DIGITAL TIMESTAMPING: A WEB-BASED ANONYMOUS DISTRIBUTED SOLUTION

Prof. S. H. Ahmed[1], A. Prof. Imane Aly Saroit[2], Ehab Emam Zakaria[3]

[1]Vice Dean of Faculty of Computers and Information
[2]Faculty of Computers and Information, Information Technology Department, Cairo University
[3]Teaching assistant, Computer Science Department, Modern Academy in Maddi

Email: *ehabemam_it@yahoo.com, iasi63@yahoo.com*

Abstract

Timestamping is a technique used to prove the existence of a digital document prior to a specific point in time. All the existing timestamping techniques depend on a trusted third party to provide the service. In this paper, we define a reliable anonymous distributed timestamping service that distributes the trust among a number of randomly chosen entities managed by administratively independent entities.

Keyword: Timestamping, Distributed trust, Anonymity, Web-based applications.

1. Introduction

The advent of electronic commerce has made the security of communication a major concern. Many governments have chosen to communicate and conduct transactions with citizens, businesses, and other agencies in a secure online environment. The security requirements for electronic document exchange are to ensure the integrity of the official communications to protect constituent privacy, authenticate people and processes and possibly and control sensitive information. Digital signatures help to provide ongoing assurance of authenticity, data integrity, confidentiality and non-repudiation. Timestamping techniques are concerned to certify that an electronic document was created at a certain date. This certification is mandatory for a lot of applications in various domains like patent submissions, intellectual property or electronic commerce.

Haber and Stornetta presented the first timestamping protocol [1] during Crypto '90. One year later, Benaloh and de Mare proposed a formal definition for a timestamping system based on a set of participants and three protocols [2]. Since then, a lot of new schemes were proposed and their security analyzed [3-7]. Most of them use the concept of trusted Time-Stamping Authority (TSA), which is supposed to be able to securely timestamp an electronic document.

However, it may be difficult to build a third party server that can be trusted. Indeed a server may be corrupted or victim of denial of service attacks. In fact, we claim that timestamping schemes relying on a unique third party server cannot be trusted. Moreover, existing systems does not provide user anonymity, which is considered as an important requirement for users as well as for the authenticity of timestamp production. Therefore, our objective in this paper is to propose a timestamping scheme using a distributed architecture that supports user anonymity.

Our proposed timestamping protocol can be viewed as a hybrid protocol that benefits from the advantages of the linking scheme and distributed scheme but after major modifications to both of them in order to able to achieve the desired level of security and performance.

In the proposed scheme the timestamp certificate consists of two parts; the linking certificate and the witness certificate, which are produced by the stamping coordinator (Main Server) and the witnesses (Distributed Web-Services) respectively. ***The linking certificate*** is built using a modified linking scheme that accumulate all the requests falling in the same time slot (round) into a single value called round value that linked then with pervious round value producing a super round value that links all the round values iteratively together. ***The witness certificate*** is a structure that filled up with a number of signed responses from a randomly selected web services that witness the values of a certain round (round number, value, time), our scheme ensures that witnesses are chosen in a total random manner in order to prevent any pervious collision with any of the selected witnesses.

As anonymity is one of our major requirements in the design, we confronted with the problem of how the service can charge its users, since anonymity means to hide the identity of service requester. We investigated many types of payment in order to find a one that is compatible with anonymity requirement. So that; we developed a procedure that make use of wireless communication technology and prepaying charging.

The proposed scheme is used to deploy a service that produces a timestamp in a fully non-repudiatable, trusted, traceable manner. We implemented the service and made experiments to find that the proposed architecture with reference to existing evaluation efforts (although it is rare for timestamping systems, so depended also on a logical method of evaluation) was able to achieve the highest security level without neglecting other important aspects like processing power and throughput.

This paper is organized to include five sections as follows. **Section 2** provides basic definitions regarding timestamping process, a survey that classifies existing timestamping schemes, a list of existing timestamping projects with its employed stamping scheme and a previously proposed method for evaluating the security of a generic timestamping scheme. **Section 3** provides a detailed description for our proposed solution describing our design primitives, a tailored method for providing user anonymity integrated with a method for service payment, describing our system architecture and its use cases. In **Section 4,** we evaluate the security of our proposed solution using logical evaluation and results of pervious studies in the area of security evaluation of timestamping schemes, also we present the results of deploying our proposed solution. **Section 5** provides conclusion and system deployment experiments and shows our future directions of research.

2. Timestamping Schemes

Timestamping schemes are classified into three categories: simple schemes, linking schemes and distributed schemes. The term TSA here refers to the Timestamping Authority; which is the entity responsible of producing the timestamp.

2.1 Simple Protocols

Whenever a client wants to obtain a timestamp on a digital document, he/she transmits the document to a time-stamping authority (TSA). TSA records the date and time when the document is received and stores a copy of the document for safe -keeping. When one wants to check the integrity to verify the timestamp, he/she simply makes a comparison with the copy stored by TSA. This solution has at least the following problems:

- Since the document itself is transmitted, an eavesdropper can see it.
- Clients must worry not only about the trust of TSA but also about the security of TSA that might be compromised.

- The scalability is insufficient and the TSA's storage capacity might be easily exhausted.

The first privacy issue can partially solved by submitting not a document itself but a collision-resistant hashed value of the document.

On the other hand TSA appends the current date and time t to the document X, generate his/her digital signature S = sig(t, X) on the whole, and TSA Then return stands to the client. T is represented with a specific granularity depending on the application .When one wants to verify the timestamp, he/she verifies the digital signature by using TSA's public key. The weakness of this protocol is in that it relies on the security of the digital signature (and its underlying public-key infrastructure) too much. One can hardly imagine that this protocol is used to extend the life -time of a digitally signed document. Therefore, we do not suppose simple protocols in the following discussions on long-lived authenticity.

