport security formats (SSL [SSL 3.0], TLS [Allen], and WTLS [WTLS]). Cryptographic formats are primary used to protect e-mails and data files stored on some storage media.

Hence these formats provide both protection and interoperability between multiple devices. These are two important mechanisms for solving the problem addressed in this thesis. Combining information confidentiality concerning personal information with the synchronising language SyncML, without limiting the interoperability provided by the PIM formats and by SyncML. Interoperability is important since the same personal information can be synchronised with other devices[6] or with servers[7] located somewhere on the Internet. Furthermore it is important that each device is capable of automatically detecting if parts of the synchronised information are encrypted and which cryptographic format was used. The cryptographic format used is easily detected if the cipher text is encoded according to a valid MIME content type registered with IANA [http://www.iana.org/]. Furthermore if the recipient is to be able to automatically decrypt the cipher additional information and instructions are needed concerning, e.g. the type of algorithm (asymmetric or symmetric) used, which algorithm has been used (RSA, ECC, DES, 3DES, IDEA, etc.), if the cipher is digitally signed, if the confidential information has been composed as a cryptographic envelope, etc. The two most well known and widely spread cryptographic formats are Cryptographic Message Syntax (CMS) and Pretty Good Privacy (PGP). Besides these two formats, the recently developed XML Encryption cryptographic format is also of interest since it will make it possible to protect XML documents, XML entities and XML entity contents and binary information while encoding it in XML.

When encoding personal information according to one of the PIM formats vCard, vCalendar, and iCalendar, the contacts and calendar entries are divided into smaller parts called properties. Often, only a few of these properties in a contact or calendar entry are changed and need to be updated during a synchronising session. Therefore each property value must be encrypted independently of each other to preserve the synchronisation functionality provided by SyncML. Moreover these property values can be treated as very small messages. Because both CMS / S/MIME and PGP also include the necessary instructions and information needed to decrypt the cipher, the size of each property value in bytes will be considerable larger, depending upon the used cryptographic format, algorithm, etc. then the plain text property value. Consequently the total transmitted amount of data will be larger and thus it is even more important that only the updated parts of a contact or calendar entry are exchanged during synchronisation.

Despite that the protected contact or calendar object being bigger than the unprotected object, protecting personal information is important. Because when the information has been delivered to it's final destination or temporary destination[8] the transport security protection is removed, if the information is not protected, it would be vulnerable to a possible attack.

---

[6] Local Synchronisation.
[7] Remote Synchronisation.
[8] Some sort of temporary storage facility, for example an e-mail server.

If the personal information is synchronised with a server located on the Internet, operated by a third party, the server operator should some sort of storage protection that protects the personal information while it is stored on its server. Obviously the operator must have access to the encryption and decryption keys to this secure storage to be able to retrieve the information upon request from the owner or another "authorised" user. Unfortunately the access to the decryption key also makes it theoretically possible for someone working for or with the server operator with the appropriate knowledge to decrypt and access the information. Hence even with storage protection the server should still not be considered as trusted. Moreover since some information, often-corporate data might be of interest to some people [Hamblen], using a cryptographic format lets the owner of the information take control over the security. Finally, since a general cryptographic format has been used, other applications on other devices such as a home computer, a mobile communication device, a PDA, etc. can synchronise with the server and automatically, without assistance or prior knowledge, decrypt the cipher.

4.1 CMS and S/MIME

CMS is an abbreviation for Cryptographic Message Syntax and specifies how to implement applications, producing an interoperable output between separate devices for protection of confidential information. CMS is based on PKCS#7, which was developed by the U.S. company RSA Security. CMS is backwards compatible with PKCS#7. Unfortunately changes were necessary to accommodate attribute certificate transfer and key agreement techniques for key management [Housley].

A CMS compliant application is capable of both encrypting the information into cipher text as well as adding necessary information and instructions needed by other applications to decrypt the cipher text into plain text. To reduce the number of possible cryptographic algorithms available to encrypt plain text into cipher text only a few selected algorithms, the majority were developed by RSA Security are required to be supported by CMS compliant applications. This ensures the recipient uses algorithms supported by the originator. Examples of these cryptographic algorithms are: the signature and public key algorithm RSA, the message digest algorithm SHA-1, and the symmetric cryptographic algorithm 3DES [Housley].

CMS specifies six different types of cryptographic content that are possible to compose with a CMS compliant application. These are: data, signed-data, enveloped-data, digested-data, encrypted-data, and authenticated-data. Of these six content types the data, signed-data, and enveloped-data must be supported and implemented by CMS compliant applications, while the other three are optional.

The cipher texts generated by CMS are represented as octet strings (8 bit characters). It is known that such binary strings might not be reliably transported by many e-mail systems. This is however a transport problem and is not dealt with in CMS [Housely]. Nor does the CMS specification elucidate how applications should handle certificates or how to mark the output as a valid, registered MIME type. To both provide reliable transport of CMS cipher text as well as handling of certificates, an additional specification called S/MIME [Ramsdell 2] was developed.

Cryptographic certificates are used to substantiate that a certain public-key belongs to a specific user. CMS and S/MIME makes heavy use of Public Key Infrastructure (PKI) [Borison] and demand that every user possess and use so called X .509 "set of public keys" with a corresponding X.509 certificate, issued by a certified Certificate Authority (CA).

Furthermore S/MIME makes it possible to compose CMS output (cipher text, additional decryption information, and instructions) as a valid IANA registered MIME content type. For this purpose three different MIME types were registered at IANA. These three MIME content types are: Application/pkcs#7-mime (signedData, envelopedData) with file extension *.p7m, Application/pkcs#7-mime (degenerate signedData “cert.-only” message) with file extension *.p7c, and Application/pkcs#7-signature with file extension *.p7s [Ramsdell 2].

The main advantage of CMS is the support for cryptographic envelopes. As mentioned earlier, cryptographic envelopes combine the high encryption speed of symmetric algorithms with the easy key sharing that comes from using asymmetric algorithms. Moreover cryptographic envelopes simplify the procedure of sending the same cipher text to multiple recipients. This may be the case when the same calendar information is shared among several users.

The calendar information (M) is encrypted with a symmetric algorithm (E(M), where E is the key). The symmetric algorithm used key is then encrypted with each recipient’s public key ($P_x(E)$). Consequently a cryptographic envelope consists of the symmetrically encrypted message (E(M)) and a number of encrypted, symmetric keys ($P_x(E)$), one per recipient.

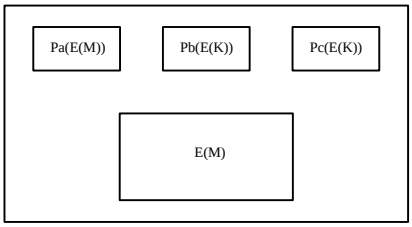


Figure 4.1 Cryptographic Envelope

To retrieve the plain text (M), the symmetric key is decrypted with the recipient’s private key. The cipher text (E(M)) is then decrypted using the retrieved symmetric key. Since the cryptographic envelope may contain several versions of the symmetric key, each recipient’s version is distinguished with the recipient’s e-mail address [Housley]. The e-mail address is used by the receiving user’s application to select the correct encrypted key, designated for the recipient. Using the recipient’s e-mail address as a recognition parameter is a weakness if CMS is to be used to solve the problems addressed in this project. There is no point in encrypting the calendar entry field containing the participants e-mail address using a cryptographic envelope and then use the e-mail addresses in plain text as a recognition parameter in the cryptographic envelope containing the encrypted e-mail addresses!

