Signtiming Scheme based on Aggregate Signatures

Duc-Phong Le, Alexis Bonnecaze, and Alban Gabillon

***Abstract*—The aim of timestamping systems is to provide a proof-of-existence of a digital document at a given time. By timestamping a digital document after it has been signed by concerned parties, we know when the document was signed. Timestamping services also provide the non-repudiation and long term properties for digital signatures. Combining both digital signature and *provable* timestamping guarantees authentication, integrity and non-repudiation of digital documents. In this paper, we introduce such a service, so called signtiming, that signs and timestamps in a single step. Our scheme is based on an ID-based aggregate signature and is secure in the random oracle model.**

***Index Terms*—Timestamping Scheme, Digital Signature, Aggregate Signature, Identity-based Signature, Pairing**

I. Introduction

Digital signatures are used to provide authentication of digital documents. They have been developed for different applications such as secure and legally binding information exchange. In many applications like E-voting, E-commerce or E-administration, signature must be completed by a provable dating of the document. In the context of E-voting, it is indeed important for a voter to be able to prove that she or he voted before the end of the session. Digital contracts should also be associated a date and time where they were signed by concerned parties. Timestamping is widely recognized as an important technique used to certify that an electronic document was created, modified or signed at a certain point in time. Timestamps are also helpful to solve problems related to repudiation and long term of digital signatures. In fact, a private key may be compromised and therefore the digital signature for itself is not sufficient to warrant non repudiation. In order to repudiate a signature, a malicious signer may even claim that his credentials were already compromised when the signature was issued. Digital signatures would also lose their validity if the signing keys were revoked, even if the public key does not compromise or the signature scheme is not broken.

In the literature, timestamping schemes have been proposed independently of digital signature schemes. However, in real life, manuscript documents (patent, will, purchase orders, ...) are always dated and signed at the same time. Digital document should be dealt with the same philosophy. This combination would allow us to simultaneously answer two questions: "*Who is the author of the document?* " and "*When was the document created, last modified or signed?* ". It would guarantee integrity, authentication of document, and non-repudiation.

In this paper, we propose a so called *signtiming service*. Our scheme is to simultaneously perform in a single step both functions of digital signature and provable timestamp of a document. A trivial way to implement such a service is as follows: one who wants to sign a message m adds the time t into m, signs this concatenation and then sends to the TSA (Time-Stamping Authority). The TSA checks the time t and the validity of this signature. If so, the TSA certificates by generating a timestamp for this signature. This implementation requires the trust of the TSA. Besides, the TSA should store all timestamp issued for verifying after. Further, the TSA does not provide the *off-line comparability* property for timestamps. In order to eliminate these drawbacks, we propose a linking signtiming scheme. Our scheme is based on an aggregate signature scheme introduced by Gentry and Ramzan in [GR06]. The idea is to aggregate all the signatures received by the TSA (a trusted third party) in a period of time and produce a aggregate signature that depends one-way on all these signatures. The aggregate signature is then published, corresponding to that period of time. This aggregate signature serves as a witness, and is used during the verification phase. This value also allows us to verify timestamps in an off-line manner, i.e. the verification is only based on the timestamps itself, there needs no interaction with the timestamping service. By both timestamping and signing documents, signtiming prevents adversaries to back-date/forward-date or repudiate a digital signature.

The paper is organized as follows. In Section III, we recall the notion of timestamping schemes and their security requirements. Section II briefly introduces existing timestamping schemes, with their specific advantages and their weaknesses and aggregate signatures. In Section IV, we present our solution to simultaneously sign and timestamp a document. Then, we analyze the security of our scheme and conclude.

II. Preliminary

A. Timestamping Scheme and its Security Requirements

1) Timestamping Schemes: A timestamping scheme is generally made up of three parties: *Clients*, *Time-Stamping Authority (TSA)*, *Verifiers*.

This work was supported by Conseil Général des Landes and the French Ministry for Research under Project ANR-07-SESU-FLUOR.

D-P. Le is Ph.D. student at laboratoire LIUPPA, Université de Pau, B.P. 1155 64013 Pau France. (Contact him at phone: +33 5 58 51 37 18; e-mail: duc-phong.le@etud.univ-pau.fr).