2.2 Linking Protocols

To solve the problem in TSA's trust and security, the clients may want the TSA to link all timestamps together into a chain or a graph by using a collision-resistant hash function *H*. In the case of a linear linking chain [1], the timestamp for the hashed value $H_n = H(X_n)$ of the n-th document X_n is defined as:

$$S = Sig_{TSA}(t_n, ID_n, H_n, L_n) \quad (1)$$

Where t_n is the appended time, ID_n is the client's identifier, and L_n is the linking information which maybe expressed as:

$$L_n = (t_{n-1}, ID_{n-1}, H_{n-1}, H(L_{n-1})) \quad (2)$$

In this way all the produced timestamps are linked in a directed graph structure with arcs follow the time ordering of the stamping operation, which prevent the timestamp issuer from backdating certain timestamp by inserting it in a previously linked chain.

2.3 Distributed Schemes

Another way of lowering the required level of trust in the TSA is to distribute the trust. In that approach, multiple users / TSAs cooperate to generate a timestamp, possibly using a secure distribution of secret data necessary to generate a timestamp. In this way, forgery of a time-stamp requires the collusion of a predetermined (high) number of parties, which is considered to be very unlikely. Example of such protocols can be found in Benaloh et al. [2] and Ansper et al. [7]. They can be extensions of the schemes described above, and as such provide relative temporal authentication or absolute temporal authentication.

The distributed scheme has a number of drawbacks. First, the set of possible clients should be established a priori, and the pseudorandom generator should determine only existing clients. Even if this condition is assured, availability of random chosen clients could easily determine a weakness in the system. Moreover, the bandwidth required for a single timestamp issuance would be k times larger than that for the simple scheme (for the k randomly chosen clients). Further problems are related to the synchronization of many distinct time sources and to the legal definition of liability for the certified time. In fact, the distributed scheme, proposed along with the linking scheme in 1991, has not been studied afterwards. And actually this lack of study for distributed schemes and the defeats of the

existing distributed algorithm leads us to study and build a new distributed trust that overcomes weaknesses of existing solutions while being reliable.

2.4 Security Evaluation of Timestamping Schemes

Although there have been several studies of the security of timestamping protocols, **there are no strict security measures for them in a cryptographic (computational) sense**. Instead we studied different timestamping systems to find that; backdating and postdating a timestamp are the most basic attacks. Backdating attack takes the form of issuer collusion with the requester in order to produce a timestamp with a pervious date/time. While postdating attack may take the form of Denial of Service (DoS), where an attacker blocks all the requests before it reaches to the issuer, therefore the requester may need to retransmit the request which result in delaying (postdating) the request [17].

Our finding was supported by the work done by Masashi Une [16] in the field of timestamping systems security evaluation. Their work can be summarized as follows: they find that the best method of classifying timestamping systems is through **classifying the used operations during timestamp verification phase**. They choice this way as they think that these verification operations can discover any possible collision or alteration in the timestamp, putting in mind that security of any timestamping scheme lies in the ability to discover the alteration, if happened.

Une, Masashi, Tsutomu Matsumoto [17, 18] classified timestamping schemes into classes. Each class employs the same verification procedure. They divided the employed verification operations into six operations, which are:

- **Operation a:** A verifier compares a hash value (Digest or Imprint) of message *M* with *H* (Hash value) included in a timestamp to be verified.
- **Operation b:** A verifier confirms the integrity of data included in a timestamp (checking digital signature of the issuer over the timestamp).
- **Operation c:** A verifier asks an issuer to confirm the consistency between a timestamp to be verified with the corresponding data in the issuer's database.
- **Operation d:** A verifier confirms the consistency between data included in a timestamp and the corresponding data in the issuer's database
- **Operation e:** A verifier obtains DATA from an amplifier (e.g. Newspaper) and confirms the integrity of timestamp generation process by using these DATA.
- **Operation f:** A verifier obtains DATA' from predetermined requesters and confirms the integrity of timestamp generation process by using DATA'.

They arranged the classes depending on the used verification operations in each of them, for example schemes employing operation(s) a or ae or af or aef are considered of class 1. (**Table 1**):

Class 1	**a, ae, af, aef**
Class 2	**ab, abe, abf, abef**
Class 3	**ac, ad, acd, ace, acf, acef**
Class 4	**Abc, abd, abcd, abce, abcf, abcef**
Class 5	**ade, acde**

Class 6	**abde, abcde**
Class 7	**adf, acdf**
Class 8	**abdf, abcdf**
Class 9	**adef, acdef**
Class 10	**abdef, abcdef**

Table 1: Classes and Corresponding Operations

They showed that, the higher the class in which the scheme exist (depending on the operations employed in this scheme) is the higher security grade that it deserve (scheme is more secure against timestamp backdating/ postdating).

3. Proposed Timestamping Protocol

In this section a proposal of timestamping protocol is presented. The design primitives, anonymity, and system architecture are presented.

3.1 Design Primitives

A number of primitives related to our design are discussed here; these primitives are: the goals of our design, our deployment methodology and assumption.

3.1.1 Design goals

The main idea behind this system is to explore the strengths of peer-to-peer distributed systems for building trusted services. To provide the service the following three properties must exist:

1. **Longevity:** The system must have a high probability of remaining operational for long, continuous periods (on the order of several decades).
2. **Wide Trust:** The system must be able to convince parties from different areas of the world, with different affiliations, business practices, or loyalties, at the same time that the other entities performs its tasks correctly.
3. **Anonymity:** The system should not associate the customer to the request.

3.1.2 Deployment Methodology

In order to deploy the system a number points were imposed in order to achieve the desired goals of the design. These points are:

- Building the system on a loosely coupled, decentralized, distributed system of nodes. These nodes are a randomly chosen Web Services (managed by profit seeking organizations) to participate in the timestamping process.
- No system node is more significant than any other node.
- Any and all nodes are replaceable.
- This replaceability eliminates the threat of single-point short- or long-term failures.
- No matter how many participating nodes die, as long as new nodes join at least as frequently as needed, the system will survive for longer periods than any single, well-maintained node could by itself.
- The main server; is just a coordinator for the entire processes, we call it Secure Time Authority (STA).