A sample message of an enveloped CMS and S/MIME message would be:

```
Content-Type: application/pkcs7-mime; smime-type=signed-data;
name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

The composition of a CMS output as an S/MIME content type encoded message varies depending upon the CMS / S/MIME content-type problems.

4.2 PGP

PGP (Pretty Good Privacy) [Atkins], [Callas] was originally developed by Phillip Zimmerman to provide secure exchange and storage of confidential information. Similar to CMS, PGP specifies how to develop PGP compliant applications, providing information protection to exchanged messages and data files. PGP supports digital signatures, encryption, compression, RADIX64 conversion, key management, and handling of certificates. Unlike CMS, PGP allows users to create and use their own "set of public keys" with a corresponding certificate called web-of-trust certificates, besides allowing the use of X.509 certificates.

PGP utilises cryptographic algorithms developed in Europe, e.g. the symmetric cryptographic algorithm IDEA. IDEA uses a key with 128 bits length making the cipher text stronger than, the available U.S developed symmetric algorithms at the time. Earlier[9] when cryptographic algorithms, developed within the U.S. were to be exported to foreign countries[10] the length of the key in bits was restricted to a maximum of 64 bits. This restriction made the cipher text weaker and easier to be exposed by a brute force attack (i.e. decrypting the cipher text without prior knowledge of the encryption key used). In addition, PGP supports a much wider selection of algorithms such as ECC described in chapter 2.2 and the new Advanced Encryption Standard (AES) also known as Rijndeal [Mattila].

A web-of-trust certificate makes it both easier and possible for any person to create a "set of public keys" with the corresponding certificate without involving one of the available Certificate Authorities. Obviously such a web-of-trust certificate hasn't the same trustworthiness as a X.509 certificate, but they allow close friends and colleagues easily exchanging confidential information excluding a hazardous key exchange. Such persons wouldn't normally go through the process of applying and paying for a X.509 "set of public keys" with a certificate valid for protecting their information exchanged. To increase the trustworthiness of a web-of-trust certificate a user can have other users sign his or her certificate guaranteeing that the user who presents the certificate really is who he or she claims to be.

The cipher text and the enclosed information (needed for decryption) are composed as binary packages consisting of a header and a body; of which are both encoded as octet strings. The header consists of specific binary code strings, each string containing information and instructions such as algorithm used, the content of the body, etc. Consequently the information can only be recovered using a PGP compliant application capable of decoding the binary information in the header and using it to decrypt the cipher in the body. Similar to CMS, binary encoded information isn't reliably transported in some environments (e.g. e-mail systems). For this purpose an encoding scheme called RADIX64 conversion is used. RADIX64 expresses the binary data as a base64 format, identical to the MIME base64 content-transfer-encoding (RFC 2231). RADIX64 conversion, converts the binary data to characters in the RADIX64 alphabet consisting of ordinary alphabetic characters [Elkins 2]. Apart from making it possible to reliably transport PGP content in such environments, RADIX64 may also be used in environments requiring information, encoded as ordinary text such as the vCalendar, iCalendar, and vCard PIM format described in chapter 5.

PGP is well thought-out with regard to exchange of information, but just as CMS it has no support for composing PGP packages as MIME messages.

[9] This restriction is now removed. In addition, the new AES (Advanced Encryption Standard), originally called Rijndeal, was developed in Europe. This symmetric cryptographic algorithm may use keys with a length of 256 bits.
[10] This restriction didn't count when U.S. companies or citizens used these cryptographic algorithms.

Because the PGP specification contains all, necessary mechanisms an additional specification similar to S/MIME was not necessary. Instead this issue is solved using MIME multipart encrypted and multipart signed [Galvin] in combination with PGP [Elkins 2]. Furthermore this solution was also chosen to avoid the inability of recovering signed message bodies, without parsing data structures specific to PGP. A multipart encrypted MIME message consists of two parts: One containing plain text information and the second containing the cipher text.

When a multipart encrypted MIME message is used in combination with PGP the first part contains the necessary decryption information in text and the second part contains the binary PGP package. MIME security with PGP defines three MIME content types: Application/pgp-encrypted, Application/pgp-signature, and Application/pgp-keys
[Elkins 2].

Despite the considerably better algorithm support and the greater flexibility of PGP, CMS has gained the most attention in many applications. The reasons is that earlier U.S. based application developers couldn't incorporate PGP in their products, because of the export regulations prohibiting export of applications supporting symmetric cryptographic algorithms using a key greater than 64 bits. Furthermore PGP only supports cryptographic envelopes for one recipient unlike CMS supporting cryptographic envelopes sent to multiple recipients.

4.3 XML Encryption

XML Encryption [Eastlake] is a recently developed cryptographic format for protecting information and encoding cipher text including related information in XML. The format only exists as a "work in progress" document and may be updated, replaced, or obsoleted by other documents at any time. The latest working draft is based on the 15-December-Proposal and was released June 26, 2001 and was a result of a meeting on March 01, 2001 and subsequent discussion on the XML Encryption e-mail list.

XML Encryption may be used on a whole XML document, XML entity, XML entity content, or on arbitrary binary data. What makes this cryptographic format especially interesting is the possibility to encrypt the content of single entities, independent of other entities. This can't be done with either CMS or PGP, which operates on whole messages or documents. Furthermore XML has grown in importance for representing networked data formats and for encoding documents exchanged between different terminals. The SyncML initiative has looked into the possibility of using the XML mark-up language to develop a contact, calendar, and schedule XML representation, founded on vCard, vCalendar, and iCalendar. If such a XML PIM format is used the information (telephone number, e-mail address, surname, starting time for a calendar event, etc.) are encoded as single entities. Therefore it would theoretically be possible to use XML Encryption in combination with such an XML based PIM format without sacrificing interoperability, i.e. the primary goal of this thesis.

XML Encryption supports both symmetric cryptographic algorithms (AES and Triple DES) and asymmetric cryptographic algorithms (also referred to as key transport algorithms such as RSA) [Eastlake]. However, unlike both CMS and PGP, XML Encryption isn't capable of creating or verifying electronic signatures or handling certificates, only protecting i.e. encrypting and decrypting information. Signature and certificate handling mechanisms are provided by another format fully compatible with XML Encryption called XML Signature [Eastlake 2]. Combining XML Encryption with XML Signature provides both message digest and message authentication functionality and together they found a complete security solution for protection of XML documents. Unfortunately the current draft of XML Encryption doesn't support cryptographic envelopes to multiple recipients, which makes it much more complex to synchronise XML Encrypted information with multiple recipients.

Using XML Encryption on XML marked information is merely a simple transform operation. The resulting cipher text is either included within the original XML document encoded as a base64 octet sequence or if the cipher text is located outside the document an URI reference reveals the location where the cipher text can be found. The result of the transform operation is the XML Encryption `<EncryptedData>` entity replaces the original entity or entity content. The `<EncryptedData>` contains both the cipher text and related information, needed for decryption of the cipher text into plain text. An example is shown below[11]:

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Now the credit card number is encrypted, using XML Encryption and replaced with the `<EncryptedData>` containing the cipher and related information (in this case the used algorithm triple DES (3DES)).

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <EncryptionMethod Algorithm='http://www.w3.org/2001/04/
                          xmlenc#3des-cbc '/>
        <CipherData><CipherValue>A23B45C56TfcveidshrA</CipherValue></CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

If the encrypted information is located externally, the `<CipherValue>` entity is replaced with the `<CipherReference>` entity containing the URI of the location where the cipher text is located.

Since XML Encryption only exists as a working draft and may be changed I have chosen not to explain it any further. For those interested the latest version is found at http://www.w3.org/TR/xmlenc-core/.

[11] The example below is taken from the working draft (June 26, 2001) XML Encryption Syntax and processing chapter 2.1.2.

5. Personal Information Management Formats

5.1 Introduction to PIM formats

Formats for exchanging personal information more commonly known as PIM (Personal Information Management) formats convert and marks-up information found in personal information database entries as plain text. Plain text information, i.e. ASCII characters, is an information format that is both easily understood and interoperable between dissimilar platforms, operating systems, and applications. Therefore information represented as plain text can easily be exchanged and shared among different devices independently of application, platform, and operating system as long as the applications support the same PIM data. Similar to SyncML, these PIM formats only define the coding and decoding of personal information found in PIM database entries, but does not define how to implement the processes related to them.

The three most frequent used formats for expressing personal information as plain text are the two calendar formats (vCalendar and its successor iCalendar) and the vCard contact format. These formats use so called properties and special property parameters to mark-up the contacts and calendar database entries and describe specific characteristics found in these entries. A property string has the following grammar:

Property name [';' property parameters '=' property parameter value] ':' Property value

In addition, to simplify the understanding of the exchanged information, the property values often have preformatted value types (a URL address, an e-mail address, time, etc.). For example a property value describing time is formatted according to the international standard ISO 8601[12]. Unfortunately these property value types are also the main problem since encrypting the property values and replacing the plain text information with its cipher value is impossible without violating the strict defined property value types.

Despite the fact that it is impossible to combine confidentiality with parsability of these three PIM formats they are important due to their wide acceptance when exchanging PIM information between different devices, from different vendors. In addition, the SyncML specification encourages clients and servers to use these formats when exchanging personal information. The SyncML Device Information DTD enables clients to inform the server, both which formats and which properties in each format the client (the device) support [Device Inf.]. Exchanging this information prevents the server from sending unsupported information to the client and thereby save bandwidth as noted in previous chapter. Furthermore these formats are already wide spread and supported by many mobile communication devices (e.g. Ericsson R380, Ericsson R520, Ericsson T39, etc.).

It is impossible to find a generally supported solution of how to add confidentiality to these PIM formats since encrypting these property values will interfere with the type formatting rules of these formats and their property values. The best solution would be to modify the PIM formats in such way that that securing only parts of the information in a PIM format object would be possible and at the same time make it possible to parse these PIM objects. Further it must be possible to mark that the information in a property value is encrypted. Such functionality can for example be provided by a global property parameter, for example called "ENCRYPTED" who's property value can be either a PGP or a CMS MIME type.

[12] ISO 8601 is an international standard for a numeric representation of information regarding date and time of the day [ISO 8601].

Though even if it was possible to encrypt single property values, they would probably be considerably larger than their “plain text” representation and considerations must be made regarding the additional benefit to the personal privacy by encrypting a specific property value. In some cases encrypting the property value wouldn’t contribute at all to raise the personal privacy and will most likely be both a waste of mobile communication device resources (such as CPU usage and reduced battery capacity) and network resources. The cost in reduced resources and money for exchanging the larger cipher text in bytes instead of the smaller plain text is most likely to be larger than the contribution to the personal privacy of doing it.

To better illuminate why the property values to these three PIM formats are impossible to encrypt and why some property values do not need to be encrypted, I have chosen to explain and analyse the simpler vCalendar format and its different properties and property parameters. Which property values are worth encrypting or not varies of course from person to person. Therefore the division of properties for the vCalendar format made in this chapter regarding which property values that is worth encrypting or not should be considered more as a suggestion than a proposed solution. The same type of analysis can easily be made for both the iCalendar format and the vCard format, but since giving all three PIM formats a thorough description was not in the scope of this project I have chosen to describe the vCalendar case.

5.1.1 Existing Formats

