The HP Time Vault Service: Innovating the Way Confidential Information is Disclosed, at the Right Time

Marco Casassa Mont, Keith Harrison, Martin Sadler
Trusted E-Services Laboratory
HP Laboratories Bristol
HPL-2002-243
September 4th , 2002*

E-mail: marco_casassa-mont@hp.com, keith_harrison@hp.com, martin_sadler@hp.com

TIME VAULT, IBE, confidentiality, time, trusted system, privacy infrastructure

Digital information is increasingly more and more important to underpin interactions, transactions and exchanges of knowledge. On the other hand, leakages of sensitive digital information can spread rapidly with harmful effects for people, organizations and governments. This paper focuses on the problem of protecting confidential information from unauthorized disclosures, subject to time-based criteria: it is a common issue in the industry, government and day-to-day life. We introduce an innovative service, the Time Vault Service, that leverages the emerging Identity-based Encryption (IBE) cryptography schema to enforce time-based confidentiality of digital documents and simplify their distribution. We discuss the advantages of this approach against a similar approach based on traditional cryptography. A working prototype of the Time Vault Service is described, as proof of the concept.

* Internal Accession Date Only
Approved for External Publication

1. Introduction

In today's world digital information is more and more relevant for people, enterprises and institutions to underpin and enable interactions, transactions and exchanges of knowledge.

The Internet and the web boosted the process of representing, accessing and distributing information in a digital way. On one hand this is a very important step towards a more global and simpler way to distribute and access information. On the other hand digital information can be very sensitive and valuable to their owners or the entities that are affected by it: dealing with confidentiality and privacy issues is of primary importance.

It is common practice, in the ordinary life, to deal with aspects related to the confidentiality of information. People handle secrets and sensitive information on daily basis, including their personal profiles, their identity information, their financial details, etc. Enterprises define policies and mechanisms to enforce the correct management of confidential documents such as strategic business decisions, research activities and customers' information. Similarly, governments and social institutions address analogous issues when dealing with citizens' data and profiles.

It happens more and more frequently to hear about leakages or disclosures of sensitive digital information by unauthorised entities and the consequent negative impacts on business reputations, people's careers and financial markets. A recent case in the computer industry involved the unauthorised disclosure of confidential e-mails and voice-mails to mass media. The implications of this action have been severe, including fluctuations in company's stock price, disruptions in the working and business environment and legal disputes.

The paradigm shift from paper-based documents to digital documents has amplified the consequences that leakages of confidential information might have, because of the rapidity by which digital data can be propagated across the Internet and the large availability of tools enabling digital communication.

Information leakages can be due to people that fraudulently take advantage of their roles, their knowledge and the trust granted to them. Other leakages happen as a result of hacking activities or faults on IT systems and services. Others again are accidental, because of incompetence or inadequacy of IT systems and services.

Dealing with the confidentiality aspect of digital documents is a complex task. It has strong implication in term of security and privacy. In involves the management of disclosure conditions and constraints, access control, the satisfaction of trading and business policies along with their legal and legislative implications. In some cases it is purely a matter of trust: this implies relying on people to act in a trustworthy way. In other cases technology can be used to address part of the involved problems [1].

This paper describes the Time Vault Service, an innovative technical work done at HP Laboratories, Bristol, UK to enforce the confidentiality of digital documents until their intended disclosure time.

2. Addressed Problem

This paper addresses the problem of enforcing the confidentiality of digital documents according to predefined constraints (policies) and facilitating an efficient distribution of the content of these documents.

In general, any kind of constraint can be used to specify the conditions under which information can be disclosed. *We focus on a specific aspect: the time of disclosure of confidential information*.

In other words, the addressed problem involves two factors: a digital document to be kept confidential and its intended disclosure date and time.

This problem is quite common in the physical world and has personal, social and business implications. A few examples follow:

- In the enterprise and business environment, confidential documents (containing business analysis, strategic decisions, communications to employees, etc.) are generated for the consumption of managers, boards or working groups: quite often these documents can be disclosed to the employees and stakeholders only at well defined points of time, dictated by trading, business and legislative constraints.

- In the B2B and e-commerce environments (such as auctions, e-marketplaces, supply-chains) confidential information is exchanged to enable interactions or transactions. The involved parties might be prevented form accessing the exchanged information for predefined periods of time. For example, in blind auctions a market maker can only access and potentially disclose participants' bids at the end of the bidding time.

- In the ordinary life, for example, students know the content of their exams or their final marks only at the time dictated by local authorities or the department of education.

The period of confidentiality of digital documents can vary depending on the context. Some confidential documents need to be kept secret for short periods of time. In other cases secrets might need to be preserved for months or years (for example in case of top-secret documents, in the military environment).

There are people who know about the content of confidential information since it was generated. Other people will have access to this content only at well-defined points in time. For the former category of people, trust and accountability are fundamental requirements. When they deal with confidential material, on one hand they need to take the proper precautions to avoid unwanted leakages and, on the other hand, they must ensure that information is accessible and available to the interested parties once disclosed.

In a traditional scenario, confidential information is protected against the access of unauthorised people and disclosed to them only at a specific point in time, as shown in Figure 1a:

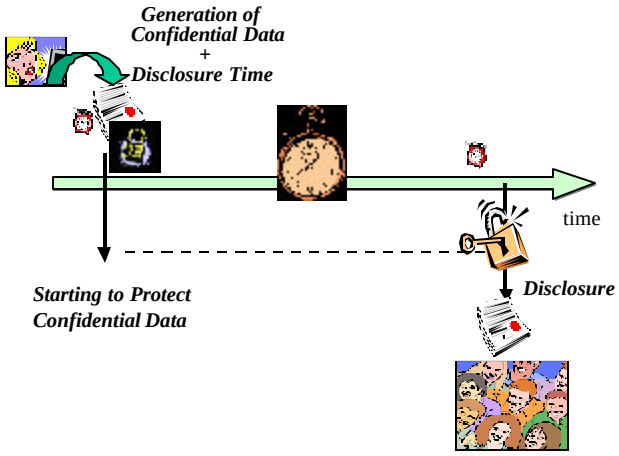

Figure 1a: Reference Scenario #1

In the above picture, a confidential document is generated by an entity (person, system, application, service, etc.) with a clear indication of its intended disclosure time. This document is protected (and kept secret) from unauthorised accesses till its intended disclosure time. Afterwards, it is distributed to the intended parties.

In the physical world confidential information is usually protected and stored in a secure place until it can be disclosed. Similarly, confidential digital information has to be protected and secured for its entire period of confidentiality.