- For timestamping main server there is always redundant servers in geographically distrusted data centers.
- The critical data are replicated between data centers so each center can serve as a backup to the other.
- Participating web services are **randomly** chosen in a traceable, trusted and secure manner.
- The availability of a fair auditing and verification mechanisms for the system.
- The built system is based on a prepaying basis, which support the anonymity of the user (like prepaid cards of GSM phones).
- Enabling the clients to use their wireless PDA in this application to send their messages through wireless network which hide the identity of a message sender.

3.1.3 System Assumptions

The following assumptions take the form of "minimum system requirement", while others place restrictions on environmental parameters. These assumptions are:

1. At any one time, no more than f nodes are faulty. If the number of all the nodes at any time is n, then $n \geq 3f+1$.
2. System nodes fail or die independently.
3. The current system nodes, as well as any other entity external to the system, may only perform polynomially bounded computations.
4. System nodes and clients have a well-known mechanism for authenticated communication amongst themselves.
5. All correct nodes have a time sources that may slightly drift than the correct global time (in order of milliseconds).
6. The certification authority Track identity certificates for registered identities (the candidate nodes) and update the archive with their timestamped changes.
7. Any system node must meet the system's minimum requirements, such as processing, connection speed and, stability.

3.2 User Anonymity

Anonymity [11], [12] is an important property for timestamping service user where;

- In many situations, a user requesting the service may not want to disclose his identity for any reason (for example, to prevent fraud referring to his identity by service).
- In addition, anonymity prevents the service itself from postdating or backdating the timestamp for a particular user, since it does not know his identity.

3.2.1 Lack of Anonymity in Postpaid Services

The problem with postpaid services is that the service provider (the STA, for example), can now recognizes the identity of the request sender, and the number of requests he send per day, *while a user may want to keep those information secret.* But with a postpaid service this is not possible, since for the sake of user authentication, user's computer or PDA can use any hardware (the serial number of his CPU) or software identifier (an assigned number, for example) previously agreed upon with the service provider (STA). A side effect of this scheme is that user's identifier can readily be used by STA to find out all about the information that the user wants to hide.

3.2.2 True Anonymity in Prepaid Service

One Solution to provide anonymity is that, the user can purchase top-up scratch card of the STA, loaded with a certain amount of credit from which the cost of the service is deducted, similar to the ones

used by the current prepaid phones [13- 15] and uses the hidden number (PIN code) as a secret identifier to access the service. This PIN code is appended to the request and encrypted as a part of the entire request using the public key of the STA, (to save the PIN code from being stolen while transmitted through network if transmitted without encryption). However, there is still a problem, if the user sends the request from his personal computer through the internet, his IP address, of course will be known, and hence his identity will be revealed. The simplest way for accessing a service under complete anonymity is to access it through IP addressless device like wireless PDA

3.2.3 Sending True Anonymous Requests

In this section we present a scheme for sending an anonymous request to the STA. The scheme is shown in figure 2, which is a tailored version of scheme provided in [11].

- We use anonymous e-cash for the anonymous payment for the wireless network service.
- While we use the scratch cards, as an anonymous method of payment for the STA.

Entities involved in the scheme are:

Bob: is the anonymous request sender.
STA: is the recipient of the anonymous request on his server, and
Doug: is the owner of the MSS and offers communication services on a pay-for-time-used basis.
Clare: is a bank owner and offers support for anonymous e-cash payments to her account holders (Bob and Doug). Finally,
Ebe: is another PDA user.

The algorithm works as follows (described graphically in figure 2):

K_{pu}: is short for public key
K_s**:** is for secret key.

1. Bob turns on his PDA and learns the K_{pu} of the MSS by listening to its advertisement.
2. The PDA creates a K_s, encrypts it using the K_{pu}, and sends it to the MSS for approval, waiting t units of time for a reply.
3. The MSS checks that the K_s suggested by Bob is correct and not in use;
    a. If so, it creates a *TmpId* for Bob, encrypts it using the K_s, and sends it to the PDA as a reply,
    b. If the K_s suggested by Bob is incorrect, the MSS does not reply,
    c. If it is correct but has been assigned to an existing user, the MSS does not reply to Bob and additionally asks the user of the existing K_s to renew his K_s,
    d. After t units of silence, Bob can try again.
4. The approved K_s is used then to encrypt and decrypt messages between the PDA and the MSS until either the end of the session or until it has to be renewed. Messages encrypted with Bob's K_s can be overheard by Ebe but they will be ignored.
5. Bob sends an anonymous e-coin to Doug to pay for the communication session. Doug consults Clare about the authenticity of the coin before accepting or refusing it.
6. When Bob wishes to anonymously send request to the STA;
    a. He appends the PIN code in the scratch card to his request and encrypt the entire message with K_{pu} of the STA,
    b. And specify target address of the message (STA address),
    c. Then encrypts the result with Ks, and sends it to the MSS.
7. The MSS decrypts the message, combines enclosed message body together with Bob's *TmpId* in a single message body, and forwards it to the STA.

8. STA has no means to discover the identity of the *TmpId* holder. Yet, STA can reply to Bob by addressing it's response to the MSS and including Bob's *TmpId.*
9. Bob's session ends when he turns off his PDA, or his MSS times-out his session.