The two formats vCard and vCalendar were developed by the multivendor initiative “Versit”, founded by Apple Computer Inc., AT&T Corp., International Business Machine (IBM) Corp., and Siemens AG. The Versit Initiative’s vision was to enable interoperability for electronic contact, calendar, and scheduling information in all environments independent of platform, operating system, or application. Shortly after the release of the vCard format version 2.1 [Versit 2] and the vCalendar format version 1.0 [Versit 1] in 1996, the Versit initiative transferred their proprietary rights, development, and marketing responsibilities to the Internet Mail Consortium (IMC).

The iCalendar format is more of a successor to the vCalendar format than a completely new calendar and schedule format standard and was developed by the calendar and scheduling workgroup (Calsch WG) within the Internet Engineering Task Force (IETF)[13]. The iCalendar format version 1.0 is defined in the RFC 2445 [Dawson 1] and it is heavily based on the vCalendar format, but is more powerful and complex than its much simpler predecessor.

5.1.2 Future Formats

Marking up information with preformatted properties can be done using many different languages. One of the most wide spread languages is HTML, but thanks to it’s inflexibility a new language was released a few years ago called XML. XML is a very flexible language and is used when interoperable solutions between different applications, platforms, and operating systems are desired, such as: SyncML and the Wireless Mark-up Language (WML). WML is part of the WAP standard and XML has grown in importance for representing networked data formats and for encoding documents exchanged between different terminals.

13 Internet Engineering Task Force (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet [http://www.ietf.org/].

How to represent contact and calendar and scheduling information using XML has been discussed during the past three years[14]. Unfortunately, no decision has so far been taken as to which draft that shall be an IETF RFC. The issue how to represent in this case calendar and scheduling information in XML is still being debated within the IETF Calsch WG[15]. Besides XML, discussion has begun in a new mail list (www-rdf-calendar@w3.org) within the W3 organisation of how to interpret calendar and scheduling information based on the iCalendar format in Resource Description Format (RDF).

The SyncML initiative has also considered the possibility of using the XML mark-up language to map these PIM formats into XML. So far an XML based contacts format corresponding to the vCard format called xCard [xCard] has been discussed within the SyncML initiative. Since there are many different proposals and drafts concerning how mapping PIM formats into XML, I have chosen focus only to suggestions discussed within the SyncML initiative and so far these discussion has only comprised the XML contacts format xCard.

It is also possible to use the same XML parser to code and decode different formats independent of each other. For example, a XML parser in a mobile communication device is able to parse both SyncML and other XML based formats; unlike a vCard format parser that is only able to parse vCard objects. In addition XML can be tokenised using a binary representation; such as defined by the wireless binary XML format (WBXML) developed by the WAP forum and implemented in WAP enabled devices. Using WBXML makes it possible to reduce the amount of data transmitted over any mobile communication network and thereby saving bandwidth.

Unfortunately this far no document regarding a calendar and scheduling representation in XML has neither been discussed, nor released by the SyncML initiative and only a short working draft of the SyncML xCard format exists. Nevertheless despite all uncertainties and questions my personal opinion is that using a XML representation of the three PIM formats described later in this chapter in combination with the W3C XML Encryption security format will most likely enable **both** parseability and confidentiality of PIM information.

5.2 Calendar Formats

The two calendar and scheduling formats vCalendar and iCalendar are used to code and decode information found in PIM database entries as plain text. This makes it possible to exchange PIM information between dissimilar applications, running on different platforms with different operating systems. Which format to use when coding the PIM information is in the scope of SyncML, up to the device and is decided during the initiation phase of a synchronisation session when the SyncML compliant device sends it's SyncML Device Information DTD to the server.

5.2.1 vCalendar Format Version 1.0

The vCalendar format is much simpler and the objects created according to this format are considerably smaller than the equivalent iCalendar object. Furthermore iCalendar format contains features and functions that are largely unnecessary for use in combination with a mobile communication device and are basically designated for calendar and scheduling servers such as the Microsoft Exchange and the Lotus Domino.

[14] I have based this date on a draft to the IETF organisation from Frank Dawson. "How to represent the vCard version 3.0 in XML" [Dawson 3], dated 22 June 1998.
[15] As recent as the 21 of May 2001 a contribution to the mail list ietf.-calendar@imc.org was made from David King with subject "iCalendar and XML".

Such functionality includes the iCalendar components [Dawson 1]: vFreebusy, iTIP [Silverberg], iMIP [Dawson 3] and CAP [Mansour]. This simplicity makes the vCalendar format more suitable for mobile communication devices since it limits the amount of information transmitted and thereby reduces the bandwidth used. Furthermore the vCalendar format specification is very simple and produces minimal calendar and scheduling object coders and decoders compared with the much more complex and extensive iCalendar format specification. Since today's mobile communication devices still have a limited amount of memory, implementing small applications using minimum memory is still important.

As briefly mentioned before, a vCalendar object is composed of a number of different properties that are used to mark-up the information found in a calendar database entry. In addition to these properties the vCalendar format contains a set of additional property parameters, used to describe different characteristics (attributes) and gives a better description of the type, encoding format, media type, etc. of the information in the property value. In general all property values formatted as text that doesn't uses pre-defined property values can more or less be directly replaced with their encrypted representation.

Formatted property values, such as time according to ISO 8601, or basically any type of format other than text, are basically impossible to encrypt without violating the vCalendar format. However, property values formatted according to ISO 8601 distinguish themselves since they don't generally contain any information worth encrypting. This will be shown later in this chapter. It might also be a disadvantage to encrypt some of these property values since they functions as timestamps when changes has been made to the vCalendar object. These timestamp properties are used during the synchronisation session [Sync. Protocol]. If other property values containing confidential information are encrypted the electronic calendar can be made accessible to other trusted authorised users, this functionality was not supported by the older vCalendar format. However, it enables these other users to find a time for a meeting, appointment, or another event. Since the sensitive and confidential information is encrypted, other persons will not understand the content of these entries, but will know that the time is reserved.

5.2.1.1 VCalendar Container

Similar to the SyncML format the beginning and end of a vCalendar object is marked by the properties "**BEGIN**" and "**END**" followed by the property value "*VCALENDAR*", i.e. "**BEGIN** : *VCALENDAR*" and "**END** : *VCALENDAR*". These two properties form a container, enclosing a plain text representation of a calendar database entry.

Depending upon the type of calendar information in the database entry the information is divided into two components (sub containers), vEvents and vTodos. If the database entry consists of information that occupies a specific amount of time such as meetings, appointments, travel time, etc. it is considered to be a vEvent or if it is active information that needs action such as ordering tickets, write e-mails, write expense reports, etc. it is a vTodo. Similar to the container encapsulating the calendar information, both vEvents and vTodos begins with the "**BEGIN**" property with the type of sub container as property value, i.e. "**BEGIN :** *VEVENT*" or "**BEGIN :** *VTODO*". In the same way that the container is terminated with the appearance of the "**END**" property combined with the type of sub container as its property value, "**END :** *VEVENT*" or "**END :** *VTODO*". Neither a vEvent nor a vTodo object can be nested within each other or within an object of the same type. However, both vEvent and vTodo objects can be related to another vEvent or vTodo object. This object can in turn be related to another vEvent or vTodo object, etc.

Fortunately neither none of the "**BEGIN**" and "**END**" properties needs to be encrypted as both of them are needed for a vCalendar format parser to distinguish the different vCalendar objects from each other.

5.2.1.2 VCalendar Properties

All vCalendar properties are found within the boundaries of a vCalendar object, i.e. after the string "**BEGIN** : *VCALENDAR*" and before the string "**END** : *VCALENDAR*".

The initiation of a vTodo or vEvent object is indicated with the string "**BEGIN** :" directly followed by either "*VTODO*" or "*VEVENT*". Naturally each of these sub objects are closed by the "**END** : " + *component type* as described above, depending upon the type of calendar information in the database entry. Besides the vEvent and vTodo specific properties there are a set of global properties that must appear before the first vEvent or vTodo object and they applies to the whole vCalendar object. These global property parameters can be redefined within the scope of a single vEvent of vTodo object except for the "**VERSION**" property.

5.2.1.3 VCalendar Property Parameters

The vCalendar format defines a set of property parameters that can be used together with vEvent and vTodo specific properties. These property parameters describe different attributes connected with the property value. However, only the "**ENCODING**" property parameter is useful in the scope of this project.

Binary Values and Encoding
Some selected properties are allowed to contain inline binary content. Since binary encoded information violates the default "7 Bit" encoding it must be using a property parameter called "**ENCODING**". The valid "**ENCODING**" property parameter values are: "*BASE64*", "*QUOTED-PRINTABLE*", or "*8-bit*".

Value Location
VCalender allows information to be stored externally instead of the inline default location, often used for binary information. If the information is placed somewhere else, e.g. on the Internet or as a part of a MIME message a property parameter called "**VALUE**" must be used. The property parameter "**VALUE**" is used to override the default inline location of information and to notify the recipient that the information is not found in the property value but in either a MIME message or somewhere on the Internet. If the information is located in a separate MIME entry, the property parameter value is then the Content ID "**VALUE** = *CID*" followed by the MIME entry sequence number. If the information is located somewhere on the Internet, the property parameter value is "**VALUE** = *URL*" followed by the Internet address to the information, conforming to RFC 1738[16]. The "**VALUE**" property may be used together with any other vCalendar property [Versit 1].

Character Set
The default character set is ASCII, but this type can be overridden with the "**CHARSET**" parameter, followed by any character set registered with IANA [http://www.iana.org/].

Language
The default language is US English (en-US), but can be overridden with the "**LANGUAGE**" parameter. The "**LANGUAGE**" parameter can be set to values consistent with RFC 1766[17].

[16] RFC 1738 specifies a Uniform Resource Locator (URL), the syntax and semantics of formalized information for location and access of resources via the Internet [Berners-Lee].
[17] The RFC 1766 describes a language tag for use in cases where it is desired to indicate the language used in an information object [Alvestrand].

5.2.1.4 Globally Defined vCalendar Properties Worth Encrypting

Geographic Position
A vCalendar object can include the geographical position of the "home" system that created the vCalendar object. The property name is "**GEO**" and the property value is the "home" systems position in longitude and latitude according to the prime meridian and the equator. Since this property is optional there is no need for vCalendar format conformant devices to support it. However, if it is used, then this property value might be important to protect since the owner of the calendar application creating, the object might not want to give out his or her geographical position.

Time Zone
The world is divided into time zones depending on the geographical position according to the Zero Meridian. This is supported by the vCalendar format and the property name is "**TZ**". The property value is the time offset in hours and minutes according to Greenwich Mean Time (UTC[18]) and is formatted as "hh" or "hh:mm" according to ISO 8601. This property value does reveal information about where in the world the calendar user is located and though the position is rough, this property value might be considered worth encrypting for the same reason that applies for the geographical position, "**GEO**", property.

5.2.1.5 Globally Defined vCalendar Properties With No Effect On The Calendar Owner's Personal Privacy

In my opinion encrypting any of these property values have no positive effect on the calendar entity's protection and will be both unnecessary and a waste of valuable resources (e.g. CPU power, battery capacity, and bandwidth).

Version
The "**VERSION**" property defines the version number according to the vCalendar version from which the vCalendar object was created. This mandatory property can't be redefined within neither a vEvent nor vTodo –object.

Product identifier
The developer of an electronic calendar application can use the property "**PRODID**" as an identifier to identify the product that created the vCalendar object. This identification number must be globally unique, using some technique such as an ISO 9070 FPI value [Versit 1].

Daylight Savings Rule
In some countries the time is adjusted back or forward according to seasonal changes in spring and autumn. This functionality is naturally automated in an electronic calendar and has the property name "**DAYLIGHT**". It consists mainly of three parts: a "*TRUE*" or "*FALSE*" parameter indicating whether daylight savings is observed (*TRUE*) or not (*False*) followed by the time offset from UTC, the time and date when the adjustment takes place and ends with the date when the adjustment ceases. The last three parts are both formatted according to ISO 8601. The string is formatted as follows:

```
DAYLIGHT:TRUE;-06;19960407T025959;19961027T010000
```

[18] UTC stands for Coordinated Universal Time.

5.2.1.6 VEvent and vTodo Properties

The following properties are found only within vEvent or vTodo objects and can be divided into three categories depending upon the type of property value format. These three properties require values formatted according to:

- The earlier mentioned ISO 8601 international standard.
- As text.
- Either a URN, URL formatted according to RFC 1738, or an e-mail address according to RFC 822[19].

As mentioned earlier in this chapter, properties with values conforming to either ISO 8601, RFC 822, or RFC 1738 or belong to a set of pre-defined property values are impossible to encrypt while preserved parseability according to the vCalendar format specification. Nevertheless these three types can be divided into two subgroups, mandatory and optional properties. A mandatory property must be implemented and supported, while an optional property can be left out by applications, conforming to the vCalendar format.

Properties with formatted date and time values according to ISO 8601

Date and Time issues
VCalendar uses the international standard ISO 8601 to describe time and date property values as well as for time duration, indicating the amount of time that a vEvents occupies in a users calendar. To avoid time zone ambiguity UTC should be used whenever possible where a time value is requested [Versit 1].

A basic representation of time and date according to ISO 8601 can be represented as “year month day T hour minute second”, for example the 6th of May 1975 2 AM is written as 19750506T020000. Since this format is well defined, property values using this format representation are impossible to encrypt and at the same time fulfil the criteria that a vCalendar parser shall accept the value as valid.

Though even if these property values were possible to encrypt, is such operation really necessary or only a waste of resources? For example, let’s say that an unauthorised person somehow manages to get hold of an exchanged vCalendar object or some how manages to access another users electronic calendar database. Further let’s anticipate that all these properties formatted according to ISO 8601 are in plain text. The unauthorised person can now easily find out information in different time and date properties since this information is in plain text, however this person then has to break the ciphers to obtain the rest of the information, such as other participants in an eventual appointment, the location of the appointment, description or a brief summary of the contents of the appointment etc. Now lets assume that the vCalendar object concerns an evening event and that the unauthorised assumes that this event is not taking place at my home (he or she can’t be certain since the “**LOCATION**” property value is encrypted). This person now knows how much time during that specific evening he or she has to break into my home and steal valuables and confidential documents. However if the personal information hadn’t been protected the unauthorised user would have been certain that I wasn’t at home since information concerning the location for the event would have been visible in plain text.

[19] RFC 822 specifies a syntax for text messages that are sent among computer users, within the framework of “electronic mail “ [Crocker].

Mandatory properties

My personal opinion is that none of these mandatory properties, conforming to ISO 8601 contains any confidential information and encrypting them will only consume resources (such as CPU power, battery capacity, and bandwidth). Naturally there are many different opinions and giving away information when different events in a persons calendar begins and ends or when the event is supposed to be completed (**DUE**) might be abused by fraudulent people as described above. Nevertheless my personal opinion is that this risk is lower than the cost of protecting the properties described below.

Start Time and Date
The "**DTSTART**" property can only occur within a vEvent object and specifies the time and date when a vEvent object begins. The property value is an ISO 8601 based date and time format and can be a local or UTC based time.

End Time and Date
This property can only be used within a vEvent object and defines the time and date when the vEvent object will end. The property name is "**DTEND**" and the value is an ISO 8601 time and date format and can be a local or UTC based time.

Time and Date Completed
This property can only be used within a vTodo object and defines the time and date that the action item in the vTodo object was completed. The parameter name is "**COMPLETED**" and the parameter value is represented by a basic time and date format as specified in ISO 8601. The value can be expressed as local time or as a UTC based time.

Due Time and Date
The "**DUE**" property can like the "**COMPLETED**" property only be defined within a vTodo object and expresses the time and date when the information in a vTodo object is due to be completed. The property value is represented by a complete ISO 8601 basic time and date format and can be either a local or a UTC based time.

Optional Properties

Time and Date Created
The "**DCREATED**" property expresses the time and date when the calendar entry was created in the originating calendar system. This value must not necessarily be the same time and date when the vCalendar object was created. The property value is expressed as local or UTC based time format as described is ISO 8601. This property value neither contains any confidential information nor does it affect the calendar owner's personal privacy in any way. Therefore I believe there is no need to encrypt the property value.

Display Reminder
This property can be used as a visual reminder for the for the vCalendar entry. The property name is "**DALARM**" and the property value is a combination of: *an ISO 8601 date and time number | the duration time when the reminder shall be executed once more after the first execution | how many times the reminder shall be repeated* and *finally a text string containing the text to be displayed*. As long as the text string is entered in such way that it doesn't reveal any information regarding the content of the calendar entry there is no need to encrypt the property value. However, since the main purpose of the text string is to give a short summary of the calendar entry information, it is always better to be on the safe side so it might be a good idea to encrypt this property value. Moreover the easiest way to parse this string is to encrypt all information that occurs after the property name "**DALARM**", shown below.

Original version

DALARM:20010626T1655;PT5M;2;Example

Encrypted version

DALARM:abcdefghijklmnopqrstuvxyz123

Mail Reminder

A reminder can also be send to an e-mail address conforming to RFC 822. The property name is "**MALARM**" and the property value is a combination of: *an ISO 8601 date and time number when the reminder shall be executed* | *the duration time when the reminder shall be executed once more after the first execution* | *how many times the reminder shall be repeated* | *the e-mail address where the text reminder should be sent* and *the text to be included in this message*. Similar with the "**DALARM**" property, as long as the text string send to the entered e-mail address is entered in such way that it doesn't reveal any information regarding the content of the calendar entry there is no need to encrypt the property value. However, since the main purpose of the text string is to give a short summary of the calendar entry information, it is always better to be on the safe side hence this property value should be encrypted if this property is used. Moreover the easiest way to parse this string is to encrypt all information that occurs after the property name "**MALARM**", in the same way as described for "**DALARM**".

Procedure Reminder

A reminder can also be executed as a special program or procedure in addition to the e-mail and display reminder. Similar to the first two reminders the property value is a combination of: *an ISO 8601 date and time format when the reminder shall be executed* | *the duration time when the reminder shall be executed once more after the first execution* | *how many times the reminder shall be repeated* and *the URL conforming to RFC 1738 where the external file is found*. This property doesn't contain any information regarding the contents of the calendar entry and is therefore unnecessary to encrypt. However, it contains a URL to an external resource, which might be a file containing a Trojan horse. Because it is possible to send vCalendar object to other persons a user must be careful before accepting the vCalendar object. This is the same problem that exists with attach files to e-mails and encrypting the property value wouldn't protect the user against such attach. The remedy for such scenario is to be conservative with accepting information from unknown persons. This property is optional and vCalendar parser developers may chose not to support it. Basically this is the only way for a user preventing being exposed of such an attack.

Exception Time and Date

The "**EXDATE**" property is a list of exception dates and times for recurring calendar entries. The property value is expressed as an ISO 8601 basic date / time format and this value can be expressed either as local or as UTC based time. It is useful of excluding certain times and dates when a recurring event should have taken place. For example if a department meeting occurs every Monday at 3 p.m. except on holidays, these dates can be excepted and the user won't be reminded about the meeting on his or her holiday. Whether an event is excluded on a special date or not is in my opinion not especially confidential. The property value only reveals times and dates when the calendar entity won't take place and therefore my personal opinion is that neither this property value contains information worth the costs of encrypting.

Recurrence Time and Date

This property enables the possibility to define a list with times and dates when a calendar entry shall recur. The Property name is "**RDATE**" and the property value is formatted according to ISO 8601. As well as the former property "**EXDATE**" neither this property is worth encrypting. For example, lets pretend that all petrol companies have agreed of meeting each other once each month at different times and dates to determine the petrol prices.

This is called a cartel and is forbidden according to Swedish laws and regulations (as well as in many other countries) and the involved persons can be convicted and sentenced to jail. Lets assume that another unauthorised person manages to get hold of the whole calendar entity marked as a vCalendar object. This person can now see that this meeting recurs once each month at different times and dates, but this person has absolutely no clue of the content of the meeting since this information is encrypted. To get this knowledge the unauthorised person must decrypt either the "**DESCRIPTION**" property or the "**SUMMARY**" property. Obviously the fraudulent information thief can tail the calendar owner to see where the meeting takes place. Nevertheless in my opinion, the risk that such a thing would happen is worth taking.

Last Modified
The "LAST-MODIFIED" property enables an application to specify when (time and date) something was changed in the calendar entry. The property value is an ISO 8601 basic time and date format and the SyncML Sync Engine may use this property parameter in the synchronisation session [Sync. Protocol]. Since it might be used by SyncML and doesn't reveal any information regarding the contents of the calendar entry it should not be encrypted.

VCalendar properties formatted as text or according to RFC 822 or RFC 1738

There are two types of properties containing text values. One type of property requires that the value be chosen from a set of pre-defined values.

The problem with these properties is that replacing the plain text value with cipher text might cause interoperability problems between vCalendar parsers since the cipher most likely won't be one of the pre-defined values. Consequently neither of these property values is possible to encrypt and at the same time ensure that the vCalendar object will be possible to decode independently of implementation.

The other type of properties with values formatted, as text is text properties where the value is free text. Since theses values don't have any constrains, encrypting the value and replacing it with the cipher text will most likely not cause any problems as long as the terminating CRLF sequence is preserved. These are basically the only properties that can be encrypted without violating the vCalendar formatting rules and grammar. However, the problem still remains of how the originator can inform the recipient that the information is encrypted and how it has been encrypted, i.e. algorithm, key length etc., since the vCalendar format doesn't support arbitrary MIME types or any other support for including binary information.

The last two property types are those containing an e-mail address according to RFC 822 or a URL address according to RFC 1738. Both these types of property values are in general desirable to encrypt since they reveal confidential information from a personal privacy perspective. Unfortunately the same thing applies to these as to the property values conforming to ISO 8601, i.e. they are impossible to encrypt while maintaining the parseability since the cipher will most likely conform to neither RFC 822 nor RFC 1738.

Mandatory properties

Categories
The vCalendar format has build-in support to categorise calendar entries information. The property name is "**CATEGORIES**" and the format of this property's value are pre-defined text values, which can be chosen from one of these values: *appointments*, *business*, *education*, *holiday*, *meeting*, *miscellaneous*, *personal*, *phone call*, *sick day*, *special occasion*, *travel* and *vacation*.

Therefore replacing the property value with cipher text will probably give rise to pars ability problems. Despite this problem the property value is interesting to encrypt since it reveals information regarding the category of the calendar entry.

Description
The "**DESCRIPTION**" property gives the calendar owner the possibility of giving a more thorough description than the "**SUMMARY**" property. This property value gives away lot of confidential information regarding the contents of the calendar entry and **absolutely** must be encrypted to ensure the calendar owners personal privacy.

Priority
The "**PRIORITY**" property specifies the priority for the calendar entry. The property value is an alphanumeric integer number where 0 is an undefined priority, 1 is the highest priority value, 2 the second highest priority, etc. This information is in my opinion confidential, since it reveals the calendar owner's opinion of the information. If the entry is copied the malicious person could easily decide which entries to attack, simply by screening the values of this property. Obviously this person begins with the entries with highest priority since these will probably contain the most confidential information. The problem is that the value is a simple integer number and knowing the used cryptographic algorithm the cipher text is easily broken. Nevertheless this property's value should somehow be protected.

Status
The "**STATUS**" property is used to indicate the status of a vCalendar object. It helps the calendar owner to keep track of different entries in his or her calendar. The default value is "**STATUS** = needs action" but this can be replaced with any of the pre-defined values. These are:

Property Value	Description
ACCEPTED	Indicates that the sent *vTodo* object was accepted.
NEEDS ACTION	Indicates that a *vEvent* or *vTodo* object requires action.
SENT	Indicates that the present *vEvent* or *vTodo* object was sent out.
TENTATIVE	Indicates that the *vEvent* was tentatively accepted.
CONFIRMED	Indicates that the recipient accepted the sent *vEvent*.
DECLINED	Indicates that the recipient has declined the *vEvent* or *vTodo* object.
COMPLETED	Indicates that the *vTodo* object has been completed.
DELEGATED	Indicates that the *vEvent* or *vTodo* object has been delegated.

The property value reveals the condition of a vCalendar object and is in my opinion confidential. Just as the "**PRIORITY**" property value this one gives a malicious person the incentive to easily select which encrypted property values to attack simply by screening the calendar entry's "**STATUS**" value. Therefore the value is confidential and should be protected.

Summary
The "**SUMMARY**" property contains a summary of the content of the associated calendar entry. The property name is "**SUMMARY**" and this property value contains information that must be encrypted and replaced with cipher text since its reveals lots of information regarding the content of the calendar entry. Furthermore the property value is not constrained to a set of pre-defined value as the "**CATEGORY**" or the "**STATUS**" property has, which makes it much easier to replace the property value with the corresponding cipher text.

Optional properties

Attachment
The vCalendar format supports attaching documents or other objects to vCalendar objects with the use of the "**ATTACH**" property. The attached object is either located inline with the "**ATTACH**" property or the property value is a reference to another MIME message body part or to a URL where the attached object can be found. Whether this property should be encrypted or not is a very interesting question. My personal opinion is that all attached information should be considered of such importance that it should be encrypted independently of whether the property value is a URL, or if it is an attached file. Perhaps an external document is attached, revealing the confidential content of the meeting?

Attendee
The "**ATTENDEE**" property enables the calendar owner to include other users in vEvent and vTodo objects. This property is useful for scheduling multi-user events such as meetings or other get-togethers. The "**ATTENDEE**" property value can be an e-mail address of the participating user or a URL reference, referring to a users vCard. If a URL value is used to point to the location of the contact information then the two property parameters, "**VALUE**" and "**TYPE**" must be used. This property value may contain other participant e-mail addresses that must be considered as confidential and hence the whole property value should be encrypted.

Classification
Support for this property is optional and it enables the possibility for the user to define the confidentiality classification of the information in the current vCalendar object. This property has three possible values: *Public*, *Private*, and *Confidential*. This Property has no real security function, but rather gives the calendar owner the opportunity to notify receiving applications how the vCalendar object should be handled. Encrypting this property value is more or less unnecessary since it doesn't really contain any confidential information. However, it reveals the calendar owners opinion regarding the information in the calendar entries and viewed from this perspective the value might be interesting to secure by using encryption.

Location
The property "**LOCATION**" makes it possible for the calendar user to define a location where the calendar entry event shall take place. A location can be a room number, a building or, a vCard. If a vCard is used, the "**VALUE**" where the vCard can be found must be set to "*URL*" and the "**TYPE**" parameter must be set to "**TYPE** = *VCARD*". This property value should be considered as confidential, in the same way as the "**SUMMARY**" property. Therefore the property value should be encrypted.

Related To
The "**RELATED-TO**" property represents relationships and references between the current vCalendar object and another vCalendar object, since they can't be nested within each other. If some information in one object is changed, the change spreads to the related vCalendar objects, which are then updated with this new information. For example, if the start time in one calendar entry is changed, the start time in all related vCalendar objects should be changed as well. The property value is a lasting, globally unique identifier or another calendar object's Unique Identifier (UID). Since this property is merely a pointer to another vCalendar object there is no reason why it's property value should be encrypted.

Recurrence Rule
