The HP Time Vault Service: Exploiting IBE for Timed Release of Confidential Information

Marco Casassa Mont
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol, BS34 8QZ, UK
+44-117-3128794
marco_casassa-mont@hp.com

Keith Harrison
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol, BS34 8QZ, UK
+44-117-3128083
keith_harrison@hp.com

Martin Sadler
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol, BS34 8QZ, UK
+44-117-3128294
martin_sadler@hp.com

ABSTRACT

Digital information is increasingly more and more important to enable interactions and transactions on the Internet. On the other hand, leakages of sensitive information can have harmful effects for people, enterprises and governments.

This paper focuses on the problems of dealing with timed release of confidential information and simplifying its access once public: it is a common issue in the industry, government and day-to-day life.

We introduce the "HP Time Vault Service", based on the emerging Identifier-based Encryption (IBE) cryptography schema. IBE (public) encryption keys specify the disclosure time. These keys are used to encrypt confidential information. An independent time server generates and publishes IBE decryption keys correspondent to the current time, at predefined intervals.

We discuss the advantages of this approach against current approaches based on traditional cryptography. A web-service based prototype is described, as a proof of concept.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption – *public key cryptosystems*

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection – *Unauthorized access*

K.4.4 [**Computers and Society**]: Electronic Commerce - *Security*

General Terms

Security, Algorithms, Design, Management, Experimentation.

Keywords

Privacy, Security, Identifier-based Encryption, Timed-Release, Disclosure Policies, Web Service

1. INTRODUCTION

Digital information is more and more relevant for people, enterprises and institutions to enable interactions, transactions and exchange of knowledge. Digital information can be very sensitive and valuable to its owners and the entities that are affected by it: addressing the privacy and confidentiality issues is of primary importance.

WWW 2003, May 20-24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

It is common practice, in 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 their profiles.

The Internet and the web boosted and simplified the process of representing, accessing and distributing information in a digital way. Unfortunately, it happens more and more frequently to hear about disclosures of sensitive digital information by unauthorized entities and the consequent negative impacts on business reputations, people's careers and financial markets. The Internet has amplified the consequences that leakages of confidential information might have, because of the rapidity by which digital data can spread.

Dealing with the confidentiality of digital documents is a complex task. It has strong implication in term of security and privacy. In involves the management of disclosure policies, access control and the satisfaction of trading and business requirements [8]: it has legal and legislative implications.

This paper focuses on a particular aspect of confidentiality: the timed release of confidential information, i.e., the protection of confidential information until a predefined disclosure time and its subsequent disclosure. We describe the HP time vault service, a technical work done at Hewlett-Packard Laboratories, Bristol, UK, leveraging Identifier-based Encryption (IBE) schema to encrypt confidential data. IBE (public) encryption keys specify the disclosure time. An independent time server generates and advertises IBE decryption keys corresponding to the current time, at predefined intervals of time. We discuss the advantages of our approach, in term of simplicity and efficiency, against current approaches based on traditional cryptography.

2. ADDRESSED PROBLEM

The problem addressed by this paper is the timed release of confidential information. It involves the enforcement of the confidentiality of digital documents, according to predefined time-based disclosure constraints, and an efficient distribution of their content, once they become public.

This problem is common in the physical world. 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, executive boards or working groups: often these documents can be disclosed to 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, stock market brokers, etc.) the involved parties might be prevented from accessing sensitive information for predefined periods of time. For example, in blind auctions a market maker can only access and disclose participants' bids at the end of the bidding time.
- In ordinary life, for example, students will 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 might 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 dealing with confidential material they need to take the proper precautions and protect it.

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:

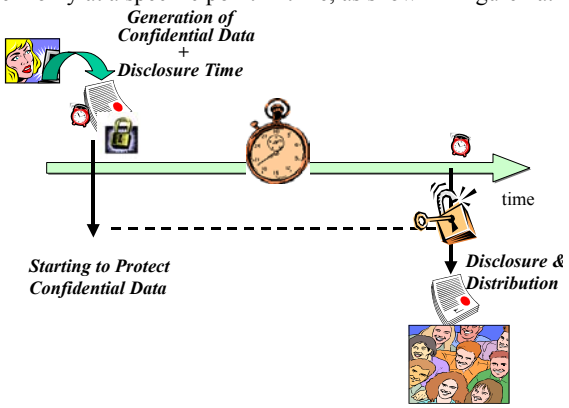

Figure 1a: Reference Scenario #1

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 kept secret until its intended disclosure time. 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. Afterwards, it can be distributed to the intended parties.