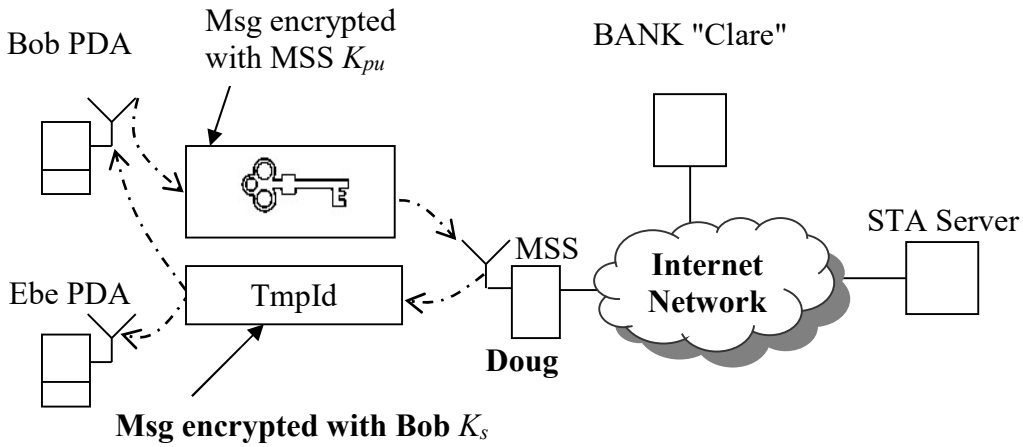

Figure 2: Sending a message anonymously

3.3 System Design

In this section we describe the architecture of our system including building components and an outline for the way that the system works with.

3.3.1 System Components

The system is composed of two components:

- The time stamping authority main server (STA) which;
  - Accept the requests,
  - Partially create a Timestamp certificate,
  - Send witnessing request to the randomly chosen nodes in order to produce the complete timestamp certificate.
- The distributed signing web services (system nodes); which
  - Considered as witnesses to the process,
  - Receive a request from the STA and append the current time of request arrival to that request and sign it to be sent back to the STA.

The STA as well as system nodes are implemented as web services, which enables as to benefits from advantages of Web Services Technology

The Distributed Signing Web Services (Witnesses)

- Web Services are owned and managed by profit seeking organizations. Owners of those web services may send a registration request for joining the system as a participating node. This request includes:
  - IP address of the service.
  - Public key certificate from a trusted certification authority.
- The STA can accept or refuse the request based on the repudiation of the service or a past experience with it.

- STA publish a list of the contributing web services with its specific order through **unmodifiable media** prior to starting the operation of the system (for sake of cost reduction, the digest of the list is published instead of the list itself).
- For each round the STA select only *n* web services from all the registered ones to witness the round values.

ID	IP Address	Company Name	Public Key Certificate
…	…	…	…

Table 2: The newspaper published list of system nodes (candidate web services)

Web service obligations:
- Only, it appends the current universal absolute time and signs the result.
- It has no storage obligations.
- It has no commitment for longevity (Where **it can go out of business**).
- **Fortunately the above aspects does not affect the process of timestamp verification in which we need to confirm about timestamp generation :**
    - Through checking the ***Certification Authority*** that maintain an archive of public key updates, therefore we can confirm if a certain web service signed a certain requests or not even if this web service has no copy of the signed request or even it had gone out of business at all.

3.3.2 System Architecture

Our architecture uses a round-based linear linking scheme where:

A. Timestamping requests falling in a given time window (round) are gathered in a tree-like structure and compressed into a single data item to be linked with the results coming from the previous time windows, as appears in figure 2:

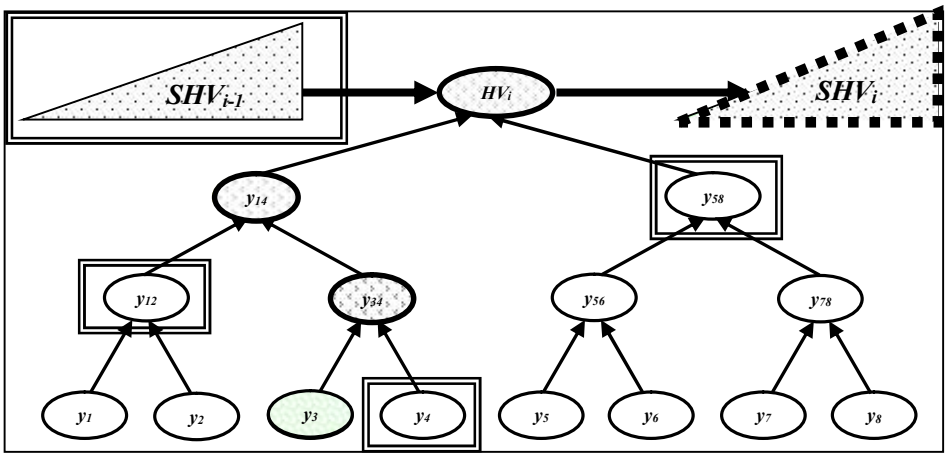

Figure 2: Linking protocol with a binary-tree structure.

In Figure 2, $y_1, y_2, \ldots, y_8$ represents the requests arrived during round *i*, y_{12} is generated through concatenating y_1 and y_2 and hashing the result, the same is done for y_{34},... . In next level, the same is

done but for y_{14}, y_{58} until we reach to the root of the tree which is called hash value of round n (HV_n), which is linked to the hash value of pervious round (SHV_{n-1}) to produce the Super Hash Value of the current round (SHV_n).
In order to prove the inclusion of a certain request in the generation of a certain Super Hash Value, a membership path is given to each user to prove his participation in the round. For example the membership path for y_3 is: y_4, y_{12}, y_{58},SHV_{i-1}

This linking information of the round (i, t_i ,SHV_i) is periodically (by the end of each day) published in a media, which is widely and publicly recorded. A popular example is to print the linking information in a newspaper. Fortunately, a newspaper;

- Is physically dated (i.e., the issuing date is tamper resistant printed) and examined for signs of after -the-fact tampering,
- Contains temporal data which could not be predicted in advance (weather information, stock- market information, sport results, etc.),
- Functions as a global public record whose long -term availability at many locations (e.g., major libraries) makes tamper-proof property far more better,
- Could be finally recorded even on library microfilms.

B. The round information (*round number, time, SHV_n*) is then submitted to randomly chosen web services, that are responsible for putting the current time (witness) and sign the resultant structure with it's private key and send it back to the STA,
C. Our system eliminates the need for a single point of trust. So the system aims to distribute this trust among a number of randomly selected entities that are the system nodes (Web Services).