One advantage of an electronic calendar is that recurring events can easily be maintained and only need to be entered once. To enable this, the Versit initiative has developed a property called "**RRULE**" that makes it possible for the calendar owner to establish rules based on the Basic Recurrence Rule Grammar of XAPIA's CSA[20] for repeating pattern for a recurring calendar entry. The property value tells how often a certain calendar entry occurs and from this value conclusions might be drawn regarding the contents of the information. If the other suggested properties are encrypted, then the calendar owner's personal privacy should be protected well enough to leave this property value in plain text.

Exception Rule
The exception rule "**EXRULE**" specifies a rule or repeating pattern for an exception to a recurring calendar entry. The property value is based on the Basic Recurrence Rule Grammar of XAPIA's CSA. Following the same reasoning as for the "**RRULE**" property encrypting this property might be unnecessary.

Resources
This property makes it possible for the calendar owner to specify any equipment or resources needed in the vCalendar object. The property name is "**RESOURCES**" and uses pre-defined property values such as: "*CATERING*", "*CHAIRS*", "*TV*", "*VIDEO PHONE*", or "*VEHICLE*", etc. This property value reveals calendar entry information that can be classified as confidential and should therefore be encrypted.

Sequence Number
This property makes it possible to easily detect if any changes have been made to the vCalendar object. This can be useful in a synchronising session. Unfortunately the property doesn't specify what has been changed, only that a change has been made. The property value is a number, indicating the number of times that the calendar entry has been changed since it was created for the first time. The property name is "**SEQUENCE**" and the initial value when the calendar object is created is 0. The SyncML Sync Engine can use this property value for synchronising properties and therefore it should be left untouched [Sync. Protocol].

Time Transparency
The "**TRANSP**" property defines whether a calendar entry is transparent to free time searches or not. The property value is an integer value, where 0 guarantees that the entry will block out time and will block a free time search by other users. 1 specifies that the entry will not block out time and will not block a free time search by other users. "**TRANSP**" property values higher than 1 are used for implementation specific transparency semantics. Some applications may treat these values greater than one as non-blocking or transparent calendar entries. Other applications may use theses higher numeric value to provide a layering of levels of transparency. This property value should not be encrypted since vCalendar parsers use it. In addition it doesn't reveal any confidential information as no conclusions can be drawn from this.

Unique Identifier
The unique identifier "**UID**" property specifies a persistent, globally unique identifier that can be used to identify a specific vCalendar object, for example the property value can be an ISO 9070 Formal Public Identifier (FPI) [Versit 1], X.500[21] distinguished name, machine-generated "random" number with a suitable probability of being globally unique or a Uniform Resource Location (URL). The "**UID**" is an optional property, although the vCalendar format recommends support for this property in more complex calendar applications [Versit 1].

[20] XAPIA CSA stands for X.400 API Association's Calendaring and Scheduling API [Versit 1]. This Recommendation provides an overview of system and service of Message Handling Systems (MHS).
[21] The ITU specification X.500 describes "open systems interconnection"

Extensions
The vCalendar format allows calendar applications to include their own properties. Such property must begin with the two-character combination "X-" and all vCalendar parsers are expected to be able to decode these extension properties, but may ignore them. Since there is no central register for these vendor specific properties, the Versit initiative recommends that a short identification string for the vendor be added after the initiation string "X-". For example vendor ABC's extension for an audio-clip might have the following appearance:

X-ABC-MMSUBJ; TYPE=WAV; VALUE=URL; http://load.noise.org/mysubj.wav

This property is very interesting since it allows application developers to define their own properties. A number of the vCalendar format properties that should be encrypted, but would cause interoperability problems; can be redefined as a X-property. For example, the property "**ATTENDE**E" would then be: "**X–ATTENDEE**". Application developers can decide whether to support these properties containing encrypted data or not. Nevertheless, the problem still remains that these properties can't contain arbitrary MIME type property values and only three MIME types (WAVE, PCM, and VCARD) are supported by the vCalendar format. However, the iCalendar format, which is a successor to the vCalendar format, allows arbitrary MIME types as property value.

5.2.1.6 A Brief Summary of the vCalendar Format Version 1.0

As a summary of the vCalendar format a table can be made containing all available properties indicating whether I believe they should be encrypted or not. This table shows that approximately half of all properties are not worth encrypting without jeopardising the calendar owner's personal privacy or any other person occurring in the calendar. Naturally there is an abundance of objections as to whether a specific property value should be encrypted or not. Although, as explained earlier the main reason for this analysis was to show why it is not possible to combine confidentiality with the three PIM formats vCalendar, iCalendar, and vCard; and to investigate if it was possible that both resources and used bandwidth could be reduced, which are important for many mobile communication devices and their users.

Property	Property name abbreviation	Worth encrypting	Not worth encrypting	Mandatory
Geographic position	**GEO**	X		
Time Zone	**TZ**	X		
Version	**VERSION**		X	X
Product identifier	**PRODID**		X	
Daylight savings rule	**DAYLIGHT**		X	
Start Time and Date	**DTSTART**		X	X
End Time and Date	**DTEND**		X	X
Time and Date completed	**COMPLETED**		X	X
Due time and date	**DUE**		X	X
Time and date created	**DCREATED**		X	
Display Reminder	**DALARM**	X		
Mail Reminder	**MALARM**	X		
Procedure Reminder	**PALARM**		X	
Exception Time and Date	**EXDATE**		X	
Recurrence Time and Date	**RDATE**		X	
Last Modified	**LAST-MODIFIED**		X	
Categories	**CATEGORIES**	X		X
Description	**DESCRIPTION**	X		X
Priority	**PRIORITY**	X		X
Status	**STATUS**	X		X
Summary	**SUMMARY**	X		X
Attachment	**ATTACH**	X		

Attendee	**ATTENDEE**	X		
Classification	**CLASSIFICATION**	X		
Location	**LOCATION**	X		
Related To	**RELATED-TO**		X	
Recurrence Rule	**RRULE**		X	
Exception Rule	**EXRULE**		X	
Resources	**RESOURCES**	X		
Sequence Number	**SEQUENCE**		X	
Time Transparency	**TRANSP**		X	
Unique Identifier	**UID**	X		

This analysis clearly shows that deciding whether encrypting a property value is necessary or not can save both resources and bandwidth. More importantly the number of properties that might be redefined using the experimental "X-" property is also limited and mainly restricted to a few mandatory properties.

5.2.2 iCalendar Format

The iCalendar 1.0 core object specification was developed by the Calendar and Scheduling workgroup (Calsch WG) within the IETF and its intention was to increase the level of interoperability between devices with dissimilar calendar and scheduling applications. The specification can be found in RFC 2445 [Dawson 1]. It is the core specification from which three additional protocols concerning PIM are based upon.

These additional protocols include the two-transport protocols iTIP [Silverberg] described in RFC 2446 and iMIP [Dawson 3] described in RFC 2447 for calendar information exchange between different dissimilar applications, and a Calendar Access Protocol (CAP) described in [Mansour]. CAP describes how to access a calendar server[22] in real time [Mahoney]. The iCalendar Transport Independent Interoperability Protocol (iTIP) specifies how calendar and scheduling applications can use iCalendar objects to exchange calendar and scheduling information. Such interoperability can involve scheduling of events, to-dos, etc. The iCalendar Message-Based Interoperability Protocol (iMIP) specifies a binding from iTIP to Internet email-based transports. All three additional protocols are used to simplify PIM exchange between different person's electronic calendars and are not used by SyncML compliant mobile communication devices for information up-date or maintenance. In addition they are all based on the iCalendar 1.0 core object specification and therefore do not need any further explanation.

The iCalendar format is heavily based on its predecessor the vCalendar format and calendar and scheduling information are translated into plain text and marked-up in the same way using almost the same properties, property parameters, and property values.

Property name [';' *property parameters* '=' *property parameter value*] ':' *Property value*

The differences between the two calendar and scheduling formats are that the iCalendar format has an extended set of available property parameters. Furthermore these are more strictly specified as to which property parameters they could be combined with, to associate and to describe meta-data of either the property or the property value and in addition there no longer exists any property specific parameters. Another difference is the extension of components from two in the vCalendar format (vEvent and vTodo) to six in the iCalendar format (vEvent, vTodo, vJournal, vFreebusy, vAlarm, and vTimezone).
The vJournal, vAlarm, and vTimezone can also be found within the vCalendar format, but with a simplified functionality and as single properties [Versit 1].

[22] Examples of calendar servers are Microsoft Exchange and Lotus Domino.

The vFreebusy component is new and is used by other calendar users to send a time frame request to another calendar user. How to use this component to utilize this functionality between two person's electronic calendars is described in [Silverberg]. This functionality simplifies the procedure when two or more persons try to schedule an appointment. In addition it eliminates the need for other users to access another user's calendar to search for a free time frame to schedule an appointment, which was the only way to realise this functionality in the vCalendar format. In addition, this component is not involved in the SyncML synchronisation process.

Concerning the personal privacy of the calendar and scheduling information the same problems that exist in the vCalendar format regarding encrypting single property values independently of each other while preserving parseability exists in the iCalendar 1.0 core object specification. The same property value format types, which make it impossible to protect information, are also found in the iCalendar format. Furthermore there is no possibility to indicate and mark-up, for example via a property parameter, that a specific property value is encrypted. Although the iCalendar format has one significant advantage over the vCalendar format in that it eliminates the need for other users to access another user's electronic calendar by instead using the vFreebusy component. This functionality doesn't really exist in the vCalendar format, but could be provided by encrypting the calendar information and allowing other authorised users to have access to the encrypted information in the electronic calendar.

Thanks to the close relationship between the iCalendar and the vCalendar formats our analysis of how the information in a calendar database entry should be marked is similar. Thus the analysis that was made for the vCalendar format can be directly applied to the iCalendar format, even though the iCalendar format is more extensive and complex. Therefore a similar analysis will not further contribute to the objectives of this project. In addition such analysis is very subjective and naturally varies from person to person. The intention of my analysis of the vCalendar format was to first of all to prove that it isn't possible to encrypt single property values and preserve the parseability of these vCalendar objects.

However, it is worth noting that the corresponding "TYPE" property parameter, in the iCalendar format called "FMTYPE" can have any IANA registered MIME type as a property parameter value unlike the vCalendar format (that only could have three pre-defined values). Unfortunately this property parameter can only be used together with the "ATTACH" property and can't be used for solving the problems addressed in this thesis.

5.3 Contact Formats

5.3.1 vCard

VCard is an electronic business card format for marking-up information regarding people and resources, better known as contact information. This type of information is often found on business cards, in personal contact books, or in mobile communication devices; and is often shared among different persons and devices. Since different mobile communication devices from different manufacturers most likely are equipped with different contact applications the contact information must be as interoperable as possible between these devices and therefore the vCard format is based on a plain text representation.

Furthermore the contact information is also mark-up with properties, each one containing specific parts of the contact information in a similar way as the calendar and scheduling information is in both the vCalendar and iCalendar formats.

Property name [';' *property parameters* '=' *property parameter value*] ':' *Property value*

The vCard format is available in two versions 2.1 and 3.0 and the SyncML Device Information Document Type Declaration supports both versions. The vCard format 2.1 can be found in [Versit 2] and vCard version 3.0 can be found in RFC 2426 [Dawson 3].

VCard version 3.0 has been reviewed and approved by the IETF and contains some changes compared with version 2.1. However, none of these changes contributes to protecting single property values. The changes in version 3.0 compared with version 2.1 can be found in chapter 5 [Dawson 3].

The different properties and the information in the property values in both versions of the vCard formats are based upon the International Telegraph Union (ITU) X.500 Series Recommendation for Directory Services. Furthermore the vCard format was designed to make it possible to map attributes and objects found in X.520 and X.521 into and out of a vCard object [Versit 2].

The property value format types such as time formatted according to the ISO 8601 specification, email according to the RFC 822, and URLs according to the RFC 1738 that are found in the vCalendar and iCalendar formats are also found in the vCard format. Besides these, additional contact information specific value types have been added; for example, telephone number according to the X.520 Telephone Number Attribute, physical mail delivery address for the vCard object according to attributes in both X.500 and x.520, etc. [Versit 2].

Unfortunately, as with both the vCalendar and iCalendar formats these property value types make it impossible to encrypt single property values and at the same time preserve the parseability of the vCard object. Moreover there exists no support to encrypt single property values independently of each other. Even if it was possible to encrypt these property values while preserving parseability vCard (just as both the vCalendar and iCalendar formats) doesn't contain any possibility to indicate and mark that the contacts information in the property value is encrypted. Similar to the vCalendar format a property specific parameter "TYPE" can be used to mark-up the type of information in combination with a limited number of properties. Unfortunately each of these properties has their own pre-defined set of property parameter values and arbitrary IANA registered types cannot be used. For example the Public Key property "KEY" whose value is the public key belonging to the item described by a vCard object has two possible parameter values. These are the two different types of Public Key certificates [Versit 2]:

- Certificates conforming to the ITU X.509 specification
- Certificates conforming to the IETF PGP specification

Naturally these two property specific parameters can't be used together with for example, the telephone number property "TEL" and vice versa. As their property specific parameter "TYPE" has a completely different set of pre-defined values.

Since it is impossible to achieve confidentiality for single, independent property values and at the same time preserve the parseability of the vCard object no further examination of this electronic- business card format will be made. Since the same analysis can easily be made for both versions of the vCard format such an analysis will not be made in this report.

5.4 Future PIM Formats

Since it is not possible to combine the three PIM formats vCard, vCalendar, and iCalendar with confidentiality without modifying them does there exist any other standardised (future) PIM format that could be used instead? Well the answer to this question is both yes and no. There do exist proposals of how to describe PIM information in XML, but none of these has yet been standardised (as described earlier in the introduction to PIM formats).

Thanks to the power and flexibility of XML there have been many suggestions of how to mark-up PIM formats as XML, unfortunately all of these only exists as drafts and none of these has been standardised by either the IETF or any other organisation (such as the SyncML initiative). Despite these uncertainties and questions my personal opinion is that using a XML representation of the three PIM formats as described in this chapter in combination with the W3C XML Encryption security format will most likely enable both parseability and confidentiality of PIM information. How this will be possible is described below.

5.4.1 xCard

The SyncML initiative has discussed developing a XML contacts format called xCard. It is thought of as an alternative XML representation of the vCard format version 3.0. The xCard format is a direct mapping of the properties and property specific parameters found in the vCard format into XML entities. Just as a vCard object contains contacts information a SyncML xCard also object contains contacts information, but the information itself is marked-up as XML entities; unlike vCard where the information is marked-up via properties. This is not a major semantic difference although by using XML it now becomes possible to use XML Encryption to encrypt the plain text and replace it by cipher text.

An important difference is that the property value format types that were found in both calendar and scheduling formats and in the vCard format only partially exist in the SyncML xCard format (according to the SyncML xCard draft). Fortunately none of the affected property values are necessary to encrypt. The affected entities are those concerning the geographical position of the item described by a SyncML xCard object and entities concerning time and date. Entities concerning time and date must be formatted as an ISO 8601 valid number and the geographical position are the decimal degrees of the Latitude and Longitude. Entities containing time and date information are “BIRTHDAY”, “TZ” (Time zone), and “REV” (Last Revised) and they don’t reveal much confidential information and thus do not affect the personal privacy in a major negative way. Regarding the “GEO” entity some persons might consider this information as confidential according to the analysis that was made for the vCalendar format, however this property (according to the vCard format version 3.0[23]) is an optional property and can be excluded. But if the property is used then the property value must remain in plain text.

Beyond these restrictions all other entities containing contact information must be formatted as character data (#PCDATA). Therefore it is at least theoretical possible to replace the unprotected contact information with its cipher text representation. Unfortunately the SyncML xCard format doesn’t provide (just as the other PIM formats) any means to indicate that the information in the entity has been replaced by cipher text.

[23] The RFC 2426 describing the vCard format 3.0 don’t explain whether some property is optional or not and therefore I have presumed that the same properties in the vCard format version 2.1 are optional in the vCard format version 3.0.

Fortunately this is a minor problem and can be solved by using XML Encryption, shown in the following example[24]:

E-mail address in plain text:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<XCARD>
    <VERSION>2.0</VERSION>
    <FN>Torbjörn Borison</FN>
    <N>
        <FAMILY>Borison</FAMILY>
        <GIVEN>Torbjörn</FAMILY>
    </N>
    <EMAIL>
        <PREF>
            <USERID>t.borison@home.se</USERID>
        </PREF>
    </EMAIL>
</XCARD>
```