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

This scenario is particularly interesting, as it allows a gradual distribution of digital information by leveraging any kind of digital communication infrastructure (e.g. e-mails, web, web services, etc.) without compromising its confidentiality. Confidential information is obfuscated before being distributed.

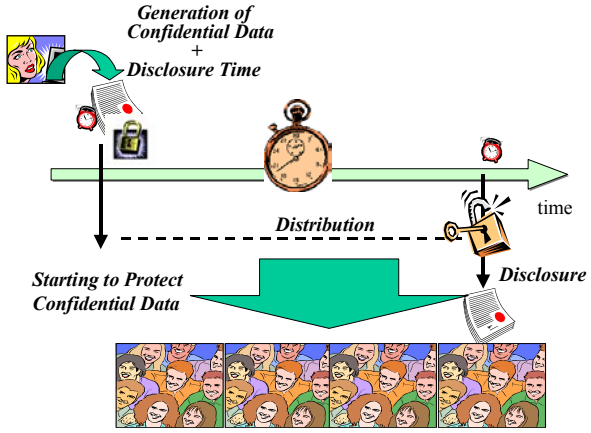

Figure 1b: Reference Scenario #2

The above scenario has advantages if compared to the scenario in figure 1a. For example, in large organizations the distribution of confidential information could be planned in advance, to avoid peaks of network traffic. Even for small groups this could be convenient. In case of people who 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-to-day life, people could receive in advance confidential 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) and the result of school tests. People could receive in advance their renewed digital credentials and access them when the current ones are expired, etc.

The key problems related to the scenario in figure 1b are the obfuscation of confidential documents and enabling people to access their content, at the intended disclosure time. This paper focuses on these problems but the proposed solution applies also for the scenario in figure 1a. *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 information confidential until its intended disclosure time.*

3. REQUIREMENTS

The enforcement of the confidentiality of digital information for predefined periods of time is not straightforward. Enterprises, governments or organizations usually define policies to address this problem but, at the very end, it is up to the individuals to understand and implement them. Often ad-hoc solutions are adopted. There is a need for simple mechanisms and tools to achieve the goal.

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 will be made available to people (and systems), once disclosed, is fundamental. As anticipated in the previous section, 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 timed release of confidential documents should:

- Provide mechanisms that strongly enforce the constraints on the disclosure time of confidential documents. Specifically these mechanisms must not allow the access to the content of confidential information until their intended disclosure time;
- Avoid bottlenecks: these solutions should provide mechanisms that allow a gradual distribution of confidential documents to the intended recipients with the assurance that recipients will be able to access the content only at or after their intended disclosure time;
- Be simple: people should be able to deal with confidentiality aspects of information in an intuitive and straightforward way;
- 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.

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

Section 4 describes related work and current approaches: it highlights their weak points. Sections 5 and 6 describe an alternative approach, invented at HP Labs, Bristol, UK.

4. RELATED WORK

Prior and related work is in the area of time-lock puzzles [9,13] and timed-release cryptography [10,13].

Time-lock puzzle mechanisms are based on computational complexity. They require a precise time to solve and an intensive usage of computational resources during this time. Although this approach is interesting, it is impractical in traditional enterprise and e-commerce scenarios.

The approach based on timed-release cryptography makes use of trusted agents. Paper [10] describes a mechanism by which a confidential document is stored by the trusted agent (or by multiple agents) until its intended release time. The disadvantage of this approach is the cost in term of resources (CPU, storage, etc.) to be used by the trust agent(s). Paper [13] proposes an alternative approach where trusted agents are not "escrow agents" as they do not have to store any information that is given to them by users. Their main task is to periodically publish a previously secret value. The disadvantage is that users must interact with a trusted agent every time they need to encrypt a confidential document.