An alternative scenario consists of distributing confidential documents to the intended recipients by making sure that their contents are unintelligible (obfuscated) until their disclosure time, as shown in figure 1b:

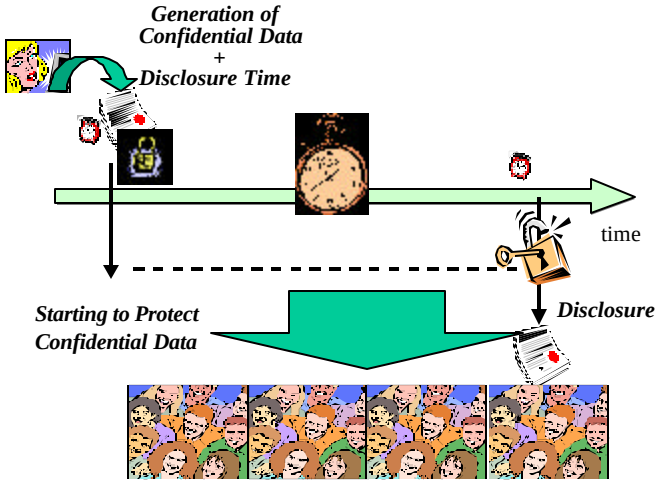

Figure 1b: Reference Scenario #2

This scenario is particularly interesting, as it allows a gradual distribution of digital information without compromising its confidentiality. It would simplify the life of people and administrators in many situations. For example in large organisations, the distribution of confidential information could be planned in advance, to avoid peaks of network traffic. Even for small group this could be convenient. In case of people that frequently travel (or are remotely located) and need to be kept up-to-date with the content of presentations or enterprise communications, they might download in advance large confidential presentations or documents and access their contents once their intended disclosure time comes. In the day-by-day life, people could receive in advance confidential information and documents and access their content afterwards, once approved for disclosure: this would apply in case of results of contests and lotteries (people will know if they are the winners only at a precise date and time), the result of school tests, people could receive in advance their renewed digital credentials and access them when the current ones are expired, etc.

In the above scenario, the additional problem is to allow people to access the content of the confidential documents (they received in advance), once the time is right for disclosure.

This paper mainly focuses on aspects related to this scenario although the proposed solution applies also for the scenario in Figure 1a.

In this paper we do not specifically address the problem of who can access confidential information once it has been disclosed: it is an orthogonal problem and it is of secondary importance if compared to the problem of keeping this information confidential until its intended disclosure time.

3. Requirements

In today's world it is not straightforward to enforce the confidentiality of digital information for predefined periods of time. Enterprises, governments or organisations usually define policies to address this problem but, at the very end, it is up to the individuals to understand and implement them. This usually happen by adopting ad-hoc solutions and specific tactics for each specific case. In some cases there is a lack of knowledge about how to achieve it and a lack of simple mechanisms and tools to address this problem.

The event of disclosing confidential information can generate large interest. Large masses of people might try to access this information at the same time. This could cause delays and frustration when the information is not immediately accessible (for example on the Internet or the Intranet) because of the high traffic and the inadequacy of the underlying distribution infrastructure. Planning in advance the way confidential data is made available to people (and systems), once disclosed, is fundamental. As anticipated in the previous chapter, in some cases it would help to distribute in advance digital confidential documents to the intended recipients, before their intended disclosure time, without disclosing their content.

Solutions dealing with the management of the time-based confidentiality problem should:

1. Provide a professional and accountable service. People (systems and applications) should have access to mechanisms, infrastructures and services that are run by competent and accountable entities, with the adequate levels of security, availability and assurance;

2. Provide mechanisms that strongly enforce the constraints on the disclosure time, associated to confidential documents. Specifically these mechanisms must not allow the access to the content of confidential information until their intended disclosure time;

3. Avoid bottlenecks: these solutions should provide mechanisms that allow the (gradual) distribution of confidential documents to the intended recipients with the assurance that these recipients will be able to access their content only at or after their intended disclosure time.

4. Be Simple: people should be able to deal with confidentiality aspects of information in an intuitive and straightforward way.

These requirements equally apply for solutions that are deployed in business, social and government environments.

Chapter 4 describes the current approaches that can be used to address the time-based confidentiality problem and highlights their weak points. Chapter 5 and 6 describes an alternative approach, invented at HP Labs, Bristol.

4. Current Approaches

Different mechanisms and technologies are currently available to enforce the confidentiality of digital documents for a predefined period of time:

- Usage of strong access control to protect the access to data;
- Encryption of confidential data;
- Hybrid models based on the above two models.

Access control mechanisms [2], [3], [4] allow the definition of which entities can access confidential information and what they can do with it. Access control can be modified over time.

The encryption of confidential data [5] adds further protection as the original content can only be accessed only if the proper decryption key is known. This has strong implications on protecting and storing decryption keys in a secure and safe way till the intended disclosure time.

Secure infrastructures and systems are necessary to underpin the correct working of both access control and encryption mechanisms, including secure storage systems.

Many access control products and solutions (including IBM/Tivoli, CA and Netegrity security products) are currently available within enterprises, e-commerce and web sites to locally protect the access to digital documents. Solutions based on access

control mechanisms can be used to make sure that only authorised people can access confidential documents by defining proper access control list. Of course administrators of these solutions must be accountable for running them in a convenient and professional way. Access control lists (ACLs) must be modified to extend the access to other people. Current Access Control systems, based on ACLs, do not involve the explicit management of time factors. On their own, access control mechanisms only provide and enforce barriers to the access of documents. If, for any reason, these barriers are defeated, the content of documents can be directly accessed.

Strong encryption mechanisms are more suitable to enforce the confidentiality of digital documents. They can be used to encrypt and obfuscate the content of a confidential document for a well-defined period of time. The intended disclosure date of a confidential document must in some way be associated to the correspondent encrypted document and protected against manipulations. Once a digital document is encrypted it can potentially be distributed to third parties. The correspondent decryption key is distributed to the intended recipients only after the intended disclosure time.

Traditional encryption mechanisms rely on public key cryptography [5] (including RSA cryptography), symmetric key cryptography [5] or hybrid combinations [6] of the two (such as enveloping techniques). These mechanisms require the generation of a decryption key at the encryption time. In case of public key cryptography, when a public key is created, the correspondent private key is generated as well, because of the way cryptographic algorithms work. In case of a symmetric key, this key is used both for encryption and decryption purposes.

Back to the problem of keeping a document confidential for a predefined period of time, when this document is encrypted, the correspondent decryption key must be protected for the whole period of confidentiality.

This has implications on keeping the decryption key secure and safe: it must "survive" until the correspondent documents can be disclosed. The involved risks and costs might increase with the length of the confidentiality period. The more digital documents need to be kept confidential for different periods of time the more different "secrets" need to be generated, protected and preserved. Management systems must be used to deal with the storage and the secure preservation of secrets along with their associations to the correspondent encrypted documents. Scalability and management of complexity are two key aspects that need to be addressed. In addition, running these management systems has strong implications in terms of accountability and liability.

Services can be built on top of cryptographic mechanisms to provide added value functionalities to users, store and protect secrets and be outsourced to expert and accountable service providers.

An example of such a service, built by using traditional cryptography, allows people to encrypt and distribute confidential documents, along with their intended disclosure time. The encryption mechanism is based on enveloping techniques, involving symmetric keys and a certified public key associated to the service (which has the correspondent private key). A confidential document is encrypted with a symmetric

key generated on-the-fly. The symmetric key is encrypted with the public key of the service. Encrypted documents can be distributed to third parties. A receiver of the document must interact with the service to obtain the correspondent decryption key (symmetric key): the service will reveal it only at or after the intended disclosure time of the document. A detailed description of the above service can be found in chapter 7.