E-mail address encrypted, using XML Encryption (entity content encryption):

```
<?xml version=”1.0” encoding=”UTF-8”?>
<XCARD>
    <VERSION>2.0</VERSION>
    <FN>Torbjörn Borison</FN>
    <N>
        <FAMILY>Borison</FAMILY>
        <GIVEN>Torbjörn</GIVEN>
    </N>
    <EMAIL>
        <PREF>
            <USEDID>
              <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
                             Type='http://www.w3.org/2001/04/xmlenc#Content'>
                <EncryptionMethod Algorithm='http://www.w3.org/2001/04/
                          xmlenc#3des-cbc '/>
                <CipherData><CipherValue>A23B45C56</CipherValue></CipherData>
              </EncryptedData>
            </USEDID>
        </PREF>
    </EMAIL>
</XCARD>
```

Due to the flexibility of the XML mark-up language the SyncML xCard format in combination with XML Encryption will most likely make it possible to achieve the goals o this master’s thesis. Another advantage is the possibility to use WBXML [WBXML]. Using WBXML allows the entities to be expressed as much shorter token streams thereby reduce the amount of data that must be transmitted and consequently saving bandwidth. Upon reception the WBXML coded information is expanded back to XML

Unfortunately so far there exists no XML representation describing how to mark-up calendar and scheduling information and it is still uncertain whether the SyncML xCard format will be released or not. If it is, changes might have to be made to that SyncML xCard version to avoid prohibiting encrypting single entities of SyncML xCard objects in the same way as for the other three PIM formats. Nevertheless changing the vCalendar version 1.0, iCalendar version 1.0 and the vCard version 2.1 and version 3.0 formats will be more difficult than using a combination of XML based PIM formats and XML Encryption. Hence using a PIM XML representation is more or less the only possibility to solve the problems addressed in this project.

[24] This example is based on information found in the working draft from the XML Encryption workgroup within W3, dated June 26, 2001

6. Solution

The focus of this thesis project was to investigate and solve the problem of combining SyncML, object security and thus protection of personal information, such as contacts and calendar entries in mobile devices. This problem was divided into three smaller ones:

1. Is it possible to encrypt single fields in independently of each other, using any of the PIM formats iCalendar, vCalendar and vCard without interfering with the synchronisation specification SyncML?

2. If it proves to be impossible to add encryption of the information in the fields, what are the problems and can they be solved. Are there any other formats that can be used without interfering with SyncML?

3. Finally if it is too complicated to add protection to the different fields in an entry or if there aren't any suitable formats that can be used will it be possible to support this type of protection in future formats. What is the ideal solution that requires as little as possible to be changed such that the solution is as attractive as possible?

My efforts have resulted in the following answers:

Unfortunately as presented in chapter 5, it isn't possible to encrypt single fields using any of the PIM formats iCalendar, vCalendar and vCard. The primary reason is the advantage of synchronising information; the possibility to update and replace parts of the information, leaves the rest of the information intact. If the added protection is not to interfere with synchronising of information using SyncML, each field of a PIM database entry must be encrypted independently of others. These fields in a database entry are then encoded and mapped by a PIM format into easily understandable parts, i.e. properties. Furthermore the property values often have preformatted value types (a URL address, an e-mail address, time). For example property values describing time are formatted according to the international standard ISO 8601. Unfortunately these property value types are also the primary problem since replacing the plain text with cipher text is impossible without violating the strictly defined property value types in the PIM format. Since PIM formats are used by applications to encode personal information for easier exchange between them, violating these PIM format may cause interoperability problems between these applications.

Consequently these property value types are one problem. Since these are static the only solution is changing the PIM formats. A subsequent problem is how to recognise the content of the property values are cipher text? Using IANA registered MIME types, if we consider the property value as a small message; we use either CMS and S/MIME or PGP to partially solve this problem. The remaining problem is how the PIM format compliant application may recognise the MIME content type of the property value? Both vCalendar and vCard contain a property specific parameter called "**TYPE**". Unfortunately neither of the two PIM formats allows arbitrary IANA registered MIME types as parameter values or allows the "**TYPE**" parameter to be used in combination with any properties other than the specified. Furthermore none of the "**TYPE**" parameter values found in either vCard [Versit 2, p. 25] or vCalendar [Versit 1, p.34] are suitable for use with either CMS and S/MIME or PGP. However, the corresponding property parameter, "**FMTYPE**" in iCalendar is global, i.e. it is defined for all properties and it's value can be any IANA registered MIME type. Unfortunately this global property parameter may only be used in combination with the "**ATTACH**" property [DAWSON 1].

Consequently the vCard, vCalendar and iCalendar format, all have some support for marking the content of the property value. However, this support isn't possible to use in an appropriate way of solving the problems addressed in this thesis. What is needed is a global property parameter defined for all properties with values corresponding to any IANA registered CMS and S/MIME or PGP MIME types. Unfortunately adding such a property parameter is the only solution would require changing the PIM formats.