The following steps provide an explanation for the process of timestamp generation, which is graphically illustrated in the figure 3:

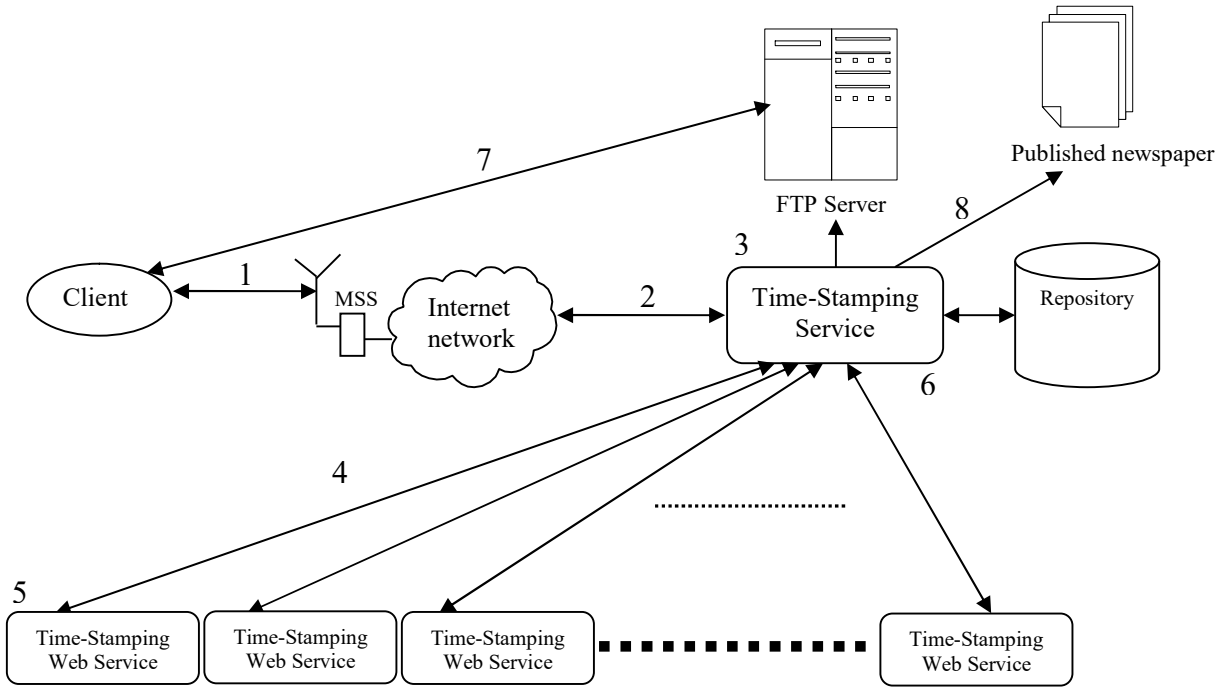

Figure 3: System Architecture

1. A user u designates a document y and produce h which is SHA-1 digest and h' is RIPEMED-160 digest. The client sends the request (y^h, $y^{h'}$) attached with a PIN code that used as a payment method. The entire request structure is then encrypted with STA public key to the time stamping service (STA) anonymously, through a wireless network,
2. Given that the user is authorized and the request correctly formatted, the STA sends an identification number (ID_u) to the client for his request, which used later for timestamp retrieval,
3. Upon accepting the request and assigning the ID_u the service do internally a number of actions:
   a. The STA computes the round tree,
   b. The STA signs and store the round values HV_i (Hash Value), SHV_i (Super Hash Value), round number i and time t_i; this structure represents the first part of the Timestamp certificate which we call the **Linking Certificate** due to its production process,
   c. The STA will take the resulting the SHV_i, SHV'_i (concatenated) of the round as a seed value for the pseudorandom number generator to select the desired number of random web services addresses,
   d. The STA will construct the request which include the round number, round time and SHV_i, SHV_i' of that round and sign it (signature is for authentication purposes),
4. The STA send the constructed request structure to each of the randomly chosen web services,
5. At each web service; the service append the current universal time to the request and signs it, return it back to the STA,
6. After the desired number of web services sign, the STA will make a structure contains all the responses belonging to that round, and publish it on the online server, where either the user or the verifier can get and check it; this structure represents the second part of the Timestamp certificate which we call the **Witness Certificate**,
7. Using the identification number ID_u given to the user by the STA the user accesses the server and retrieves his timestamp certificate,
8. By the end of each day, the STA publishes the SHV_i, SHV_i' of the last round of the day in a widely witnessed newspaper.

From pervious description; it is clear that our produced stamp is consisting from two parts.

1. First part produced internally using the linking procedure (Linking Certificate),
2. While the other is produced using the randomly selected web services as a witnesses (Witness Certificate).

3.3.3 Comments about the design

Step 1:

a) We aimed from using two hash algorithms for producing message digest to provide more security for the system in case of that one of the hash algorithms is broke.
b) The entire message is encrypted after appending the PIN code is done in order to prevent a man-in-the-middle attack from stealing the PIN code if it has been sent clearly without being encrypted

Step 2:

Since the user is anonymous to the timestamping service, so the service can not deliver the produced timestamp to him. Therefore he is given an ID that he can use it to retrieve his timestamp from an FTP server.

Step 3:
The service builds two trees for the received requests, each tree using a different hash algorithm for the motive of more security.

Step 4:
The choice of the web services (witnesses) that will receive a witnessing request is done through using the super hash value of the round (SHV_i) as a seed to a pseudorandom number generator that produce n random IDs from the table (). Then those IDs are used to retrieve the corresponding IP addresses of the randomly selected web services. The selected IPs can not be changed to other ones since they corresponding to IDs that are already published in a widely witnessed newspaper.