We propose a related but alternative approach. It leverages the Identifier-based Encryption (IBE) schema [1,4,7]: users do not have to interact with a trusted agent to encrypt confidential documents. Users perform the encryption tasks, by using their computational resources, in a stand-alone way. This simplifies the model and introduces efficiency by reducing the number of required interactions. The main activity of our trusted agent is to generate and publish decryption keys: it is not affected by users' interactions.

Sections 5 and 6 describe in detail our approach. The remaining part of this section describes current technologies and solutions to address the problem. These solutions are based on:

- 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 [3,6,15] define which entities can access confidential information and what they can do with it. Access control lists (ACLs) can be modified over time.

Access control products and solutions, including IBM/Tivoli, CA and Netegrity products, are currently available for enterprises, e-commerce and web sites to locally protect the access to digital documents. Administrators of these solutions must be accountable for running them in a convenient and professional way. ACLs must be modified to grant/deny access to other people. Current access control systems, based on ACLs, do not explicitly manage the time factor. 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.

The encryption of confidential data [5] adds further protection as the original content can be accessed only if the proper decryption key is known. The intended disclosure date of a confidential document must be associated to the correspondent encrypted document and protected against manipulations. The encrypted digital document can 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 and symmetric key cryptography or hybrid combinations [14] 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, 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 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.

Services can be built on top of traditional cryptographic mechanisms to address part of the above issues. An example of such a service would allow people to directly 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 owns the correspondent private key). A confidential document is encrypted by a user via 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. To achieve this it has to perform cryptographic computations for every user interaction. A detailed description of the above service can be found in section 7.

We propose an alternative approach, based on the IBE schema, to build a service that is simpler and more efficient, in term of computation and required user interactions. No substantial computational overhead is introduced at the user side. The next section describes the properties of this schema.

Section 6 describes the *time vault service*, a service built to enforce the time-based confidentiality of digital documents and based on the IBE technology. Section 7 compares this service with a similar service built by using traditional cryptography.

5. IBE-BASED APPROACH

The ideal solution to the problem of the timed release of confidential documents would be a cryptographic mechanism where "decryption keys" are generated only at the intended disclosure time of encrypted documents: no secret specific to the document would exist before that time. Unfortunately there is no such kind of solution available today, as all known solutions directly or indirectly rely on one or more initial secrets (that must be protected over time).

This should not prevent from investigating alternative approaches to build simpler, easier to secure and more efficient solutions. The IBE schema can be successfully used to achieve these objectives. Next section briefly describes the IBE core properties and the IBE basic interaction model. Various IBE approaches are described in the literature. Section 5.2 briefly describes the QR IBE cryptographic approach [4]. Alternative IBE cryptographic schemas are described in [1,2,7].

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 IBE interaction model. Three players are involved: 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 fulfillment 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.

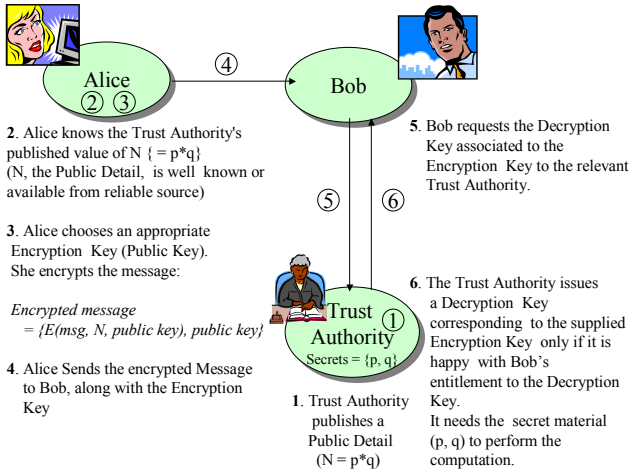

Figure 2: High-level IBE Interaction Model

5.2 Example of IBE Cryptography Schema

This section briefly explains how the quadratic residuosity (QR) IBE scheme works. This is accomplished by giving an annotated description of the various algorithms. For a more formal description, see [4].

5.2.1 The Interaction Model

The three players in the interaction model are 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.

5.2.2 Initialization

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 $p \equiv 3 \bmod 4$
2. Choose a different prime q such that $q \equiv 3 \bmod 4$
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, let 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)

5.2.3 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.