Another solution is using the extension "**X-**" property to add developer dependent properties. All properties considered to be worth encrypting could be extended, for example "**X-DESCRIPTION**". Moreover both supported property parameters and property type formats are free of choice. The "**X-**" extension may also be used to specify new "**TYPE**" property parameter values in both vCard and vCalendar. For example, such new "**TYPE**" property parameter values may be "**TYPE** = *X-Applicationpkcs#7-mime*" or "**TYPE** = *X-Applicationpgp-encrypted*". Such a solution does change the PIM formats, but causes no interoperability problems, since "**X-**"properties and property parameter values are disregarded if not supported by the decoding application. Taken together the new extended properties with CMS / S/MIME or PGP encoded values might look like this:

```
X-"Property name -Developer"; ENCODED=BASE64; TYPE= X-Applicationpkcs#7-mime:
                              CMS / S/MIME content

X-"Property name -Developer"; ENCODED=BASE64; TYPE = X- Applicationpgp-encrypted:
                           PGP content
```

Obviously none of these solutions are preferred, since changing PIM formats causes parsing problems with applications not conforming to the changed PIM format. Using the extension possibility causes information loss and synchronisation problems, since the "**X-**" property and it's content is simply disregarded if not supported. When this information is synchronised, parts of the information will always be missing and retransmitted because they are always disregarded and never stored. Thus both the interoperability and the synchronising goals are violated.

Another solution is founded in drafts of future PIM formats, and future cryptographic formats. As mentioned in chapter 5, the SyncML initiative has discussed the possibility to use an XML representation of the vCard contact format called xCard. This format unlike vCard allows any content type to be stored in all entities except three. As explained in chapter 5.4.1, two of these entities neither affects nor reduce the protection of the rest of the information found in the xCard object. The third contains the geographical position of the item, described by the xCard object and may be considered confidential by some users. Luckily this entity is optional and support for it can be excluded in xCard parsers.

Furthermore a new cryptographic format, XML Encryption intended for being used on XML documents, XML entities, or the content found in an XML entity, was discussed in the XML Encryption workgroup at W3. This WG has recently (June 26, 2001) released a promising working draft, solving the both the protection and the content type detection problem.

Using an XML PIM format also makes it possible to use XML Encryption. Since XML Encryption supports encryption of content information, while the rest of the information remains intact. Combining XML PIM formats with the XML Encryption format may be a possible solution to the problems of this thesis project. However, some conditions must be fulfilled. Primary XML PIM formats and the XML Encryption format must become standards, preferably expressed as a RFC documents.

Furthermore implementations of both formats must be made in all SyncML compliant devices and servers. Whether agreements concerning standardisation of XML PIM formats will be made or not is uncertain, while the release of the XML Encryption format it is only a question of time.

Despite of the uncertainty regarding the release and design of future XML PIM formats standards' combining them with XML Encryption is the ideal solution. The solution is both transparent to SyncML and fulfils the interoperability condition since no changes of either format are made.

Nevertheless an interesting question remains. How to permit cipher text as property values in all properties of the vCard, vCalendar, and iCalendar formats under the condition that as little as possible has to be changed in these formats. The best solution is adding a global property parameter in all three formats, called "**ENCRYPTED**", this is allowed to override all default property value types with base64. Moreover the property parameter value marks the cryptographic format used and is either an IANA registered PGP or CMS MIME type.

An example of this, using CMS / S/MIME could be:

```
BEGIN:VCALENDAR
  TZ:+01:00
  VERSION:1.0
  BEGIN:VEVENT
    DTSTART:20010910T090000z
    DTEND:20010910T100000z
    SUMMARY;ENCRYPTED = Application/pkcs#7-mime:
              Content-Type: application/pkcs7-mime; name=smime.p7m
              Content-Transfer-Encoding: base64
              Content-Disposition: attachment; filename=smime.p7m
              GhyHhfHf126GhIHKT6
    DESCRITION;ENCRYPTED = Application/pkcs#7-mime:
                Content-Type: application/pkcs7-mime; name=smime.p7m
                Content-Transfer-Encoding: base64
                Content-Disposition: attachment; filename=smime.p7m
                567GhIGfHfYT6ghyHh
  END:VEVENT
END:VCALENDAR
```

7. Conclusions and Future Work

7.1 Conclusions

This M.Sc. project was a research project to investigate whether it is possible or not to encrypt personal information marked according to the three most frequent used PIM formats (vCard, vCalendar, and iCalendar) under condition that the personal information would still be possible to synchronise using SyncML. There is no question that protecting personal information is growing in importance with the increasing use of new managing possibilities such as SyncML and more powerful mobile communication devices. Information security applications for encrypting contact and calendar databases of mobile communication terminals have already been released. However, personal information can also be stored on an Internet server, making it accessible independent of network or environment (office, home, in the car, etc.), and easily managed on multiple devices using SyncML. All the solutions presented provide additional protection to the synchronised personal information and preferably prevent or at least discourage information theft.

The negative aspect of the convenient information accessibility of storing information on an Internet server are, as Matt Hamblen pointed out in an article "Contact information is of corporate value that needs to be protected from prying eyes and is a target for corporate espionage, even if you have a secure transmission, you still have your corporate data sitting on someone else's server. What's the deal with that?" [Hamblen]. This also applies to calendar information, which if it falls into the wrong hands can cause enormous problems for the affected company. Furthermore there are a growing number of unscrupulous companies, buying e-mail lists and using them to send advertising e-mails, i.e. spam mail. Even if the Internet server uses some sort of secure storage, the operator still has knowledge regarding the keys, needed for decryption. Therefore it is theoretically possible for an employee, working for the server operator to simply copy the cipher text stored on the server, get hold of the keys, and decrypting the information and sell it to others. Angry employees stealing corporate information do exist. In addition, soon we will be receiving SMS[25] messages containing advertising material and then there will be a market for our mobile phone numbers as well.

When I first began to work on this project I was focused upon finding a solution founded on the three PIM formats vCard, vCalendar, and iCalendar in combination with CMS, S/MIME, or PGP, and SyncML. The primary reason was their wide acceptance by different applications and consequently such a solution would be easy to implement on devices already supporting these specification. However, as the project continued I discovered that it wouldn't be possible without changing these PIM formats. Obviously such changes will cause interoperability problems between applications supporting cipher text property values and the existing application

Then I discovered a draft from the SyncML initiative proposing a XML representation of a contact format. Simultaneously there were a number of different proposals of a cryptographic format, XML Encryption. Based on these drafts, the combination seemed to form a possible solution, despite raising many questions. Today the XML Encryption WG has released a "working draft" that confirmed my earlier assumption that encrypting only the entity content would be possible. The remaining questions are whether the SyncML xCard contact format will be released and mandatory supported by SyncML compliant devices or not? Furthermore will the SyncML initiative also develop and release an XML based calendar format?

[25] SMS is an abbreviation for Short Message Service and is a function found within GSM networks. SMS is an easy and convenient way for transmitting short messages (maximum 160 bytes) to other GSM compliant devices by using the mobile phone number of that device.

7.2 Future Work

What I have left out in this project is an actual implementation of the three proposed solutions. The first proposed solution was to add a property parameter called “Encrypted” to the three PIM formats. The second proposed solution was to use the extension property “**X-**“, defining own properties with cipher text content. Both solutions would require new implementations of all three PIM formats (vCard, vCalendar, and iCalendar) capable of coding and decoding cipher text as proposed in chapter 6. Furthermore this implementation must also be able to collect database entries, encrypt the single fields with an appropriate cryptographic format, and finally mapping the encrypted database entity field into the correct property.

The third solution is easiest to implement, because XML Encryption is already capable of encrypting and marking XML entity contents. What is required is combining a SyncML xCard coder / decoder with an XML Encryption coder/decoder. To my knowledge a SyncML xCard parser doesn’t exist. However, an implementation of an XML Encryption parser exists implemented by a research group within IBM [http://www.alphaworks.ibm.work/tech/xmlsecuritysuite].

8. References

[AirCalendar] Ericsson Mobile Communications AB, *AirCalendar white paper, Revision A*, April 5, 2000.

[Allen] T. Dierks, C. Allen, *The TLS Protocol [RFC 2246]*, http://www.ietf.org/rfc/rfc2246.txt, (Accessed: February 16, 2001).

[Alvestrand] H. Alvestrand, *Tags for the Identification of Languages [RFC 1766]*, http://www.ietf.org/rfc/rfc1766.txt (Accessed: March 13, 2001).

[Atkins] D. Atkins, W. Stallings, P. Zimmermann, *PGP Message Exchange Formats [RFC 1991]*, http://www.ietf.org/rfc/rfc1991.txt (Accessed: January 19, 2001).

[Berners-Lee] T. Berners-Lee, L. Masinter, M. McCahill, *Uniform Resource Locators (URL) [RFC 1738]*,http://www.ietf.org/rfc/rfc1738.txt (Accessed: March 12, 2001).

[Blom] R. Blom, M. Näslund, G. Selande, *Object Security and Personal Information Management*, Ericsson Research, SE-16480 Stockholm, Sweden, April 27 2001.

[Borison] T. Borison, *M-Commerce and security (white paper)*, Mobile Applications Initiative, Ericsson Inc., U.S. April 2001.

[Callas] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *OpenPGP Message Format [RFC 2440]*, http://www.ietf.org/rfc/rfc2440.txt (Accessed: February 20, 2001).

[Calsch] *Calendar and Scheduling (Calsch)*, http://www.ietf.org/html.charters/calsch-charter.html (Accessed February 19, 2001).

[Carrara] E. Carrara, *Wireless Adaptation of a Security Management Protocol Suite (M.Sc. Thesis)*, KTH –Department of Teleinformatics, June 1999.

[Certicom] Certicom Corp., Current public-key cryptographic systems (white paper), http://www.certicom.com/research.html (Accessed, February 16, 2001).

[Crocker] D. Crocker, *Standard for the Format of ARPA Internet Text Messages [RFC 822]*, http://www.ietf.org/rfc/rfc822.txt (Accessed April 26, 2001).

[Dawson 1] F. Dawson, D. Stenerson, *Internet Calendaring and Scheduling Core Object Specification (iCalendar) [RFC 2445]*, http://www.ietf.org/rfc/rfc2445.txt (Accessed: January 10, 2001).

[Dawson 2] F. Dawson, S. Mansour, S. Silverberg, *iCalendar Message-Based Interoperability Protocol (iMIP) [RFC 2447]*, http://www.ietf.org/rfc/rfc2447.txt (Accessed: January 10, 2001).

[Dawson 3] F. Dawson, T. Howes, v*Card MIME Directory Profile [RFC 2426]*, http://www.ietf.org/rfc/rfc2426.txt (Accessed: January 16, 2001).

[Device Inf.] SyncML Initiative, *SyncML Device Information DTD Specification V1.0*, http://www.syncml.org/docs/syncml_devinf_v10_20001207.pdf (Accessed: January 9, 2001).

[Dusse] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, *S/MIME Version 2 Message Specification [RFC 2311]*, http://www.ietf.org/rfc/rfc2611.txt (Accessed: January 19, 2001).

[Eastlake] D. Eastlake, J. Reagle, *XML Encryption Syntax and Processing, Working Draft 26 June 2001*, http://www.w3.org/TR/2001/WD-xmlenc-core-20010626 (Accessed: September 8, 2001).

[Eastlake 2] D. Eastlake, J. Reagle, D. Solo, *XML-Signature Syntax and Processing [RFC 3075]*, http://www.ietf.org/rfc/rfc3075.txt (Accessed: September 14, 2001).