Identity-based Encryption (IBE) schema [7], [8] an emerging cryptography schema, can be used to build services that are simpler and more efficient than the ones based on traditional cryptography (based on RSA public key cryptography, or similar, and symmetric key cryptography). The next chapter describes the properties of this schema.

Chapter 6 describe the *time vault service*, an innovative service built to enforce the time-based confidentiality of digital documents, based on the IBE technology.

Chapter 7 compares this service with a similar service built by using traditional cryptography.

5. IBE Approach

The ideal solution to the time-based confidentiality problem would be a cryptographic mechanism where "decryption keys" are generated only at the intended disclosure time of the encrypted documents: no secret would exist before that time. This would avoid additional cost in managing and storing secrets, as they would be generated only at the decryption time.

Unfortunately there is not such a kind of solution available today, as all known solutions directly or indirectly rely on one or more initial secrets (that must then be protected over time).

This should not prevent researchers from investigating alternative approaches that help to build simpler, easier to secure and more efficient solutions that the ones that can be built today by using traditional cryptography.

The Identity-based Encryption schema (IBE a.k.a. ID-PKC [7]) is an emerging encryption schema that can be successfully used to achieve these objectives. Next section briefly describes the IBE core properties and the IBE basic interaction model. Please read the *Appendix A* for more details about the QR IBE cryptographic schema.

5.1 IBE Cryptography Schema

The IBE cryptography schema has two basic properties:

- 1st Property: any kind of string can be used as an IBE encryption key (public key). This "string" can consist of any sequence of characters or bytes such as a text, a name, an e-mail address, a picture, a list of terms and conditions, etc.
  Information is encrypted by using a string along with a "public detail", uniquely associated to a specific trusted third party, referred in this paper as

trust authority. This trust authority is the only entity that can generate the correspondent IBE decryption key;

- 2nd Property: the generation of an IBE decryption key (associated to an IBE encryption key, i.e. a string) can be postponed in time. In other words an IBE decryption key can be generated (by a trust authority) a long time after the correspondent IBE encryption key was created.

Figure 2 shows the details about the (three corners-based) IBE interaction model:

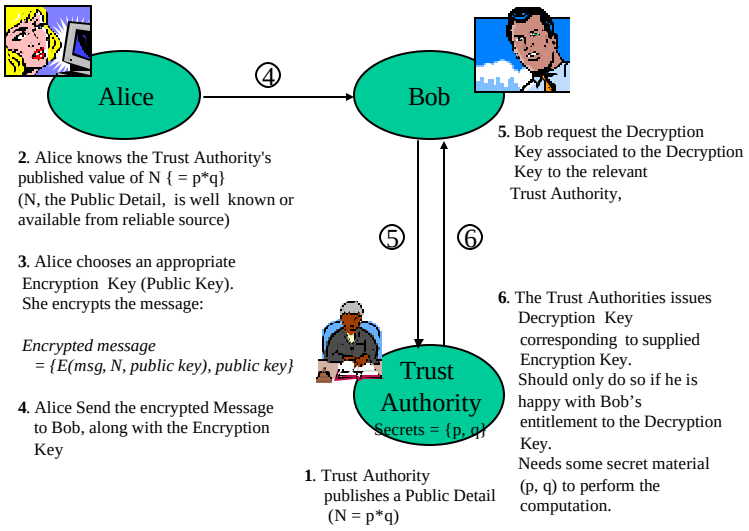

Figure 2: High-level IBE Interaction Model

Three players are involved in the above interaction model: a sender of an encrypted message (Alice), the receiver of the encrypted message (Bob) and a trust authority in charge of issuing decryption keys.

Alice wants to send an encrypted message to Bob. Alice and Bob trust a third party, the trust authority (TA). The following steps take place:

1. During the TA's initialisation phase, the TA generates a secret (stored and protected at the TA site) and a correspondent "public detail" that is made publicly available.

2. Alice trusts the TA. She retrieves the public detail from the TA site;

3. Alice wants to send a message to Bob. She defines an appropriate IBE encryption key (public key) to encrypt this message. The IBE encryption key can be any type of string, for example Bob's e-mail address. Alice's message is encrypted by making use of this IBE encryption key and the TA's public detail.

4. Alice sends the encrypted message to Bob, along with the IBE encryption key she used to encrypt the message.

5. Bob needs the decryption key associated to the above IBE encryption key, to decrypt Alice's message. Bob has to interact with the trust authority. He might have to provide additional information (credentials) to prove he is the legitimate receiver of the message.

6. The trust authority generates and issues to Bob the IBE decryption key (associated to the IBE encryption key chosen by Alice) if it is satisfied by Bob's "credentials". The trust authority might decide to generate the IBE decryption key depending on the fulfilment of specific constraints as specified by the correspondent IBE encryption key. For example a trust authority might issue an IBE decryption key to Bob only if he is compliant with a well-defined list of terms and conditions. Please notice that the IBE public key (i.e. a string), used to encrypt the document, would directly specify the list of these terms and conditions.

5.2 Leveraging the IBE schema

A service can be built to address the time-based confidentiality problem by leveraging the two core IBE properties:

- Any string can be used as an IBE encryption key (public key). Specifically, this string could be the date and time of disclosure of the confidential document. For example, the string **"GMT200301011200"** can be used to encrypt a document and specify that its disclosure date is on January, 1st, 2003, at 12:00 noon (GMT).

- An IBE decryption key can be generated after the creation of the correspondent IBE encryption key. Specifically this decryption key can be generated by a trust authority exactly at the time of the intended disclosure of the document. The IBE encryption key itself specifies the intended disclosure time, as described above.

The entity that provides the above service runs a trust authority in charge of generating the IBE decryption keys. Of course it has to create, store and protect the trust authority's secret necessary to generate all the subsequent decryption keys. Nevertheless, this is the only secret (independent by the number of decryption keys that are generated from it) that needs to be preserved and security efforts can be concentrated to protect it.

The IBE encryption keys used to specify time-based constraints are intuitive strings, intelligible both to humans and computers. They are self-contained: they are used for encryption purposes and, at the same time, define the constraints under which IBE decryption keys can be generated.

The problem of storing and protecting the corresponding IBE decryption keys does not exist, as they are physically generated by trust authorities only when they are required.

Chapter 6 describe in more details this service, named "The Time Vault Service".

Chapter 7 compares this service with a correspondent service based on traditional cryptography and highlights its advantages.

6. The Time Vault Service

The "Time Vault Service" is an innovative service based on IBE cryptography, designed to address the time-based confidentiality problem. It can be used to encrypt sensitive information and ensure that the correspondent decryption keys are issued only at its disclosure time.