3.4 System implementation design (Service Core)

In order to achieve the design goals mentioned earlier, we deployed the next implementation scheme to generate the proposed timestamp. We divided charges of the timestamp generation process into independent steps; we try to decouple the load between them. Each step has a working queue. Those queues are in charge of softening the speed differences between the different process steps. A schematic outline of the process is given in Figure 4. The components are the following:

- **Network Listener:** The "Network Listener" is in charge of continuously listening to the clients' timestamp requests.
- **Round Timer:** The "Round Timer" receives the constructed requests from the "Network Listener" to times them (determine the round of the request depending on its arrival time), to be forwarded to the "Round Queue Coordinator".
- **Round Queue Coordinator:** Each round has its own "Round Queue Coordinator", which is in charge of compiling and processing into a tree all the requests belonging to the round.
- **Random IP Generator:** The "Random IP Generator" takes the round value (SHV_i) from the "Round Queue Coordinator", and uses it to generate N random web services' IPs.
- **Request Generator:** Once the "Request Generator" receives the N IPs from the "Random IP Generator", it starts preparing the requests structures; with each include the details of the round to be sent to the web services through the "Request Sender"**.**
- **Request Sender:** Once a request structure is generated, the "Request Generator" forwards it to the "Request Sender", which in turn forwards it to the web service IP specified in the structure.
- **Web Service Response Listener:** the "Web Service Response Listener" is in charger continuously listening to the web services' responses.
- **Response Discriminator:** the "Response Discriminator" is in charge of assembling web services' responses of the same round into a single structure (since responses for different rounds may be arrived at the same time) to be forwarded to the "Web Answer".
- **Timestamp Generator:** When the round tree has been computed, it is forwarded to the "Timestamp Generator", which generates the corresponding timestamps.
- **Web Answer:** Once a timestamp or web services response structure are generated, the "Timestamp Generator" or the "Response Discriminator" respectively forwards it to the "Web Answer", which in turn forwards it to the STA web server.

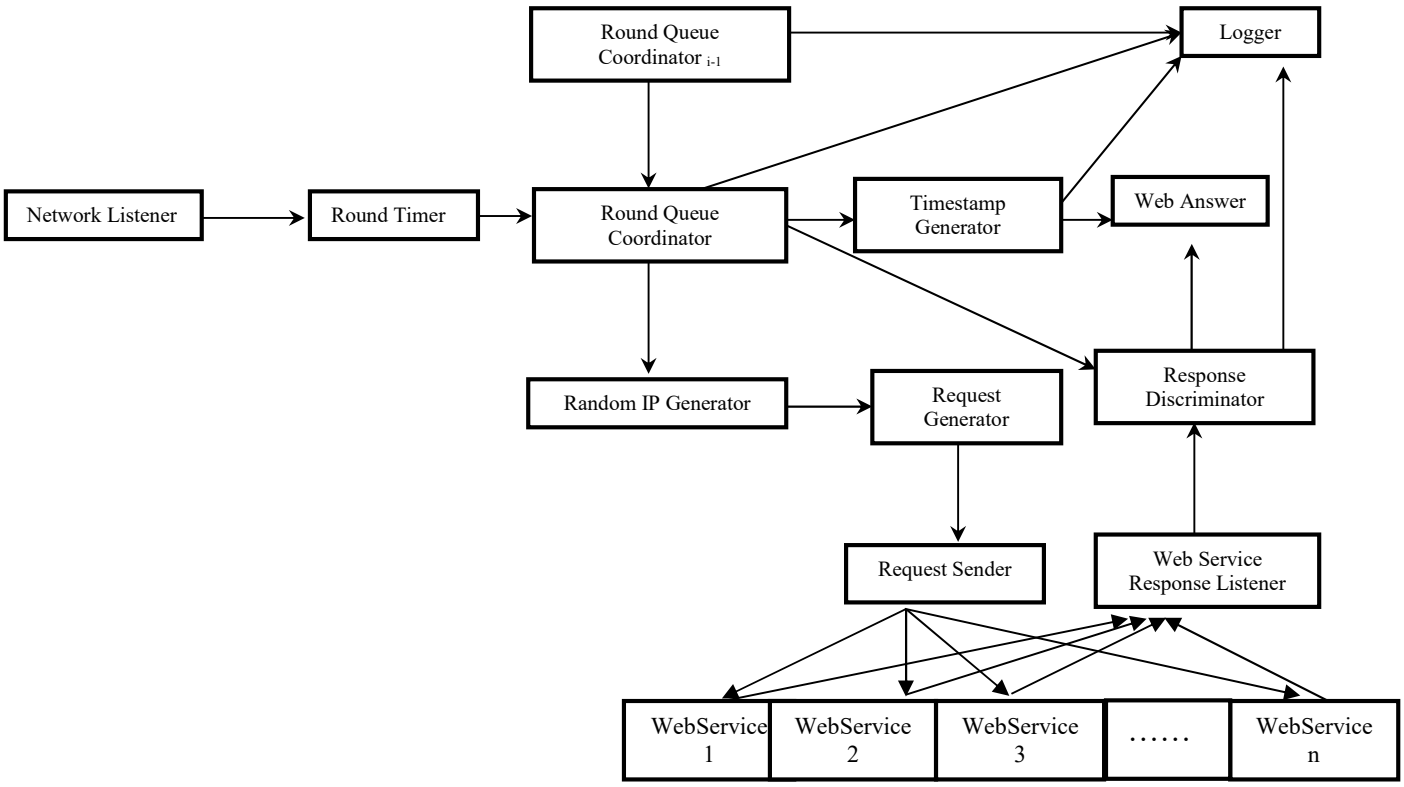

Figure 4: Interaction between the components.