A. Bonnecaze is with laboratoire IML, Université de la Méditerranée, 13288 Marseille France. (Contact him at phone: +33 4 91 82 86 77; e-mail: alexis.bonnecaze@esil.univmed.fr).

A. Gabillon is with laboratoire GePaSud, Université de la Polynésie Francaise, 98702 FAA'A - Tahiti - Polynésie francaise (Contact him at phone: +689 803 880; e-mail: alban.gabillon@upf.pf.

Most of the existing timestamping schemes are linking and based on a notion of round. Such a scheme typically consists of three phases:

- **Aggregation :** In this phase, the TSA collects all requests in a round, constructs a authenticated dictionary and computes a round token that depends on all requests received in this round.
- **Timestamps generation :** The TSA generates timestamps for requests. Each timestamp consists of the round token, the hash value of the document timestamped and an authentication path proving that the round token depends on the hash value.
- **Publishing :** The TSA publishes the round token and other information concerning this round on a newspaper. The published values allow us to verify the issued timestamps.

There exists two major methods to aggregate requests (generate the commitment) in a round. The first method uses a tree-like data structure such as the Merkle tree (e.g. in [HS91], [BHS93]), the threaded authentication tree [BLS00] or the skip list [BG05]. This method allows us to reduce to a logarithmic factor the amount of information to be stored and the verification consists in rebuilding a half of the tree. In the other methods, the TSA uses a one-way accumulator [BdM94] which represent an (algebraic) alternative to the aforementioned data structures. Using these functions, the size of timestamps and computation time are constant with respect to the number of requests in the round. Besides that, the verification process can be done in only one operation. Accumulator functions which are generally used are modular exponentiation. Unlike these existing methods, in this paper, we make use of a aggregate signature scheme for this purpose.

2) Security Requirements: The security objectives of timestamping schemes is to guarantee the properties: *correctness*, *data integrity* and *availability*. The first property requires that the verification succeeds only if a timestamp issued by the TSA is correct. The second means that an adversary cannot tamper the timestamp by back or forward dating it, or modify the request associated to it, or insert an old timestamp in the list of timestamps previously issued. The last property means that timestamping and verification must be available despite processes failures.

In general, there exist two major types of attacks on timestamping protocols: *back-dating* and *forward-dating* attacks. In the former attack, an adversary may try to "back-date" the valid time-stamp. This is a fatal attack for applications in which the priority is based on descendant time order (e.g. patents ...). The adversary may corrupt the TSA and may try to create a forged but valid timestamp token. In the later, an adversary may try to "forward-date" the timestamp without the approval of the valid requester. This is a fatal attack for applications in which the priority is based on ascendant time order (e.g. Will ...).

The forward-dating attacks can be prevented by requiring the client's identity which allows us to determine who had timestamped the document. This type of attacks was analyzed in more details by Matsuo and Oguro in [MO04]. Thus, security problems of timestamping systems only concentrate on back-dating attacks. The simple protocol is clearly not secure against back-dating attacks if TSA is corrupted, but the linking protocol is since a verifier can check the validity by computing the chain of hash values using published hash values.

B. Bilinear map

Our signtimig scheme makes use of a bilinear map, which is often called a pairing, to implement a decision procedure for the *Diffie-Hellman problem*. Typically, the pairing used is a modified Weil or Tate pairing. In this section, we briefly review the necessary facts about bilinear maps.

Let G_1 be an cyclic (additive) group of a prime order q and G_2 be a multiplicative group of the same order q. A map $e : G_1 \times G_1 \rightarrow G_2$ is called a cryptographic bilinear map if it satisfies the following properties:

1) **bilinearity:** for all $P,\ Q \in G_1$ and $a,\ b \in \mathbf{Z}_q,\ e(aP,\ bQ) = e(P,\ Q)^{ab}$
2) **non-degeneracy:** $e(P, P)$ is a generator of G_2 and therefore $e(P, P) \neq 1$
3) **computable:** there exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Earlier bilinear pairings, namely Weil pairing and Tate pairing of algebraic curves were used in cryptography for the MOV attack [MOV93] (Weil pairing) and FR attack [FR94] (Tate pairing). These attacks reduce the discrete logarithm problem on some elliptic or hyperelliptic curves to the discrete logarithm problem in a finite field. Modified Weil Pairing [BF01] and Tate Pairing [BLS04], [BGHS07] are examples of cryptographic bilinear maps. The later pairing is, in practice, much more efficiently computable than the former. On algebraic curves in particular, such pairings are very efficiently computable using Miller's algorithm [Mil04]. Currently, active research is being carried out to obtain efficient algorithms to compute pairings.