Figure 3 shows the high-level aspects of this service:

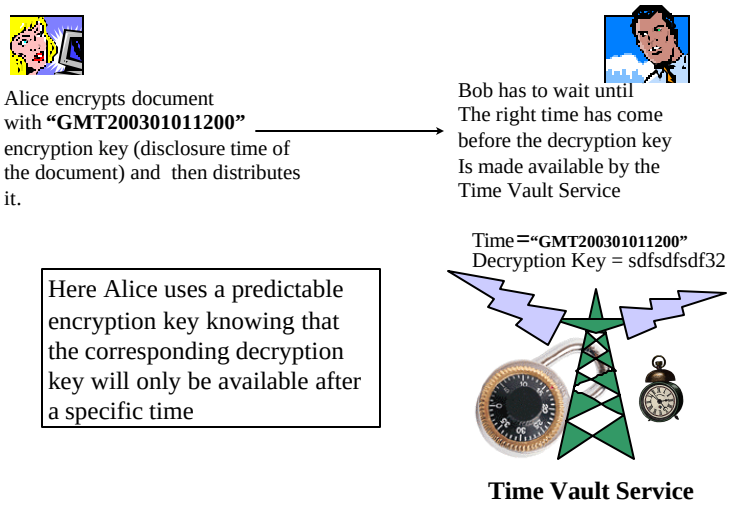

Figure 3: Time Vault Service: key aspects

Alice has a confidential document. She wants to send it to Bob but at the same time, she wants to keep its content confidential till a predefined date and time, for example January 1st, 2003, 12:00 noon (GMT). She uses the time vault service, a service run by an accountable trusted third party. She has to encrypt her document with a string, an IBE encryption key. This string contains the date and time of disclosure of the document (for example **"GMT200301011200"**, i.e. January 1st, 2003 – 12:00 noon (GMT)). To achieve this she also need to use the IBE public detail associated to the time vault service (the time vault service is providing the functionality of a trust authority). She sends the encrypted document to Bob, along with the expected disclosure time (IBE encryption key).

The time vault service continually generates and publishes IBE decryption keys (given a predefined frequency) associated to the current date and time (obtained by a trusted clock). *Please notice that the current date and time is interpreted as if it was an IBE encryption key.* For example, the time vault service could have been configured to generate an IBE decryption key every hour. If the current time is, for instance, August 8th, 2002, 13:15 (GMT), the time vault service will generate an IBE decryption key at 14:00, correspondent to the **"GMT200208081400"** IBE encryption key.

It did exactly the same thing at 13:00, by generating an IBE decryption key associated to the **"GMT 2002080813:00"** IBE encryption key and so on. *The time vault service is completely unaware of the usage that people make of the IBE decryption keys it generates.*

The IBE decryption key that Bob needs to decrypt Alice's document will be generated by the time vault service only at the date and time specified by Alice's IBE encryption key. Nevertheless, because of the above property, the time vault service does not need to know about the existence of Alice's IBE encryption key and the fact that Bob is waiting for the correspondent decryption key.

When Alice generates her time-based IBE encryption keys, she must use the same (string) format adopted by the time vault service for its time-based IBE encryption keys (used to generate the correspondent IBE decryption keys).

In the specific example, Bob has to wait until January 1st, 2003, 12:00 noon (GMT) to obtain the decryption key necessary to decrypt Alice's document: only at that time, the time vault service (independently by Bob's needs) will generate the required IBE decryption key.

With this simple service it is possible to encrypt a confidential document just by specifying its disclosure time as an IBE encryption key. The encrypted document can then be distributed to the intended receivers. These receivers rely on the time vault service for generating the decryption keys at the right time.

Figure 4 shows more details about the architecture of the time vault service:

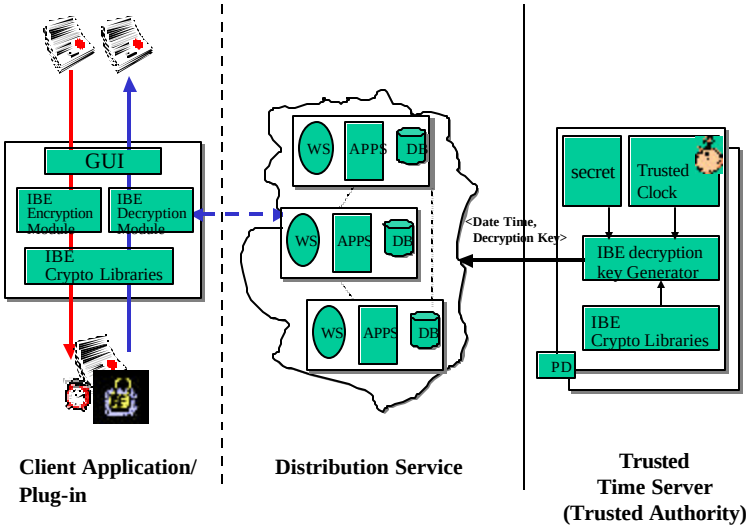

Figure 4: Time Vault Service: High Level Architecture

This architecture consists of three main components:

- **Trusted Time Server**: it is based on an IBE trust authority. A public detail is initially created by this server and publicly made available. The correspondent secret is locally stored and secured. The purpose of the time server is to *continually* issue IBE decryption keys correspondent to the current date and time, independently by the usage that third parties are going to make of it. Specifically, the frequency of issuances of these decryption keys depends on a predefined granularity, such as every minute or every hour or every day.
  The current date and time is considered by the time server as an IBE encryption key (public key). The time server issues decryption keys correspondent to these IBE encryption keys, based on a date and a time. For example if the current time is 2002-08-08 12:00 GMT, the trust server will issue an IBE decryption key associated to the "GMT200208081200" encryption key (assuming that this is the string format used to represent the current date time and it is used as an IBE encryption key).
  The time server is built in a way that it cannot generate decryption keys that correspond to points of time greater than the current time. Once IBE decryption keys are generated, they are publicly disclosed and published on an external distribution service along with their correspondent IBE encryption keys (i.e. strings containing a date and time).
  The time server makes use of a reliable and secure clock, for example an atomic clock. In general the time server is built with secure and fault tolerant technology: it is run in a protected environment inaccessible from the external world. It is important to notice that the time server is a self-contained component. It is a robust component that does its job independently of how the decryption keys it generates are going to be used (if they are). Because of this aspect, it is possible to minimise the interactions the time server has with the external world (basically it only interacts with the distribution service with an outgoing interaction, to publish key pairs), strongly protect and secure it.

- **Distribution Service**: the distribution service is an Internet portal specialised in publishing IBE decryption keys (along with the correspondent time-based IBE encryption keys) issued by the time server. Basically the distribution service stores, indexes and publishes <IBE encryption key, IBE decryption key> pairs, where the first component is a formatted date-time IBE encryption key and the second component is the correspondent IBE decryption key as generated by the time server.
  Users can access this information either by means of traditional browsers or programmatically, by querying the portal with a date and time (IBE encryption key) and obtaining back the correspondent IBE decryption key (if currently available).
  For example, if a person wants to know what is the IBE decryption key associated to the "GMT200208081200" encryption key, they will use this string to query the distribution service. The distribution service will return the correspondent decryption key if it has been issued by the time server, otherwise an error message will be thrown.
  The distribution service can be implemented as a fairly traditional Internet portal including a web server, front-end scripts (such as CGI scripts, servlets, etc.) and back-end applications and databases. These resources might be replicated across multiple web servers for load balancing. Similarly, the stored information (<IBE encryption key, IBE decryption key> pairs) can be

replicated across distributed databases which are securely run in the back-end of the portal infrastructure.
The distribution service has to be reasonably protected against denial of service attacks and attempts to modify or destroy its information. Nevertheless if any part of this information is destroyed or lost, it can be recreated by the time server, up to the current date and time.