3.5 Timestamp Verification

The following steps take place in order to check correctness of a timestamp by a verifier (the entity who needs to check timestamp correctness):

1. The verifier designates a document and a "Timestamp Certificate" (Linking and Witness Certificates).
2. The verifier system generates two document hashes using RIPEMED-160 and SHA-1, and checks if it matches with the one in the linking certificate **[Operation A]**.
3. The linking certificate structure is checked, including membership of the document in the round (the path from the hash value of the document to the round value through the authentication path included in the certificate). STA signature over the linking certificate is then checked to ensure the authenticity of that certificate **[Operation B]**

Extensions

The following steps are considered additional verification steps that may be done by the verifier in order to achieve higher confidence level about the investigated timestamp.

A. The verifier wants to check the certificate against its round value. (Linking Certificate Check)[Operation D]

i. The verifier system accesses the log of the STA to obtain the round value of the certificate.
ii. The verifier system checks if the round values match.

B. The verifier wants to check the round construction (Big Round Check) [Operation E].

i. The verifier provides his system with the two big round values (*SHV*) that can be found in the "unmodifiable media" (provided by the publisher).
ii. The verifier system accesses the log of the STA and obtains all the round values (*HV*) between the two big round values.
iii. The verifier system checks the coherency of all the rounds values obtained between the big round values.

C. The verifier checks the validity and the time for each of the web services responses included in the witness certificate (Witness Certificate Check) [Operation F].

Note: This process can be repeated for each of the web services responses, until the verifier is convinced by the truth of the indicated time.

i. The verifier system accesses the log of the STA to get the web services responses (Witness certificate) of the round (or the verifier can get it from the user).
ii. The verifier system checks the correctness of STA signature over the witness certificate.
iii. For each of the responses in the certificate, the verifier checks the validity of web service signature over its response (has produced using valid key) by checking the certification authority of that signing key.
iv. The verifier system checks that the information in the response matches the information in the linking certificate.
v. The verifier checks that, the time specified by the STA in the linking certificate (time of the round) is close enough to the time appended by the web service (time it receive the request) in the witness certificate.

4. Security Evaluation and experiment results

In spite of the non-availability of a systematic procedure to evaluate timestamping schemes, we tried to evaluate our system from security point of view depending on logic and an effort done through Une, Tsutomu and Masashi (section 2.4) [16-18].

4.1 Security Evaluation

Using result of Une, Masashi, Tsutomu Matsumoto, we find that thanks to our employed verification procedure our scheme deserved the highest security grade which is **Class 10** (**Section 2.4**), this was due employing our system to verification operations ***abdef***.

Next a list of all the possible attacks the any timestamping system may encounter; and show how our system addresses it: (due to the employed verification algorithm):

Attack: *STA backdate or postdate a request (Collusion with a requester):*
Solution:

1. The linking scheme used to generate the linking certificate prevent the issuer from inserting a request in pervious (past) position, because this would lead to different HV_i and SHV_i.
2. Original round values (i, t_i, SHV_i) are witnessed and signed by web services
3. The publishing process for the **Super Hash Value** through newspaper prevents the issuer from altering it to a tailored value.

Attack: *Attacker impersonation of the issuer through:*

1. Faking STA signature
2. Faking signatures of all the web services.

Solution:

1. Signing key length secure enough (1024 bit), it's a algorithm (RSA signature)
2. The signing key is periodically changed.

Attack: *The possibility of attacker collusion with or impersonation of web services.*

Solution:

1. The algorithm used to select the witnesses ensures randomness of their selection (Not Predetermined)
2. The list of candidate web services are priory established and published.
3. Signing key length secure enough (1024 bit).
4. The signing key is periodically changed in order to eliminate the possibility of key breaking.
5. Moreover, it would be ***impossible task*** to forge all that key of all the web services.

Attack: *The possibility of issuer manipulation of a certain reques.*

Solution:

1. Enabling the requester to send his request in an anonymous way.

4.2 Experiment Results

In order to ensures that our system does not only achieve an acceptable security level, but also the system will be running using an acceptable level of computing recourses, we implemented the system outlined in section 3.4.

4.2.1 Experiment Details

The code of our timestamping system is built using the following technologies:

- *C#* for *Win® Application.*
- *. Net®* Web Service.

It has been runs on a Dell PowerEdge™ 1855 server with:

- 2 dual core 64-bit Intel® Xeon® processors.
- 8GB ECC DDR-2 SDRAM.

4.2.2 Experiment Setup

- The proposed timestamping service (STA) was located at the server mentioned above.
- A number of web services (Witnesses) that are responsible for appending current time and sign were implemented and located at different machines each with a static IP address.
- We were able to get 10 static IP addresses, and we located 3 web services on each machine (each of them take a virtual address within the hosting machine). Hence, this enables us to select witnesses from 30 available web services registered each with an ID in system list of candidates.
- For our experiment, we set number of witnesses to be 12 randomly selected web services (out of 30) for each round.

4.2.3 Case Studies and Test Results:

We conducted massive experiments, for different values of system parameter, which are round time (window). In the experiment we flood our service with timestamping requests, and tested the service response for different values of the round duration. As a result we found that:

- Best performance: At window of 15 second.
- Throughput: of 2^{14} = 16384 document every 15 seconds. i.e. 3,932,160 documents per hour.
- CPU utilization and memory occupation: Never exceeded 65 and 80 percent, respectively.

The above results showed that the previously mentioned server was able to process 16384 documents each 15 seconds due to its processing capabilities. There was an ability to increase the throughput of the system by expanding the round time over 15 seconds, but we preferred this value because with the mentioned system specifications, it gives an acceptable level of utilization (CPU & Memory)

5. Conclusion

Timestamping is an important cryptographic technique used to bind an electronic document to a certain point in time. From our study to the existing techniques and solution, we find that there is a great need for an accountable solution and study in the distrusted trust area. That leads us to tailor our proposed system that actually can be considered as a hybrid system that make use of the benefits of the linking technique that gives an accountable level of security and absolute time technique that provides portability for the produced timestamp that produced in a fully trusted distributed manner that overcome shortages of the existing distributed scheme.