III. State of the Art

A. Timestamping Schemes

Here we are interested in *provable* timestamping scheme. Indeed, it is easy to concatenate a date/time to a given document but it is not simple to timestamp it in such a way that the timestamp can prove the correctness of that date. The first time-stamping schemes were presented in the early 90's by Haber and Stornetta [HS91] and Benaloh and de Mare [BdM91]. In the years that follow, a lot of new schemes were proposed and their security analyzed [MQ97], [BLLV98], [BL98], [MAQ99], [Jus98].

Most of the existing systems rely on a centralized server model that has to be trusted. The idea behind existing timestamping schemes is to prevent the server from forging fake timestamp tokens by linking the tokens in a chronological chain (see for example [HS91]). Periodically, a token is published in an unalterable and widely witnessed media like a newspaper. This scheme offers the following advantages:

- The publication provides us with an absolute time.

- After a token has been published at time t, the server cannot forge a fake timestamp token former to time t.
- Since tokens are linked in a chronological chain, we can obtain a relative dating of the requests submitted between two publications.

However, this scheme has the following drawbacks:

- Before the next publication, the server can tamper the tokens which have been issued since the last publication.
- The entire chronological chain must be stored for verification.

In order to reduce the amount of information to be stored (for verification) and to improve system scalability, most of the protocols use an aggregation scheme. In this case, it is not required to store all the timestamping tokens in order to allow their verification. An aggregation scheme uses the notion of round: a round is generally a period of time Δt. The TSA stores for each round a unique value called the aggregation value and then it sends to the requesters all the necessary information to re-compute this value. Aggregation schemes do not give tamper-evident information about the chronological order of timestamping requests processed in the same round. A timestamping protocol using an aggregation scheme, may build a chain by chronologically linking the aggregation values obtained between two publications. The efficiency of the protocol depends on the properties of the aggregation scheme. We now introduce the two major aggregation scheme families.

The first family gathers schemes which use tree-like data structure. The most famous structure is the Merkle Tree (recall that a Merkle Tree is a data structure introduced by Ralph Merkle in 1979 [Mer79] to build secure authentication and signature schemes from hash functions). This method allows us to reduce to a logarithmic factor the amount of information to be stored and the verification consists in rebuilding a half of the tree. However, protocols using such a scheme are not always accurate and efficient. For example, when the number of timestamped documents is very small while the frequency of publication is very low (typically a week), the accuracy of the timestamp may not be satisfying. Notice also that a scheme using a binary tree is not efficient when the number of documents is not close to a power of 2. The worst case is being reached when this number is $2^n + 1$.

Recently, Blibech and Gabillon [BG05] proposed to use a new structure called skip-list (developed by Bill Pugh [Pug90]). A skip-list is a data structure that can be used in place of a balanced tree. Algorithms for insertion and deletion in skip lists are simpler and faster than equivalent algorithms for balanced trees. Their scheme is totally ordered, i.e. it allows us to compare two timestamps even in the same round.

The second family of schemes uses accumulator functions. Accumulator functions [BdM94] represent an (algebraic) alternative to the aforementioned data structures. Using these functions, the verification process can be done in only one operation. Moreover, the amount of information that has to be stored does not depend on the number of timestamped documents. Accumulator functions which are generally used are modular exponentiation.

All these round based schemes need to regularly publish the round tokens in a widely distributed media.

A distributed approach based on a network of trusted servers is proposed by Bonnecaze et al. [BLGB06]. The objective of their scheme is to construct a multi-server timestamping scheme against denial of service attacks. Even though it is secure, the high number of interactions between servers makes the scheme difficult to implement. In [Bon06], Bonnecaze proposed a scheme based on multisignatures without process of publication. This scheme is reliable, robust and efficient but requires a wide area network.

In this paper, we choose, for the sake of simplicity, to define a system relying on a centralized server model. We adopt a new aggregate system based on aggregate signatures. Publication of round token uses a database, which may be replicated for better availability.