5.2.4 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:

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

5.2.5 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:

- Compute h = hash("Bob@trent.com")
- Compute b ≡ sqrt(h) mod N
- 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 not feasible to compute *b* unless you have knowledge of the (*p*,*q*) factors of *N*. Only Trent knows these factors.

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.

5.3 Leveraging the IBE schema

A service can be built to provide a timed release of confidential documents by leveraging the core IBE properties:

- Any string can be used as an IBE encryption key (public key). Specifically, this string could contain the disclosure date and time of the confidential document. For example, the string **"GMT200401011200"** can be used to encrypt a document and specify that its disclosure date is on January, 1st, 2004, 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 entity that provides the above service runs a trust authority in charge of generating time-based IBE decryption keys. It has to create, store and protect the trust authority's secret. This is the only secret (independent by the number of decryption keys that are generated from it) that needs to be preserved: security efforts can be concentrated to protect it. The problem of storing and protecting the IBE decryption keys does not exist, as trust authorities generate and publish them only when they are required.

Section 6 describes in more details this service, named "The HP Time Vault Service". Section 7 compares this service with a correspondent service based on traditional cryptography and highlights its advantages.

6. THE HP TIME VAULT SERVICE

The HP time vault service is a service based on IBE cryptography. This service can be deployed in a variety of environments, ranging from enterprises' Intranets to the Internet.

Figure 3 shows the high-level properties of the service:

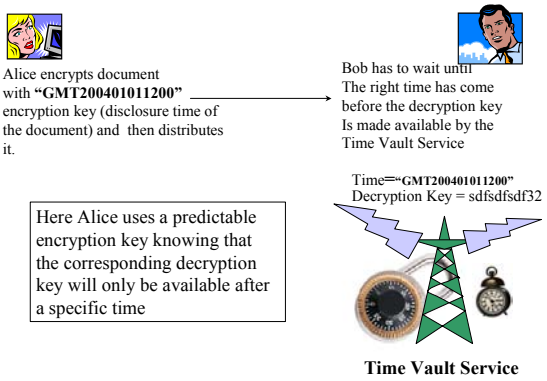

Figure 3: Time Vault Service: key aspects

People locally encrypt their sensitive documents by using their IT systems. No interaction with the time vault service is required. The time vault service ensures that the correspondent decryption keys are generated and issued only at their disclosure time.

Let us assume that 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, 2004, 12:00 noon (GMT). She uses the time vault service, a service run by an accountable and trusted third party on the Internet. 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 **"GMT200401011200"**, i.e. January 1st, 2004 – 12:00 noon (GMT)). To achieve this she also needs to use the IBE public detail associated to the time vault service (i.e. the IBE trust authority), that she already knows. 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 time. *Please notice that the current time is interpreted as if it is an IBE encryption key. When Alice generates her time-based IBE encryption keys, she must use the same (string) format adopted by the time vault service.*

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, in order 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. The time vault service does not need to know about the existence of Alice's IBE encryption key or the fact that Bob is waiting for the correspondent decryption key.

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

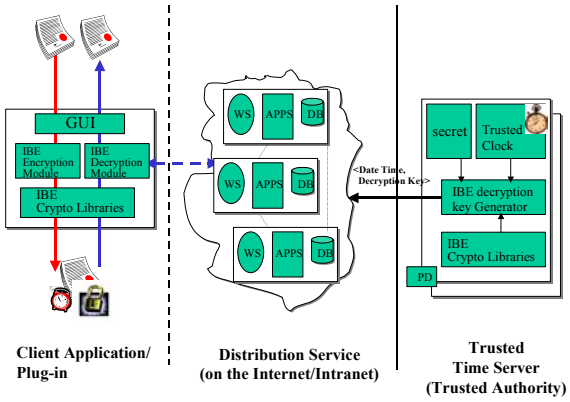

Figure 4: Time Vault Service: High Level Architecture

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. It *continually* issues IBE decryption keys correspondent to the current date and time. The frequency of issuances of these decryption keys depends on a predefined granularity, such as every minute or every hour or every day.

  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. The time server is built with secure and fault tolerant technology: it runs in a protected environment. It is important to notice that the time server is a self-contained component.