[Elkins] M. Elkins, *MIME Security with Pretty Good Privacy (PGP) [RFC 2015]*, http://www.ietf.org/rfc/rfc2015.txt (Accessed: January 19, 2001).

[Elkins 2] M. Elkins, D. Del Torto, R. Levien, T. Roessler, *MIME Security with OpenPGP [RFC 3156]*, http://www.ietf.org/rfc/rfc3156.txt (Accessed: September 4, 2001).

[Ericsson Inc.] Ericsson Mobile Communications AB, *Remote Synchronisation white paper*, February 2000.

[Freed] N. Freed, *Gateways and MIME Security Multiparts [RFC 2480]*, http://www.ietf.org/rfc/rfc2480.txt (Accessed: January 19, 2001).

[Giacometti] S. Giacometti, *WAP Security (M.Sc. Thesis)*, KTH –Department of Teleinformatics, March 2000.

[Galvin] J. Galvin, S. Murphy, S. Crocker, N. Freed, *Security Multiparts for MIME, Multipart/ Signed and Multipart/ Encrypted [RFC 1847]*, http://www.ietf.org/rfc/rfc1847.txt (Accessed: January 30, 2001).

[Hamblen] Matt Hamblen, *Users eye Yahoo-Starfish offering warily*, http://www.computerworld.com/cwi/story/0,1199,NAV47_STO26935,00.html (Accessed: February 8, 2001).

[Hamblen2] Matt Hamblen, *Wireless Data Steps Closer to Reality*, http://www.computerworld.com/cwi/story/0,1199,NAV47_STO34003,00.html (Accessed: February 8, 2001).

[Housley] R. Housley, *Cryptographic Message Syntax [RFC 2630]*, http://www.ietf.org/rfc/rfc2630.txt (Accessed: January 19, 2001).

[Howes] T. Howes, M. Smith, *A MIME Content-Type for Directory Information [RFC 2425]*, http://www.ietf.org/rfc/rfc2425.txt (Accessed: January 16, 2001).

[Häggström] M. Häggström, B. Blank, *Mobile Communication in a Multiple Device Environment (M.Sc. Thesis)*, KTH –Department of Teleinformatics, December 1998.

[IrMC 1.1] *IrMC1.1*, http://www.irda.org (Accessed January 9, 2001).

[ISO 8601] International Standards Organisation, *Data elements and interchange formats – Information interchange – Representation of dates and times*, Second edition, December 15, 2000, http://www.iso.ch (Accessed: May 16, 2001).

[Kaliski] B. Kaliski, *PKCS #7: Cryptographic Message Syntax Version 1.5 (RFC 2315)*, http://www.ietf.org/rfc/rfc2315.txt (Accessed: January 19, 2001).

[Linder] D. Linder, *Transport Security for the next Generation Mobile Terminals Security (M.Sc. Thesis)*, KTH –Department of Teleinformatics, November 2000.

[Mahoney] B Mahoney, G. Babics, *Guide to Internet Calendaring*, http://www.ietf.org/internet-drafts/draft-ietf-calsch-inetcal.guide-00.txt (Accessed: February 21, 2001).

[Mansour] S. Mansour, D. Royer, G. Babics, P. Hill, *Calendar Access Protocol (CAP)* , draft-ietf-calsch-cap-05, expires January 18, 2002, http://www.ietf.org/internet-drafts/draft-ietf-calsch-cap-05.txt (Accessed: July 26, 2001).

[Mattila] M. Mattila, *Secure Communication in Mobile Internet (M.Sc. Thesis)*, KTH – Department of Teleinformatics, February 2001.

[Menezes] A. Menezes, P. Van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac.

[Meta Inf.] SyncML Initiative, *SyncML Meta Information DTD Specification V1.0*, http://www.syncml.org/docs/syncml_metinf_v10_20001207.pdf (Accessed: January 9, 2001).

[Puma] Puma Technologies, *Invasion of the Data Snatchers (A Puma Technology White Paper)*, April 1999, http://www.pumatech.com/enterprise/wp-1.html (Accessed: February 26, 2001).

[Radcliff] D. Radcliff, *Thinking ASP? Don't Forget Security!*, http://www.computerworld.com/cwi/story/0,1199,NAV47_STO53011,00.html (Accessed: February 8, 2001).

[Ramsdell] B. Ramsdell, *S/MIME Version 3 Certificate Handling [RFC 2632]*, http://www.ietf.org/rfc/rfc2632.txt (Accessed: January 19, 2001).

[Ramsdell 2] B. Ramsdell, *S/MIME Version 3 Message Specification [RFC 2633]*, http://www.ietf.org/rfc/rfc2633.txt (Accessed: January 19, 2001).

[RSA FAQ] *RSA Laboratories, Frequently Asked Questions About Today's Cryptography 4.1*, http://www.rsasecurity.com/rsalabs/faq (Accessed: February 16, 2001).

[Saarinen] M-J. Saarinen, *Attacks Against the WAP WTLS Protocol*, University of Jyväskylä 1999, http://www.jyu.fi/~mjos/wtls.pdf, (Accessed: February 13, 2001).

[Schmeck] Prof. Dr. H. Schmeck, *Lecture notes to chapter 4 and chapter 5 in the course Algorithms for Internet Applications*, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe (TH), Germany, http://www.aifb.uni-karlsruhe.de/Lehrangebot/Winter1999-00/AIA/, (Accessed February 16, 2001).

[Silverberg] S. Silverberg, S. Mansour, F. Dawson, R. Hopson, *iCalendar Transport-Independent Interoperability Protocol (iTIP)[RFC 2446]*, http://www.ietf.org/rfc/rfc2446.txt (Accessed: January 10, 2001).

[SSL 3.0] Netscape Corp., *SSL 3.0 Specification*, http://home.netscape.com/eng/ssl3/3-SPEC.HTM (Accessed: February 16, 2001).

[Sync. Arch] SyncML Initiative, *SyncML Architecture* http://www.syncml.org (Accessed: March 22, 2001).

[Sync. HTTP] SyncML Initiative, *SyncML HTTP Binding Version 1.0*, http://www.syncml.org/docs/syncml_http_v10_20001207.pdf (Accessed: January 9, 2001).

[Sync. OBEX] SyncML Initiative, *SyncML OBEX Binding Version 1.0*, http://www.syncml.org/docs/syncml_obex_v10_20001207.pdf (Accessed: January 9, 2001).

[Sync. Protocol] SyncML Initiative, *SyncML Synchronisation Protocol Specification Version 1.0*, http://www.syncml.org/docs/syncml_protocol_v10_20001207.pdf (Accessed: January 9, 2001).

[Sync. Rep.] SyncML Initiative, *SyncML Representation Protocol Specification Version 1.0*, http://www.syncml.org/docs/syncml_represent_v10_20001207.pdf (Accessed: January 9, 2001).

[Sync. Whitep.] SyncML Initiative, *SyncML Whitepaper Version 1.0* http://www.syncml.org/download/whitepaper.pdf (Accessed: January 31, 2001).

[Sync. WSP] SyncML Initiative, *SyncML WSP Binding Version 1.0*, http://www.syncml.org/docs/syncml_wsp_v10_20001207.pdf (Accessed: January 9, 2001).

[Synchrologic 1] Synchrologic Inc., *Synchronising Handhelds to Exchange & Notes Servers*, http://www.synchrologic.com/images/whitepapers/enterprise_mobile_computing.pdf (Accessed: February 9, 2001).

[Synchrologic 2] Synchrologic Inc., *The Future of Enterprise Mobile Computing*, http://www.synchrologic.com/images/whitepapers/enterprise_mobile_computing.pdf (Accessed: February 9, 2001).

[Synchrologic 3] Synchrologic Inc., *handheld applications guidebook*, http://www.synchrologic.com/images/whitepapers/handheld_applications_guidebook.pdf (Accessed: February 9, 2001).

[Synchrologic 4] Synchrologic Inc., *Decision Criteria for Synchronization and Replication Tools*, http://www.synchrologic.com/images/whitepapers/decision_criteria_main.html, (Accessed: February 13, 2001).

[TS 27.103] *3GPP TS 27.103 V3.1.0*, http://www.3gpp.org (Accessed: January 9, 2001)

[Versit 1] Versit Initiative, *vCalendar The Electronic Calendaring and Scheduling Exchange Format Version 1.0*, http://www.imc.org/pdi/Vcal-20.doc (Accessed: January 10, 2001).

[Versit 2] Versit Initiative, *vCard The Electronic Business Card Version 2.1*, http://www.imc.org/pdi/Vcard-21.doc (Accessed: January 10, 2001).

[WBXML] Wireless Application Protocol Forum, *WAP Binary XML Content Format, version 1.3, May 15, 2000*, http://www.wapforum.org (Accessed: January 9, 2001).

[Wirex] WireX Communications, Inc., *Parsimonious Server Security*, http://www.wirex.com (Accessed: February 19, 2001).

[WTLS] Wireless Application Protocol Forum, *WAP Wireless Transport Security Specification, version February 18, 2000*, http://www.wapforum.org (Accessed: January 9, 2001).

[Xiaodong] D. Xiaodong, D. Wagner, A. Perrig, *Practical Techniques for Searches on Encrypted Data*, ISAAC group. University of Berkeley, California, U.S. http://www.cs.berkeley.edu/~daw/papers/encsearch-oak00.ps (Accessed: March 14, 2001).

[xCard] SyncML Initiative, *xCard*, http://www.syncml.org (Accessed: March 31, 2001).

Appendix A

Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASP	Application Service Provider
CA	Certificate Authority
CAP	Calendar Access Protocol
CGI	Common Gateway Interface
CID	Content ID
CMS	Cryptographic Message Syntax
Corp.	Corporation
CPU	Central Processing Unit
DES	Digital Encryption Standard
DTD	Document Type Declaration
ECC	Elliptic Curve Cryptosystem
FAQ	Frequently Asked Questions
FPI	Formal Public Identifier
GUID	Globally Unique Identifier
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transport Protocol
IBM	International Business Machine Corporation
ID	Identity
IDEA	International Data Encryption Algorithm
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMAP	Internet Mail Access Protocol
IMC	Internet Mail Consortium
iMIP	iCalendar Message-Based Interoperability Protocol
Inc.	Incorporated
IP	Internet Protocol
IrDA	Infrared Data Association
IrMC	Infrared Mobile Communications
ISAAC	Internet Security Applications, Authentication, and Cryptography workgroup at University of California at Berkeley U.S.
iTIP	iCalendar Transport Interoperability Protocol
LUID	Locally Unique ID
ITU	International Telegraph Union
MD	Message Digest
MIME	Multimedia Mail Extensions
M.Sc.	Master of Science
OBEX	Object Exchange Protocol
OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
PIM	Personal Information Management
PKI	Public Key Infrastructure
PM	Post Meridiem (i.e. in the afternoon)
POP	Post Office Protocol
RDF	Resource Description Format
RFC	Request For Comments
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UID	Unique ID
URI	Uniformed Resource ID
URL	Uniformed Resource Locator
URN	Uniform Resource Name
U.S.	United States
UTC	Coordinated Universal Time
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WBXML	WAP Binary XML Content Format Specification
WDP	Wireless Datagram Protocol
WML	Wireless Mark-up Language
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
WTP	Wireless Transport Protocol
XAPIA	CSA X.400 Association's Calendaring and Scheduling API
XML	Extensible Mark-up Language
XOR	Extended OR