- **Client Application**: it is the application installed at the client's site (it might be a plug-in installed in traditional web browsers) that allows people to encrypt documents to be kept confidential and decrypt them only at their intended disclosure time.
  The client application contains IBE encryption and decryption modules and it knows the IBE public detail associated to the time server.
  A user that has a sensitive document to be kept confidential till a well-defined date and time can encrypt this document by simply specifying the date and time of its intended disclosure. The client application transparently derives an IBE encryption key (string) from this date and time, encrypts the document with it and adds the necessary metadata to the encrypted document (such as the IBE encryption key, i.e. the date and time of its intended disclosure). The client application does not need to interact with the external world or be online to achieve this. Everything can be done on a standalone system.
  The user can then distribute the encrypted document to whoever he/she think it might be of interest, by using any digital communication mechanism, such as e-mail, web site, newsgroup, etc.

  The receiver of the encrypted document must have installed the same client application. By using this application he/she can try to decrypt the received document. The client application interprets the metadata associated to the encrypted document, retrieves from it the associated IBE encryption key (date and time of disclosure of the document) and interacts (on-line) with the distribution service to fetch the correspondent IBE decryption key, if it has been published by the time server. In case the decryption key is available, the client application decrypts the document. In case the decryption key is not yet available, the client application warns the user about the date and time by which the decryption key will be generated and published.

  The client application can be downloaded by a trusted site, such as the time server site or the distribution service portal.

The time vault service provides a simple service that:

- enables people to encrypt confidential documents by specifying the date time of their disclosure;

- ensures that decryption keys are issued by the time server (and published by the distribution service) only in correspondence of the current date and time, given a predefined frequency.

The time vault service satisfies the requirements described in chapter 3.

- It can be outsourced and run by a trusted third party. This service provider is accountable for the management of the time server's secret and its correct functioning;

- It is simple to use. Decisions about the confidentiality of documents and their disclosure time still need to be made by people. Nevertheless, after this decision is made, the client application transparently encrypts these documents depending on the specified disclosure time. It also transparently retrieves the correspondent decryption key from the distribution service, if it is available, or provides useful information to the user about when the decryption key will be made available;

- It is simple to run. Its components are self-contained. Security efforts can be concentrated on the core component, the time server: because of how it works, it is possible to minimise security threats and vulnerabilities;

- It enables the distribution of encrypted digital documents by preserving efficiency during the decryption phase. Once a document has been encrypted, it can be distributed to the intended receivers or published. People can receive it by e-mail or download it (in their encrypted form) in advance, before its intended disclosure time.
  Documents might be very large in size and it might take time and resources for a complete distribution to the intended audience. The time vault service allows for the planning of distribution of confidential information in a balanced way, to avoid peaks and excessive traffic. For example confidential decisions, reports, presentations and results, can be progressively distributed to the employees of large companies in their encrypted format, before their content is actually disclosed.
  Once the disclosure time of a document comes, people (by means of their client applications) can access the distribution service to retrieve the decryption keys. Large number of people could try to do this at the same time but they only need to download a few hundred bytes, corresponding to the desired IBE decryption key. The traffic generated (in term of transferred bytes) is orders of magnitude smaller than the case where whole documents need to be downloaded once their content is publicly disclosed.

Next section briefly describes a prototype of the time vault service. Chapter 7 compares this service (based on IBE cryptography) with a similar service based on traditional cryptography (such as RSA cryptography) and highlights its advantages.

6.1 Prototype

A simple fully working prototype of the Time Vault Service has been implemented, as a proof of concept.

Figure 5 shows the main components of this prototype:

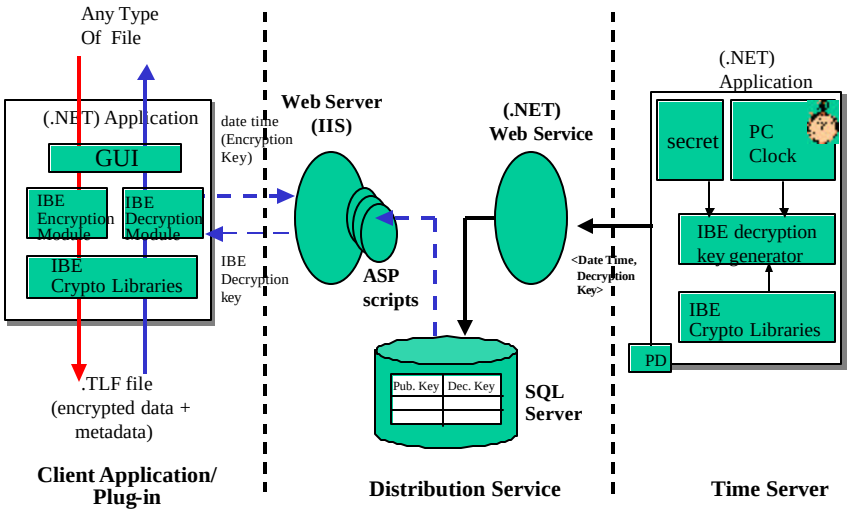

Figure 5: Prototype: The Time Vault Service

This prototype has been implemented by using the Microsoft .NET framework – release version [9].

The **time server** has been implemented as a .NET application, running on a PC and getting the current date and time information from the local clock (this is fine considering the fact that it is a prototype. In a real implementation, this clock must be reliable, trusted and secured). It uses IBE libraries developed in C (for performance reasons). The IBE secret (associated to the time server) is generated at the initialisation serialised and locally stored in a protected XML file. Similarly the IBE public detail (associated to the time server) is generated at the initialisation time and then locally stored in a XML file.

Figure 6 shows the a GUI console used to monitor the activity of the time server:

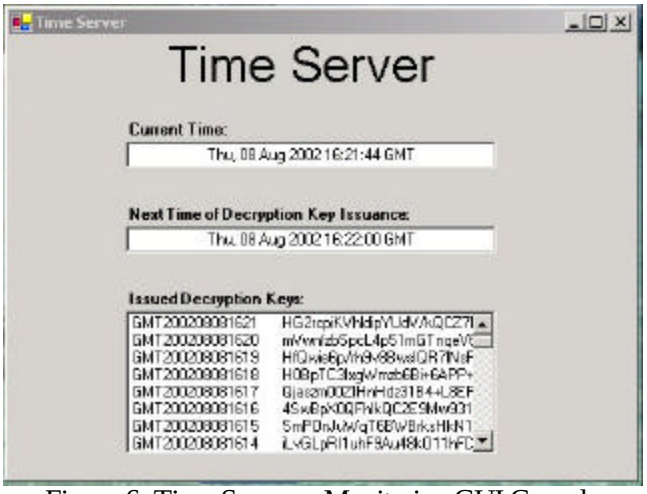

Figure 6: Time Server – Monitoring GUI Console

The above figure shows the list of IBE decryption keys generated by the time server so far, along with the correspondent IBE encryption keys (for example "GMT200208081621"). For demo purposes the frequency of issuance of IBE decryption keys was set to 1 minute.