  It 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 communication - strongly protect and secure it.
- **Distribution Service**: the distribution service is a conventional Internet/Intranet portal specialized 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. 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). The distribution service returns the correspondent decryption key only if it has been issued by the time server, otherwise it throws an error message.

  The distribution service can be implemented with traditional technologies 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 can be replicated across distributed databases.

  The distribution service has to be reasonably protected against denial of service attacks and attempts to modify or destroy its information. 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 can be a plug-in for web browsers or e-mail browsers. It allows people to locally encrypt confidential documents 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 until a well-defined point of 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), encrypts the document and adds the necessary metadata (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 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 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 satisfies the requirements described in section 3:

- It provides mechanisms to strongly enforce the constraints on the disclosure time of confidential documents, because of the way it is built;
- 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 strongly encrypted digital documents by preserving efficiency at the disclosure time. The time vault service allows for the planning of distribution of confidential information in a balanced way, to avoid peaks and excessive traffic. Once the disclosure time of a document comes, people access the distribution service to retrieve the decryption keys. Large number of people could do this at the same time but they only need to download a few hundred bytes, corresponding to the desired IBE decryption key. No interpretation or decryption activities are involved at the server site, during this phase. The traffic generated (in term of transferred bytes) is potentially orders of magnitude smaller than the case where whole documents need to be downloaded or remotely accessed once their content is publicly disclosed.

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

6.1 Prototype

A fully working prototype of the HP time vault service has been implemented by using Microsoft .NET framework [11], as a proof of concept. Figure 5 shows the main components of this prototype:

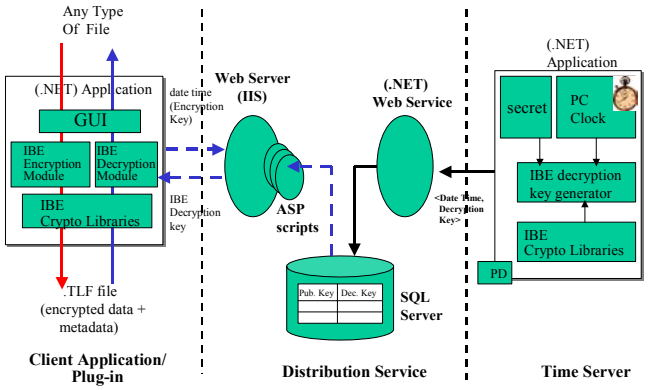

Figure 5: Prototype: The Time Vault Service

The **time server** has been implemented as a Microsoft .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 – based on [2,7] and developed in C for performance reasons. The IBE secret (associated to the time server) is generated at the initialisation, it is 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 publicised in a XML file.

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

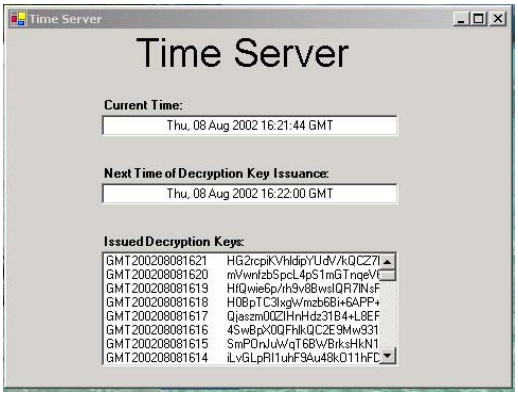

Figure 6: Time Server – Monitoring GUI Console

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 .NET web service and a SQL database server. It runs on a different 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 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.

The **client application** is a standalone .NET application, containing IBE encryption and decryption modules. The functionalities of the client application are exactly the ones described in the previous section. Any kind of document and file can be encrypted by this application: the encrypted file (.tlf file – time locked file) contains metadata – in the 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(S) protocol (the URL of the distribution service is stored in a local configuration file). Figure 8 shows the client application GUI.

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"**). 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 example where a user tries to decrypt the .tlf file before its disclosure time. In this 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(S).