B. Aggregate Signatures

An aggregate signature scheme is a digital signature that supports aggregation. It allows a collection of signatures to be compressed into one short signature. This single signature along with a given original message $m_i, 1 \leq i \leq n$ and the list of signers will convince the verifier that user i indeed signed message $m_i, 1 \leq i \leq n$. Ideally, the length of the aggregate signature (excluding the messages and the public keys of the signers) should be constant, independent of the number of signed messages.

The concept of *aggregate signature* was introduced by Boneh et al. in [BGLS03]. Their constructions are based on BLS short signature scheme [BLS01] in groups with efficiently computable bilinear maps. Their scheme is called *general aggregation scheme* since aggregation can be done by anyone and without the cooperation of the signers. Subsequently, Lysyanskaya et al. [LMRS04] proposed a RSA-based *sequential aggregate signature* scheme. In a sequential aggregation scheme, signature aggregation can only be done during the signing process. Each signer sequentially modifies the aggregate signature in turn by adding her signature to the current aggregate. Both above schemes are provably secure in the random oracle model. Recently, Lu et al. [LOS$^+$06] proposed a sequential aggregate signature scheme without random oracle based on Waters signature [Wat05]. The security of their scheme relies on the hardness of the *Computational Diffie-Hellman* (CDH) problem in bilinear groups.

The total information needed to verify the aggregate signature must include individual signers' public keys, whose lengths depend on the security parameters of the scheme. Since the verifier cannot be expected to know all n signers' public keys, practically, the length of an aggregate signature is not significantly shorter than the length of n traditional signatures. Hence, for large value of n, it is preferable to specify the signers by their identities.

In [GR06], Gentry and Ramzan introduce an identity-based aggregate signature (IBAS) which is secure in the random oracle model. The use of an identity-based scheme has mainly two advantages (for more information about identity-based schemes, see [Sha85]):

1) Communication-efficiency: the signer does not need to send an individual public key and certificate with its signature.

2) All the signers are clients of the same Private Key Generator (PKG). Hence, the verifier needs only one traditional key (the one of the PKG) to verify identity-based signatures on multiple documents.

Moreover, the scheme of Gentry et al. produces short signatures and is efficient computationally. Verification requires only three pairings computations, regardless of the number of signers.

For these reasons, we base our signtiming scheme on this IBAS.

IV. Timestamping Scheme based on Aggregate Signatures

In this section, we present our signtiming scheme based on the aforementioned aggregate signatures. The entities of the scheme are

1) the **Private Key Generator**: the PKG generates signer private keys from its *master secret* and the signer's identity; It has a traditional public key;
2) the **Time Stamping Authority**: the TSA is the trusted third party which is used to timestamp a document. In our scheme, the TSA will mainly aggregate the signatures after having checked the correctness of the time, send a token to the signer and publish a round token;
3) the **signers** sign their own document and send each signature to the TSA. A signer can be considered as a client: she wants to obtain a token from the TSA to be able to prove the authenticity of her document;
4) the **verifier** verifies the validity of the timestamp from an individual token and the round token.

In reality, the PKG can itself offer the timestamping service. The PKG can thus play the role of the TSA.

A. Protocol

1) Timestamp Procedure: We suppose that time is discretized in rounds of length Δt and all the signers are synchronized with the TSA. Each round is identified by an absolute date t. For example, a round can be identified by: August 4^{th} 2005 at 12.04am. We then describe the scheme.

1) Setup: The Private Key Generator (PKG) generates parameters and keys:
   a) generates groups G_1 and G_2 of prime order q with admissible pairing $e : G_1 \times G_1 \to G_2$;
   b) chooses an arbitrary generator $P \in G_1$;
   c) picks $s \in_R Z_q$ and sets $Q_1 = sP, Q_2 = s^2P$;
   d) chooses cryptographic hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to Z_q$ and H a traditional hash function like SHA_1.