Our solution aimed to achieve longevity, wide trust and anonymity as main goals for a generic timestamping system, and our system achieved those goals through a number of techniques, algorithms and a number of assumptions. Our developed architecture was based on the usage of wireless communication as a tool to support anonymity. A payment method that makes use of scratch card was elaborated to our timestamping system to provides total anonymity to system users that are aim to hide their identity for whatever reason. This is also enhanced with an advanced mechanism for the client retrieval for a produced timestamp.

The usage of the web services concept as a methodology for building the system enables us from benefiting from its numerous advantages like portability, heterogeneity and facility of business integration. The presented idea of enabling a random selection of Internet entities (Web Services) to witness the process of timestamp production, gives the system a wider trust and longevity and overcomes the major drawback of other existing systems which is the total trust in a single entity.

The proposed system achieves the highest security levels (**Class 10)** due to the multiple safeguards that we aimed to impose into the system in order to overcome the problems that we found in the existing schemes/systems. The system also achieved that high security level without neglecting other important aspects like cost and processing power. We think that multi-server non-centralized schemes represent the right way to obtain efficient solutions in the domain of timestamping.

In our attempt to evaluate our proposed solution we build it, and we made experimental tests to adjust our system to its best parameter values. Our system achieved CPU utilization and memory occupation that never exceeded 65 and 80 percent, respectively, with a throughput of 3,932,160 documents per hour.

We think for further work that the use of modern cryptographic tools, like for example **threshold cryptography**, may achieve a higher reliability for the protocol. Also for **Authentic-Time Provision,** we aim in future to study and explore methods that can provide an authenticated time for any demanding application or machine over any network. Moreover a detailed study in the area of **anonymous communication** and **anonymous payment** is needed.

References

[1] S. Harber and W. Stornetta, "How to time-stamp a digital document", Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, Springer-Verlag, pp. 437– 455, 1990.

[2] J. Benaloh and M. de Mare, "Efficient broadcast time-stamping", Technical Report TR 91-1, Clarkson University, Department of Math and Computer Science, 1991.

[3] D. Bayer, S.Haber, and W. Stornetta, "Improving the efficiency and reliability of digital timestamping", In Sequences'91: Methods in Communication, Security, and Computer Science, Springer-Verlag, pp. 329-334, 1992.

[4] F. Pinto, V. Freitas, "Digital time-stamping to support non repudation in electronic communication", Proceedings SECURICOM '96, pp. 397-406, 1996.

[5] C. Adams, P. Pinkas, R. Zuccherato, "Time stamp protocols", Internet draft, Internet Engineering Task Force (IETF), July 1998. Available as http://www.ietf.org/internet-drafts/dreft-adams-time-stamp-02.txt, 2006.

[6] S. Harber, W. Stornetta, "Secure names for bit-strings", Proceedings of the 4th ACM Conference on Computer and Communications Security. ACM Press, April 1997.

[7] Ansper, Arne, A. Buldas, M. Saarepera ,J. Willemson, "Improving the availability of timestamping services", Proceedings of ACISP2001, pp. 360-375, Springer-Verlag, 2001.

[8] "Web Services Architecture", http://www.service-architecture.com/, 2006.

[9] S. Weerawarana, F. Curbera, F. Leymann, "Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More", Prentice Hall, 2005

[10] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). http//:www.ietf.org/html.charters/pkix-charter.html, 2006.

[11] ISO/IEC FDIS 18014-2. Information technology – Security techniques – Time-stamping services – Part :٢ Mechanisms producing independent tokens. Draft available at http:// oberon.postech.ac.kr/kiisc-sis/timestamp/, 2006.

[12] J. Benaloh and M. de Mare. "One-way Accumulators: A Decentralized Alternative to Digital Signatures". In T. Helleseth, editor, Advances in Cryptology - Proceedings of EuroCrypt '93, volume 765 of Lecture Notes in Computer Science, pages 274–285, May 1993.

[13] Ralph C. Merkle. "Protocols for public key cryptosystems". Proceedings of the IEEE Symposium on Security and Privacy, pages 122–134, 1980.

[14] Carlos Molina- Jimenez, Lindsay Marshall, "Anonymous and Confidential Communications from an IP Addressless Computer", Proceedings of the International Symposium on Handheld and Ubiquitous Computing (HUC '99), vol. 1707, pp. 383-385, 1999.

[15] Yi-Bing, Ming-Feng Chang, Herman Chung-Hwa Rao, "Mobile prepaid phone services", IEEE Personal Communications, vol. 7, no.3, pp. 6-14, January 2000.

[16] Masashi Une, "The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies", IMES Discussion Paper Series. Technical Report, 2001.

[17] Une, Masashi, Tsutomu Matsumoto, "Relations between Security and Verification Procedures of Time Stamps", Proceedings of the 2001 Symposium on Cryptography and Information Security, The Institute of Electronics, Information and Communication Engineers, pp.629-634, 2001
[18] Une, Masashi, and Tsutomu Matsumoto, "Ten Security Classes of Time Stamping Schemes", IEICE Technical Report, ISEC2001-38, The Institute of Electronics, Information and Communication Engineers, pp.141-148, 2001.
[19] Preneel, Bart, Bart Van Rompay, Jean Jacques Quisquater, Henri Massias and J.S.Avila, "Design of a timestamping systems", TIMESEC Technical Report WP3, 1998.
[20] Fabrica Nacional de Moneda y Timbre, PKITS: Deliverable D4a Description and Results of the Unstructured Data Time-Stamping Protocol Implementation, http://www.fnmt.es/pkits/, 2006.
[21] Surety.com. "Secure Time/Data Stamping in a Public Key Infrastructure", http://www.surety.com/home/pki.pdf, 2001.
[22] "FirstUse.com", http://www.firstuse.com/, 2006.
[23] "Cybernetica", http://www.cyber.ee/english/rd/, 2006.
[24] "DigiStamp", http://www.e-timestamp.com/, 2006.
[25] "Stamper, a PGP digital timestamping service", http://www.itconsult.co.uk/stamper.htm, 2006.