Figure 7: Distribution Service – Monitoring GUI Console

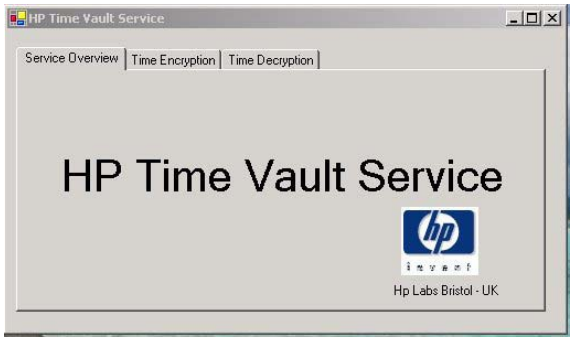

Figure 8: Client Application

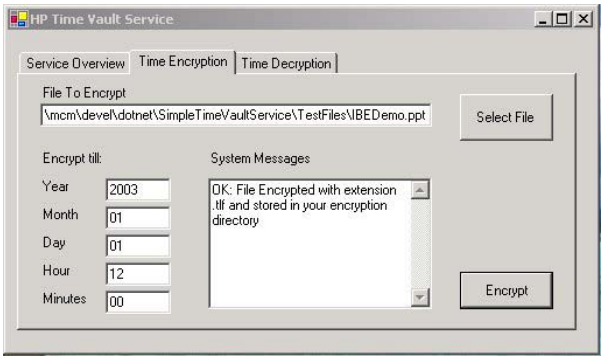

Figure 9: Example: Time-based Encryption

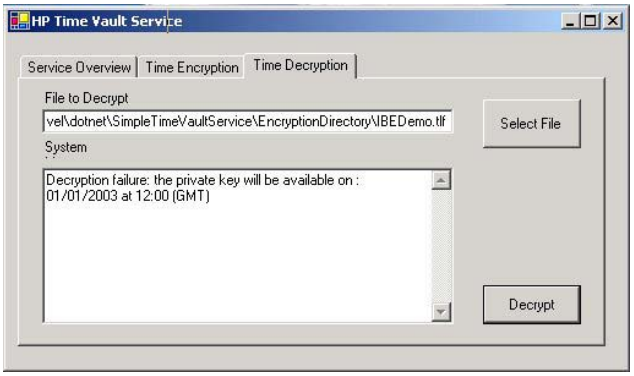

Figure 10: Example: Failure in attempting to decrypt a file before its disclosure time

7. DISCUSSION

The HP time vault service described in this paper relies on IBE cryptography to provide its core functionality. A similar service can be implemented by making use of traditional cryptography, for example, based on RSA. The aim of this section 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 as implemented in our approach, by using IBE cryptography:

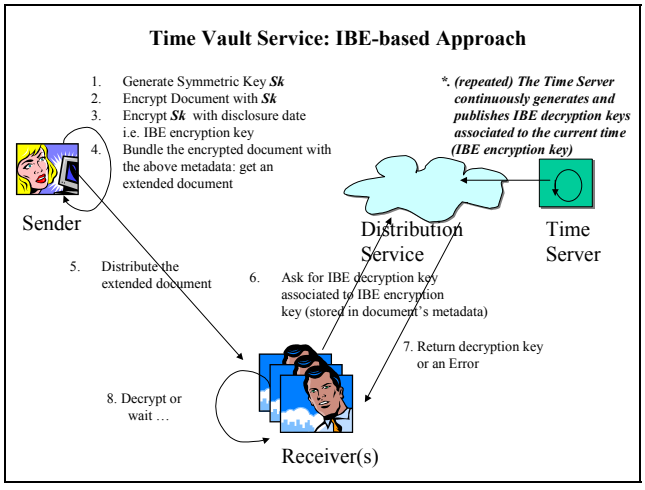

Figure 11: Time Vault Service: IBE-based approach

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 generates 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 a "correspondent" time vault service, build by using RSA public key and symmetric key cryptography:

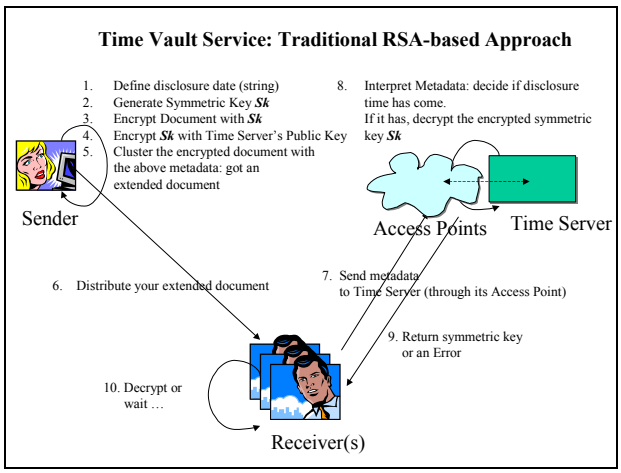

Figure 12: Time Vault Service: RSA-based approach

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

The time server model based on **traditional cryptography**, has to perform interpretation and cryptographic operations for each user interaction, before releasing a decryption key. The access point, associated to the (RSA-based) time server, must interpret documents' metadata to verify if the constraint on the associated disclosure time is satisfied. In such a case, the time server must perform a decryption to retrieve the symmetric key used for encryption. These two components need to interact and exchange information. This might cause delays especially during peaks of users' requests and introduce security 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 model based on the **IBE cryptography** is simpler to run as it is modular, each component is self-contained and interactions about these components are reduced to the minimum. The time server activity is completely independent by users' requests for IBE decryption keys. It is predictable. Its only allowed interaction is with the distribution service by means of an outgoing connection. Because of these aspects it is easier to protect.

It is also potentially more efficient. *Despite the fact that it is out of the scope of this paper to provide a quantitative analysis of our time vault service performance, it is possible to qualitatively compare our proposed model against alternative models.* In our IBE-based model, users' interactions with the distribution service do not trigger any interpretation of "time-based disclosure policies" and cryptography computations, whilst this happens for the RSA-based model. Even in case of peaks of requests, the IBE-based distribution service only need to execute simple database queries and return a few hundred bytes to the client (the size of the decryption key). If compared to the trusted agent described in [13] our method has the advantage of not requiring users to interact with the time server at the encryption time (although this is true also for the RSA based approach). The potential overhead of our 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 issuance rate of decryption keys: it can be addressed during the configuration phase of the time server.

The IBE-based model can be 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 [12], 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. CONCLUSION

The management of confidential documents is an important issue in business, social and government environments.

This paper focuses on an alternative approach to the problem of timed release of confidential information i.e. the problem of disclosing confidential digital documents only after a well-defined point of time. Our proposed approach is potentially simpler and more efficient than current solutions.

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 distributed in advance to a population 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) coupled with solutions and services that allow the access to confidential documents or their decryption keys only at their disclosure time.

We introduced the HP time vault service, based on the IBE cryptography schema. Simple strings can be used to specify the disclosure time of confidential documents: these strings are directly 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 independently generates and issues decryption keys at predefined intervals of time.

We compared the HP time vault service with a correspondent service built with traditional cryptography: we showed that our service is simpler and qualitatively more efficient. This service can be deployed in a variety of environments ranging from enterprises' Intranets to the Internet.

A prototype has been implemented and described as a proof of concept.

9. REFERENCES

[1] Boneh, D. and Franklin, M. Identity-based Encryption from the Weil Pairing. Crypto 2001, 2001

[2] Chen, L. and Harrison, K. and Moss, A. and Soldera, D. and Smart, N. P. Certification of Public Keys within an Identity Based System, LNCS 2433, ed. G. Goos, J. Hartmanis and J. van Leeuwen, Proceedings of Information Security, pp. 332-333, 2002.

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

[4] Cocks, C. 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

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

[6] Ferraiolo, D. and Kuhn, R. Role-based Access Control. NIST, 1992

[7] Frey, G. and Muller, M. and Ruck, H-G. The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. IEM Preprint No.23, 1998

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

[9] Garay, J. and Jakobsson, M. Timed Release of Standard Digital Signatures. Financial Crypto, 2002

[10] May, T.C. Timed-release crypto, February 1993

[11] Microsoft. Microsoft .NET framework. http://www.microsoft.com/net, 2002

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

[13] Rivest, R.L. and Shamir, A. and Wagner, D.A. Time-lock puzzles and timed-release Crypto. MIT laboratory for Computer Science. MIT/LCS/TR-684, 1996

[14] RSA Laboratories. PKCS#7: Cryptographic Message Syntax Standard. Version 1.5, 1993

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