   The PKG's public key is $(G_1, G_2, e, P, Q_1, Q_2, H_1, H_2, H)$ and its secret is s.
2) Signer with identity ID_i and his private keys (which were generated by the PKG), $d_{i,j} = sP_{i,j}$ and $v_{i,1} = (1/s)P_{i,1}$, where $P_{i,j} = H_1(ID_i, j) \in G_1$ for $j \in \{0,1\}$
   a) computes $w_t = H(t)$
   b) computes $P_{w_t} = H_1(w_t) \in G_1$;
   c) computes $m_i = H(D_i)$, where D_i is the document the signer wants to have timestamped;
   d) computes $c_i = H_2(m_i, ID_i, w_t) \in Z_q$;
   e) generates random $r_i \in Z_q$;
   f) computes its signature $\sigma_{i,t} = (w_t, S'_i, T'_i)$, where $S'_i = r_iP_{w_t} + d_{i,0} + c_id_{i,1}$, $T'_i = r_iP$.
   g) sends $\sigma_{i,t}$ and $V'_i = c_iv_{i,1}$ to the TSA.
3) TSA checks whether equality $w_t = H(t)$ holds. If this is not the case, TSA sends an error message to the signer.
4) TSA aggregates all the received signatures at round t to obtain (w_t, S_t, T_t) where $S_t = \sum_{i=1}^n S'_i$, and $T_t = \sum_{i=1}^n T'_i$.
5) TSA publishes the value $H_t = H(S_t, T_t, w_t)$ corresponding to round t and returns to the signer with identity ID_i the token $J_{i,t} = (t, S_t, T_t, L, SV_i)_{TSA}$, where L is a description of the list of the n signers' identity and $SV_i = \sum_{k\in[1,n],k\neq i} V'_k \in G_1$.

2) Verification Procedure: Someone who wishes to verify the timestamp $J_{i,t}$ for the document D_i will:

1) Compute the hash of the document $H(D_i)$ and check that it is part of the data that reconstructs the timestamp for the round.
2) Seeks in the database the value H_t corresponding to round t.
3) From the value $J_{i,t} = (t, S_t, T_t, L, SV_i)$, the verifier
   a) checks whether $H_t = H(S_t, T_t, w_t)$
   b) and checks the following equality:

$$e(S_t, P) = e(T_t, P_{w_t})e(Q_1, \sum_{k=1}^{n} P_{k,0} + \\ + c_iP_{i,1})e(Q_2, SV_i), \quad (1)$$

where $P_{w_t} = H_1(w_t)$ and $c_i = H_2(m_i, ID_i, w_t)$.

The token is declared valid when all steps are successfully performed.

B. Correctness of the Scheme

It is quite easy to see that this property is achieved. The verification procedure holds if and only if the signtiming certificate is true. In fact, the verification equation (1) is satisfied:

$$e(S_t, P) = e(\sum_{k=1}^{n} S'_i, P) = e(\sum_{k=1}^{n}(r_iP_{w_t} + d_{i,0} + c_id_{i,1}), P)$$

$$= e(\sum_{k=1}^{n} r_iP_{w_t}, P)e(\sum_{k=1}^{n} d_{i,0} + c_iP_{i,1}, P)e(\sum_{k=1,k\neq i}^{n} c_id_{i,1}, P)$$

$$= e(T_t, P_{w_t})e(Q_1, \sum_{k=1}^{n} P_{k,0} + c_iP_{i,1})e(\sum_{k=1,k\neq i}^{n} c_iv_{i,1}, Q_2)$$

$$= e(T_t, P_{w_t})e(Q_1, \sum_{k=1}^{n} P_{k,0} + c_iP_{i,1})e(Q_2, SV_i)$$

C. Discussion

Note that, our aggregate signature is slightly different from that of Gentry [GR06]. To verify the tokens, we need the hash values m_i of the documents sign-timestamped in the round which would be sent along with the final token to clients. Those hash values are different from each other. This would cause the size of the token to grow linearly with n. In order to eliminate this disadvantage, our scheme requires each signer to send to the TSA $V_i' = c_i v_{i,1}$ instead of m_i, then the TSA computes the value $SV_i = \sum_{i\in[1,n],i\neq k} V_k'$ and then send it to the client i for the purpose of verification.

Our scheme is basically non-interactive and does not need the cooperation of other clients to verify the timestamp. Length of the tokens does not depend on the number of signtimed documents. The published round token is a binary string (typically 160 bits if the hash function H is SHA_1) and the token of a particular document is represented by three points, a representation of the time t and the list of signers L. Length has to be compared to the token length of existing time-stamping schemes. The most common schemes use Merkle trees and have tokens whose lengths are logarithmic with respect to the number of documents in a round. Skip lists based schemes lead to a similar length. Hence, our protocol is more efficient in term of token length. In term of computation, the protocol requires four pairing computations for verification. Note that every modern computer is able to execute the verification in less than one second.