The **distribution service** is build on top of a Microsoft IIS web server. It consists of an ASP script, a web service and a SQL server. It runs on a second PC. The web service exposes a method to publish a key pair (<date and time of disclosure (i.e. IBE encryption key), IBE decryption key>) in a table of the local Microsoft SQL server. It is deployed in the distribution service back-end and it can only be accessed by the time server. The time server remotely invokes its method every time it generates a new IBE decryption key. The ASP script, accessible through the IIS web server, allows remote users to query the SQL database for a specific date and time. If successful, the ASP script returns an XML file containing the correspondent IBE decryption key.

Figure 7 shows the GUI console used to monitor the IBE encryption and decryption keys stored in the distribution service:

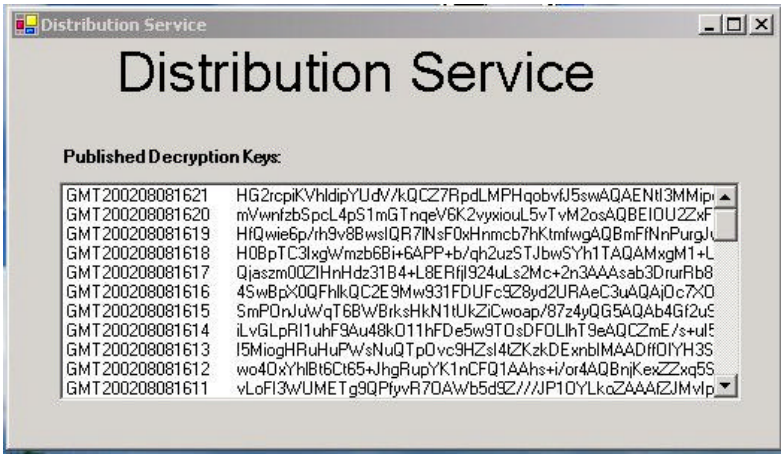

Figure 7: Distribution Service – Monitoring GUI Console

The **client application** is a standalone .NET application, containing IBE encryption and decryption modules, developed in C (for performance reasons). The functionalities of the client application are exactly the ones described in the previous chapter. Any kind of document and file can be encrypted by this application: the encrypted file (.tlf file – time locked file) contains metadata (a header) which includes the IBE encryption string (date and time of disclosure) used to encrypt the document.
The client application programmatically interacts with the distribution service by means of the HTTP protocol (the URL of the distribution service in contained in a local configuration file).

Figure 8 shows the client application:

Figure 8: Client Application

Figure 9 shows an example where a file (*IBEDemo.ppt*) is encrypted with a disclosure date and time (IBE encryption key) specified by the user, specifically January, 1st 2003 – 12:00 noon (**"GMT200301011200"**):

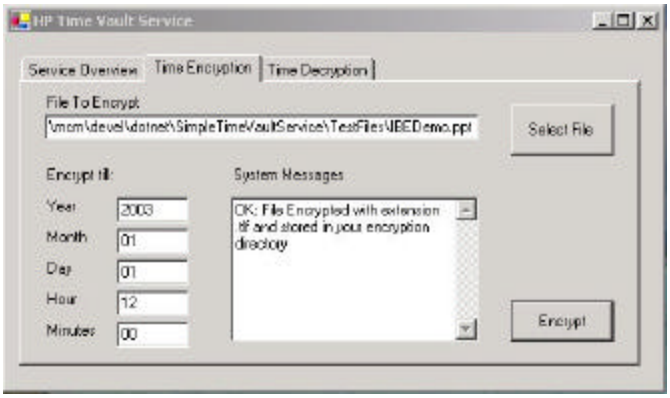

Figure 9: Example: Time-based Encryption

The user simply specifies the name of the file and the date and time of disclosure. The encrypted .tlf file is automatically generated along with the associated metadata.

Figure 10 shows an attempt of decrypting the file before its disclosure time:

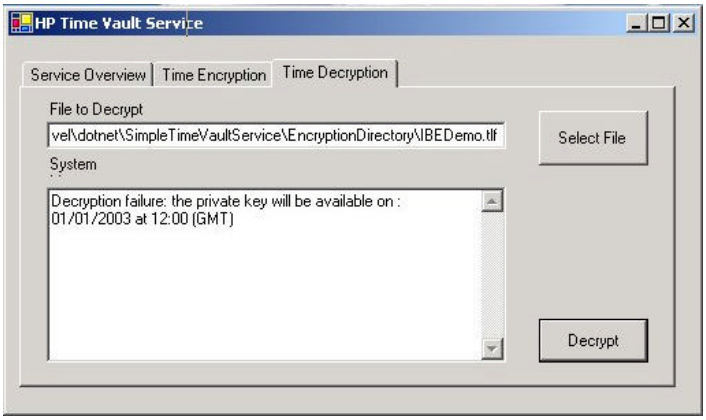

Figure 10: Example: Failure in attempting to decrypt a file before its disclosure time

In the above example, the client application notifies the user that the IBE decryption key is not yet available: it will be issued on January, 1st 2003 at 12:00 noon.

The above components have been deployed on different PCs and configured to interact by means of standard protocols such as SOAP and HTTP.

7. Discussion

The time vault service described in this paper relies on IBE cryptography to provide its functionality. A similar service can be implemented by making use of traditional cryptography. The aim of this chapter is to discuss and compare the two approaches.

Figure 11 shows the interactions between a sender of confidential information, a receiver and the time vault service, implemented by using IBE cryptography:

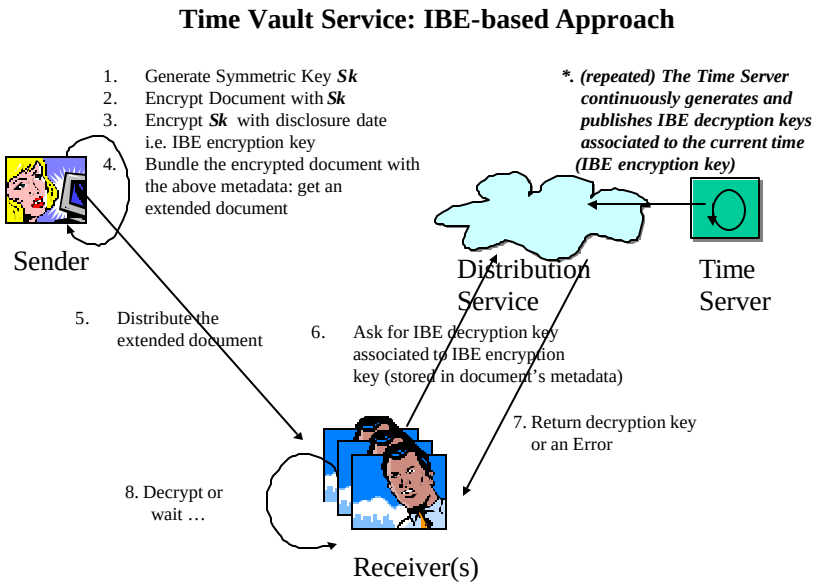

Figure 11: Time Vault Service: IBE-based approach

Let us assume that Alice, the sender, wants to send a confidential document to Bob, and make sure that Bob can access its content only after a specific date and time.