The use of identity based concept both simplifies the protocol and makes the tokens short. On the other hand, the private key generator knows the master key and may become a threat. We suggest the use of a distributed key generation protocol such as that of Pedersen [Ped91] or Gennaro et al. [GJKR07]. In these protocols, a master key is generated in a distributed way. Each of the m PKGs constructs a random share. At any given time only a part of the m PKGs must be online in order for a client to retrieve his private key.

Regarding the protocol itself, it is important to see that the TSA cannot cheat when sending the value SV_i to the signer. Actually, finding a value SV_i (which verifies the equality of the verification) without the knowledge of the secret value s is as difficult as breaking Diffie-Hellman problem.

V. Security Analysis

A. Security Requirements

There exist two major types of attacks on timestamping protocols: *"back-dating attack"* and *"forward-dating attack"*. In the back-dating attack, an adversary may try to back-date a valid timestamp. This is a fatal attack for applications in which the priority is based on descendant time order, e.g. in the case of a patent submission. The adversary may corrupt the TSA and collude with the TSA to create a forged but valid timestamp. In another configuration, the adversary, without colluding with the TSA, may create a fake timestamp, which can be successfully verified due to the correctness of the proof, by either eavesdropping or pre-computing. The simple protocol [ACPZ01] is clearly not secure against TSA corruption, but the linking protocol is.

In the forward-dating attack, an adversary may try to forward-date the timestamp without the approval of the valid requester. This is a fatal attack for applications in which the priority is based on ascendant time order. For example, an adversary will use this type of attacks if he wants to deal with the Will.

B. Analysis

The security of our signtiming scheme follows directly from the security of the identity-based aggregate signature scheme in [GR06] which is provably secure against existential forgery under an adaptively chosen-message and chosen-identity attack in the random oracle model. In this section, we demonstrate that the scheme is secure against two major types of attacks on the timestamping protocols: *back-dating* and *forward-dating*.

Theorem 5.1: If the identity based aggregate signature scheme is secure, then the signtiming scheme described above is secure.

Back-dating: Consider the following situation, Alice, an inventor, needs to timestamp her patent d at time t. After some time, the invention is disclosed to the public. Bob, an adversary tries to steal the right to Alice's invention. He slightly modifies Alice's invention (at least the author's name should be replaced), and tries to back-date it relative to Alice's invention. Bob is successful if he can construct a signtimestamp of the modified invention d' such that it can be verified at time t' ($t' < t$). Bob can then use this signtimestamp to claim his rights to the invention.

We consider two cases, the adversary cannot collude with the TSA; the adversary can collude with the TSA.

In the first case, an adversary tries to back-date a document (invention) using a published round token and/or some valid signtimestamps of this round he received at time t' in the past. This is impossible if the ID-based aggregate signature scheme in [GR06] is existential unforgeable.

In the second case, if the adversary colludes with the TSA, then he can obtain a valid signtimestamp token for any document at any time. We assume that the adversary can learn the content of the patent either after disclosure of the inventor or after the publication of the round token applying to the invention. In order to backdate the invention, the adversary, first provided an aggregate signature, all signatures and other information concerning the aggregate signature, tries to re-generate this aggregate signature so that it contains his signature on the invention. This is impossible if the above ID-based aggregate signature scheme has the property of existential unforgeability.

Forward-dating: Consider a client who has sign-timestamped the hash of his will which initially favors the adversary. After some time the will is updated, writing the adversary out and then is sign-timestamped again. Assuming that the adversary has access to the hash of the original will, he can once again re-register the hash of the original will. The TSA signtimestamps the hash of the "first" will and then sends to the adversary a token which can be used to prove authenticity of the "first" will.

These attacks can be prevented by requiring the client's identity. This allows us to determine who had timestamped the document. This type of attack was analyzed in more details by Matsuo and Oguro in [MO04]. In fact, in order to carry out a forward-dating attack, the adversary, after obtaining the original will, may try to forge the identity of the client. In our signtiming scheme, the signature of each client guarantees the authentication of document. Hence, the adversary can forward-date the document only if he can break digital signatures of the honest client.