The following steps take place:

*. *Repeated step*: the time server continuously generate and publishes IBE decryption keys associated to the current date and time (i.e. an IBE encryption key);

1. The client application, on Alice's PC, generates a symmetric key, ***Sk*** (purely for efficiency reasons. The usage of symmetric keys can be avoided as the IBE encryption key can be used directly, but at a greater cost in compute time);

2. The client application encrypts the document with the symmetric key, ***Sk***;

3. The client application encrypts the symmetric key, ***Sk***, with the date and time of disclosure of the document, i.e. the IBE encryption key;

4. The client application bundles together the encrypted document, the encrypted symmetric key and the IBE encryption key: it creates an extended document;

5. Alice sends the extended document to the receiver, Bob;

6. The client application, on Bob's PC, retrieves the IBE encryption key from the extended document and notifies Bob about the date and time by which the correspondent IBE decryption key will be issued. Bob decides to ask the time server for this key. The client application connects to the distribution service asking for the IBE decryption key associated to the IBE decryption key;

7. The distribution service looks for the decryption key in its database. If the decryption key is available (i.e. the time server has generated and published it) it will return it to Bob, otherwise an error message is thrown;

8. The client application, on Bob's PC, decrypts the message if the correct IBE decryption key is available or notifies Bob to wait till the disclosure time.

A similar service can be implemented by using traditional (RSA-based) cryptography. Figure 12 shows the interactions between a sender of confidential information, a receiver and the "correspondent" time vault service, build with RSA public key and symmetric key cryptography:

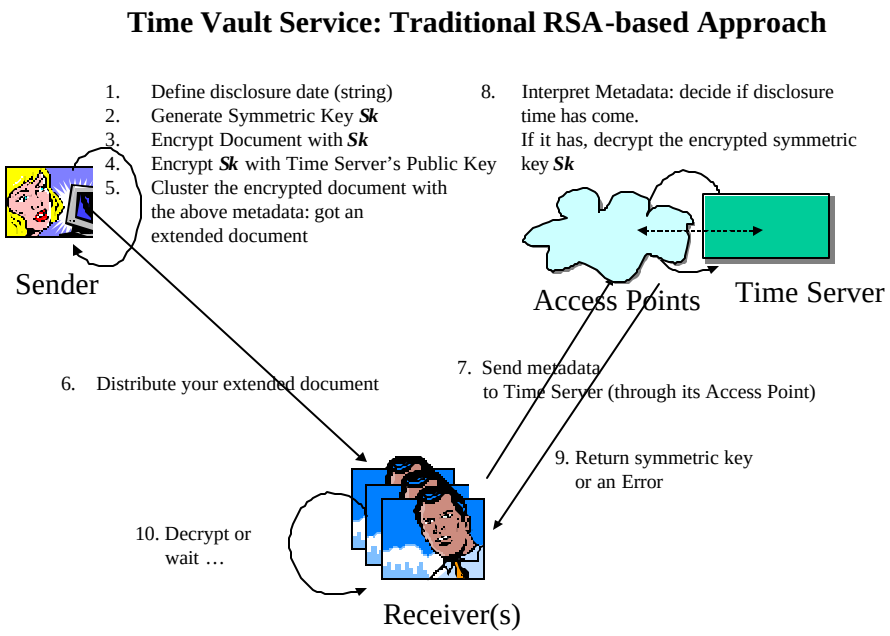

Figure 12: Time Vault Service: RSA-based approach

In this case, we assume that the time server has a private key (that is kept secret) and an associated PKI X.509 identity certificate, that is publicly made available. The following steps occur:

1. Alice defines the intended disclosure time of a document, i.e. a string;

2. The client application, on Alice's PC, generates a symmetric key, ***Sk***;

3. The client application encrypts the document with the symmetric key, ***Sk***;

4. The client application encrypts the symmetric key, ***Sk***, and the disclosure time with the time server's public key, contained in its PKI X.509 identity certificate.

5. The client application bundles together the encrypted document, the encrypted symmetric key (and the disclosure time) and clear text including the string representing the disclosure time: it creates an extended document. Specifically, with the term "metadata" we mean the bundle containing the encrypted symmetric key and the string containing the disclosure time;

6. Alice sends the extended document to the receiver, Bob;

7. The client application, on Bob's PC, retrieves the metadata from the extended document and notifies Bob about the date and time of disclosure of the document. Bob decides to ask the time server for the symmetric key, necessary to decrypt the document. The client application connects to the time server's access service and passes to it the document's metadata (date and time of disclosure and encrypted symmetric key);

8. The access service interprets the metadata: it retrieves the string containing the date and time of disclosure. If the current time is greater than the disclosure time, it asks for the decryption key to the time server in order to decrypt the encrypted symmetric key (the time server uses the private key associated to its public key published in its X.509 identity certificate). *Please notice that the time server should double check that the intended disclosure time encrypted in the metadata (as originally specified by Alice) is really antecedent to the current time*;

9. In case of success, the symmetric key is sent back to Bob. Otherwise an error message is thrown to Bob;

10. The client application, on Bob's PC, decrypts the message if the symmetric key is available or notifies Bob to wait till the disclosure time.

Both models rely on an accountable and professional trusted third party to run the service and promptly issue decryption keys at the right time. In both models the time server has a secret (either an IBE secret or a PKI private key) to be protected, as it is the core component used to generate decryption keys.

Nevertheless the IBE-based model has a clear advantage on the other model, in term of security, simplicity and efficiency.

In the model based on traditional cryptography, every confidential document can only be decrypted by heavily interacting with the time server. The access point, associated to the (RSA-based) time server, has to interpret documents' metadata to verify if the constraint on the associated disclosure time is satisfied. In such a case, the time server needs to decrypt the metadata (as it contains the symmetric key used for encryption). This needs to happen as the time server is the only entity that has access to its private key (for security reasons). Both the time server and its access point(s) might have to deal with peaks of users' requests and the time server is directly involved in the loop, every time a decryption activity need to take place. This might create delays and introduce potential vulnerabilities. Caching mechanisms and replication can be put in place to mitigate part of the problem but this increases the complexity of the overall service.

The time server based on the IBE cryptography model is compartmentalised and standalone: its activity is completely independent by users' requests for decryption keys. The only allowed interaction is with the distribution service by means of an outgoing connection. Because of these two aspects it is easier to protect this time server than the one based on traditional cryptography.

The service based on the IBE cryptography model is simpler to run than the other model as it is more modular, each component is self contained and interactions about these components are trivial.

It is also potentially more efficient. In the IBE-based model, users' interactions with the distribution service do not trigger any complex or heavy cryptography computation, as instead it happens for the other model. Even in case of peaks of requests, the distribution service only need to execute simple database queries and return a few hundred bytes to the client (the size of the decryption key). The potential overhead of this model is due to the fact that the time server has to continuously generate IBE decryption keys, even if they are not required. Nevertheless, this activity exclusively depends on a predefined frequency of issuance of decryption keys: it can be addressed during the configuration phase of the time server.