Besides above dating attacks, an adversary can also make attacks concerning time reference of the signtiming scheme such as: masquerade, delay, replay [MQ97]. In order to prevent these attacks, we recommend using the Network Time Protocol (NTP).

VI. Conclusion

In this paper, we presented a new service called signtiming. Its goal is to sign a document and securely time-stamp it at the same time. This new service provides authentication, integrity and a provable dating. Our scheme makes use of an identity based aggregate signature. Hence, there is no problem of key management and the aggregate signature is really efficient. Since our scheme can easily be implemented, it is expected to be helpful in many applications.

References

[ACPZ01] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), 2001.

[BdM91] J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report 1 TR-MCS-91-1, Clarkson University Department of Mathematics and Computer Science, August 1991.

[BdM94] J. Benaloh and M. de Mare. One-way accumulators: a decentralized alternative to digital signatures. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 274–285, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[BG05] K. Blibech and A. Gabillon. Chronos: An authenticated dictionary based on skip lists for timestamping. In *Proc. of 12th ACM Conference on Computer Security (Workshop Secure Web Services)*, George Mason University, Fairfax, VA, USA, November 2005.

[BGHS07] P. S. Barreto, S. D. Galbraith, C. O. Heigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps, 2003.

[BHS93] D. Bayer, S. Haber, and W. S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, pages 329–334, London, UK, 1993. Springer-Verlag.

[BL98] A. Buldas and P. Laud. New linking schemes for digital time-stamping. In *ICISC*, pages 3–13, 1998.

[BLGB06] A. Bonnecaze, P. Liardet, A. Gabillon, and K. Blibech. Secure time-stamping schemes: A distributed point of view. *Annals of Telecommunications*, 61(5-6):662–681, May-June 2006.

[BLLV98] A. Buldas, P. Laud, H. Lipmaa, and J. Villemson. Time-stamping with binary linking schemes. In *CRYPTO*, pages 486–501, 1998.

[BLS00] A. Buldas, H. Lipmaa, and B. Schoenmakers. Optimally efficient accountable time-stamping. In *Public Key Cryptography*, pages 293–305, 2000.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, London, UK, 2001. Springer-Verlag.

[BLS04] P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptol.*, 17(4):321–334, 2004.

[Bon06] A. Bonnecaze. A multi-signature for time stamping scheme. In *SAR/SSI 06: The 1st Conference On Security in Network Architectures and Information Systems*, Seignosse, France, June 2006.

[FR94] G. Frey and H-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[GJKR07] R. Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, 2007.

[GR06] C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273. Springer, 2006.

[HS91] S. Haber and W. S. Stornetta. How to Time-Stamp a Digital Document. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 437–455, London, UK, 1991. Springer-Verlag.

[Jus98] M. Just. Some timestamping protocol failures. In *NDSS 98: Proceedings of the Symposium on Network and Distributed Security*, pages 89–96, San Diego, CA, USA, March 1998.

[LMRS04] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027/2004 of *Lecture Notes in Computer Science*, pages 74–90, New York, NY, USA, 2004. Springer Berlin / Heidelberg.

[LOS+06] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, pages 465–485, 2006.

[MAQ99] H. Massias, X. Serret Avila, and J.-J. Quisquater. Timestamps: Main issues on their use and implementation. In *WETICE '99: Proceedings of the 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, pages 178–183, Washington, DC, USA, 1999. IEEE Computer Society.

[Mer79] R. C. Merkle. *Secrecy, authentication, and public key systems.* PhD thesis, 1979.

[Mil04] V.S. Miller. The Weil Pairing, and Its Efficient Calculation. *J. Cryptol.*, 17(4):235–261, 2004.

[MO04] S. Matsuo and H. Oguro. User-side forward-dating attack on timestamping protocol. In *Proc. of the 3rd International Workshop for Applied Public Key Infrastructure (IWAP'04)*, pages 72–83, 2004.

[MOV93] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MQ97] H. Massias and J. Quisquater. Time and cryptography. Technical report, Universit'e catholique de Louvain, 1997.

[Ped91] T. P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *EUROCRYPT*, pages 522–526, 1991.

[Pug90] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990.

[Sha85] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT '05*, pages 114–127, 2005.