In term of extensibility, the IBE-based model can be easily used in other contexts, where a broadcasting model applies. For example, the time server's capability of generating IBE decryption keys based on the current time could be coupled with radio time-signal services (such as [10]) currently operated in many countries: decryptions keys could be distributed along with the radio signal to any appliance or device able to understand and interpret the signal.

8. Conclusions

The management of confidential documents is an important issue in the business and government environments and in the day-by-day life.

In this paper we focused on the time-based aspect on confidentiality i.e. the case where confidential digital documents can only be disclosed after a well-defined date and time.

We considered two categories of related scenarios: a first category, where confidential documents are locally protected and distributed to the intended receivers only at their disclosure time. A second category, where confidential documents are potentially distributed in advance but their content is made intelligible only at their disclosure time.

Common approaches to the addressed problem are based on access control and traditional cryptography mechanisms (including RSA cryptography) along with the provision of solutions and services that allow the access to confidential documents or the disclosure of decryption keys only at their disclosure time.

These solutions and services need to be run by professional and accountable entities, including trusted third parties.

We introduced and innovative service, the time vault service, based on the emerging IBE cryptography schema. Simple strings can be used to describe the date and time of disclosure of confidential documents: these strings are also used to encrypt these documents. Users can encrypt confidential document off-line, without interacting with the time vault service. The decryption key is unknown at the encryption time. Encrypted documents can be distributed to the intended receivers. The time vault service will issue the correspondent decryption keys only at the right time.

We compared the time vault service with a correspondent service built with traditional cryptography: we showed that the service built by leveraging IBE technology is simpler, more efficient and easier to protect. A prototype of the time vault service has been implemented and described in this paper as a proof of concept.

9. Acknowledgement

We would like to thank David Soldera for designing and implementing the core IBE cryptographic modules used to build the Time Vault Service prototype. A special thank also to Pete Bramhall for his feedback and comments.

10. References

[1] F. Gallegos, D. P. Manson, S. Allen-Senft – *Information Technology Control and Audit*. Auerbach – 1999

[2] R. S. Sandhu, P. Samarati - *Access Control: Principles and Practice, IEEE Communications Magazine*. pp. 40-48 - September 1994

[3] D.D. Clark and D.R. Wilson - *A Comparison of Commercial and Military Computer Security Policies*. In IEEE Symposium on Computer Security and Privacy - April 1987.

[4] D. Ferraiolo, R. Kuhn – *Role-based Access Control.* NIST - 1992

[5] W. Diffie, M.E. Hellman – *New Directions in Cryptography* – 1976

[6] RSA Laboratories – *PKCS#7: Cryptographic Message Syntax Standard*. Version 1.5 – 1993

[7] C. Cocks - *An Identity Based Encryption Scheme based on Quadratic Residues*. Communications-Electronics Security Group (CESG), UK. http://www.cesg.gov.uk/technology/id-pkc/media/ciren.pdf - 2001

[8] D. Boneh, M. Franklin – *Identity-based Encryption from the Weil Pairing*. Crypto 2001 – 2001

[9] Microsoft – *Microsoft .NET framework*. http://www.microsoft.com/net - 2002

[10] National Physical Laboratory - *The time signal: PIPS service*. http://www.npl.co.uk, UK - 2002

Appendix A: An Introduction to the QR IBE Scheme

In this appendix we explain how the quadratic residuosity (QR) Identity-based Encryption scheme works. We accomplish this by giving an annotated description of the various algorithms. For a more formal description, see [7].

A.1. The Interaction Model

There are three players in the interaction model - traditionally named Alice, Bob and Trent.

Alice is trying to get a single bit of information to Bob in such a way that anyone else that receives the message cannot understand it. That is, Alice is going to encrypt the message in such a way that only Bob can decrypt it.

Alice does not necessarily trust Bob but she does trust Trent. She is going to trust Trent to give the decryption key to Bob if and only if Trent is satisfied that Bob is entitled to it.

A.2. Initialisation

Trent must be initialized before Alice can encrypt the message. The following example shows an instance of the required initialization steps:

1. Choose a prime p such that p = 3 mod 4
2. Choose a different prime q such that q = 3 mod 4
3. Compute N = p*q
4. Keep p and q secret
5. Publish N

Please notice that:

- We assume that both p and q are big numbers, say 150 decimal digits long. It is important that these random primes are generated by a cryptographically strong random number generator.
- We assume that both Alice and Bob know the value of N (but not p and q)

A.3. Encryption

We assume that Alice wishes to encrypt a single bit, m, so that only Bob can decrypt it. We also assume that Alice knows the public value, N, corresponding to the chosen Trent and that Alice has chosen an encryption key string that is acceptable to Trent, for example, Bob's e-mail address Bob@trent.com.

Alice goes through the following steps:

1. Let r = 2*m – 1. i.e. r = -1 if m = 0 and r = 1 if m = 1.
2. Choose a random number, t, in the range 1 .. N-1 such that jacobi(t, N) = r
3. Compute h = hash("Bob@trent.com")
4. Compute s = (t + h/t) mod N
5. Send s and "Bob@trent.com" to Bob.

Please notice that:

- The hash function converts the string into an integer in the range 1 .. N-1. This hash function is required to return a value, h, that satisfies jacobi(h, N) = 1. The hash function should have the additional property that it is difficult to choose a string that hashes to a given value of h.
- We assume that t is generated by a cryptographically secure random number and is in the range 1..N-1. We also assume that Alice keeps t secret.
- The jacobi function, jacobi(a,b) is defined to return 1 if the equation x^2 = a mod b has a solution and –1 otherwise. This function may be efficiently computed.

A.4. Decryption

Assume that Bob has just received the encrypted message, s, from Alice. Alice will also have told him which Trent she used, and the string "Bob@trent.com" that she used as the encryption string. Bob goes through the following steps:

1. Get the decrypt key, b, corresponding to "Bob@trent.com", from Trent.
2. Compute m = jacobi(s + 2*b, N)
3. msg = (m + 1) / 2 i.e. msg = 0 if m = -1 and msg = 1 if m = 1.

A.5. Conversion of the Encryption String to the Decryption Key

Trent's role is to be someone that Alice can trust to issue a decryption key corresponding to a supplied encryption string to the right person.

Trent computes:

1. Compute h = hash("Bob@trent.com")
2. Compute b = sqrt(h) mod N
3. Send "b" to Bob, only if Trent is convinced that the person claiming to be Bob really is the Bob that Alice specified.

It is outside the scope of this model to cover how "b" is securely conveyed to Bob. However, there are many traditional encryption schemas that can be used to maintain the necessary confidentiality and integrity. In that Bob will already be known by Trent, they will have previously agreed the mechanism to use for this and will have the necessary local capabilities and support infrastructure in place.

Please notice that:

- The hash function is the same hash function as used by Alice when encrypting. Indeed, the value of h computed here is the same as the value Alice would have computed.
- In practice it is not always possible to compute b. In this case Trent should compute b = sqrt(-h) mod N instead and Alice should compute s = (t – h*x) mod N. If Alice does not know whether Bob is "positive" or "negative" then she should compute both values of s (using different random values of t) and send both values to Bob. It is up to Bob to choose the correct value of s to decrypt.