One-Way Signature Chaining: A New Paradigm For Group Cryptosystems and E-Commerce

Amitabh Saxena, and Ben Soh
Dept. of Computer Science and Computer Engineering
La Trobe University, Bundoora, VIC, Australia 3086

Abstract— **In this paper, we describe the notion of signature chaining which was originally proposed in [1]. Signature chaining is essentially a method of generating proxy signatures. However, the difference from most proxy schemes is that in a chained signature, the proxies are generated sequencially rather than in parallel. The purpose of a chaining scheme is to 'link' many proxies in a chain of trust. We propose an efficient protocol using aggregate signatures that enables this to be done in an efficient and non-interactive manner. Our protocol is based on bilinear pairings and is secure against chosen ciphertext attacks under the Diffie Hellman assumption.**

I. Introduction

Over recent years, a lot of research in e-commerce systems has been on the problem of 'trust transfer'. However, the notion of trust itself is difficult to formalize. Despite this drawback, a low-level definition of trust based on cryptographic primitives (which to some sense is quite reasonable in a practical scenairio) can be used to design reliable distributed systems. We will try to give a 'formal' meaning to trust transfer using similar ideas. Trust transfer can be understood from 'proxy' signatures in which the original signer delegates the signing authority to a proxy signer [2]. In other words, the original signer 'transfers' his trust to the proxy. However, trust transfer can have other implications as well, for instance, in a mobile agent scenario [1], [3], the original platforms must convince all successive platforms about the trustworthiness of the agent code.

Informally, trust transfer is the act of transferring the trust placed on the original user (the *trusted*) to a proxy user (the *trustee*) such that some other user (the *truster*) can delegate the same responsibilities to the trustee that he would have delegated to the trusted in some *trust context* (which could be an agreement). Trust transfer makes sense only in a non-interactive environment where the original user is no longer available for interaction once the trust delegation is complete (in other words, the truster can only interact with the trustee).

Once the act of delegation is complete, the trusted and trustee are said to be connected in a *trust relationship*. It may be the case that a truster is a trustee as well. For instance, in an e commerce scenario, there may be multiple entities involved in a transaction such that interaction is not permitted (or feasible) between many of them. However, in order for the transaction to be successful, some sort of accountability is necessary. Thus in this scenario, a truster will act as trustee for the next entity in the chain such that trust is 'transferred' from the original user to the last user via the chain of trustees. More generally, where the participants of a distributed protocol (i.e. involving more than two entities) are required to authenticate to each other and non-repudiation is required, the number of interactions between participants need to be minimized.

In this paper, we propose the concept of *one-way chaining* to demonstrate how a chain of trustees can be constructed by forming individual trust relationships between the links of the chain without having to interact with any other links. Toward the end of the paper, we describe some interesting applications of this idea.

II. Background

We assume that users are identified by unique identifiers which could be an e-mail. In this paper, we assume that the user's name uniquely identifies the user (for instance, there is only one user named "Alice"). Thus if Alice communicates with Bob, Bob simply needs to know that he is talking to the person named "Alice".

A. One-Way Chaining

Imagine a scenario where many users are involved in an e-commerce transaction. Denote the contract spelling the details of the transaction by m. We assume that m is passed among the users who must approve of it by attaching a signature. Represent the users as points of a acyclic directed graph. As the message is passed, a new arc directed from the receiver to the sender is added to the graph. The edges of such a graph will represent a hop-by-hop path of the message in the reverse direction from the current host to the initiator. In this notation the statements "a passed the message to b" and "There is a path of unit length from b to a" are considered equivalent. We can consider this graph to describe the path by which trust is propagated in the system.

1. We say that a *direct* path exists from b to a if and only if b can prove (in the context of the message) something about a that no other host can. That is, b has some *extra* information about a that others cannot extract from b's proof.
2. Let $\{h_0, h_1, \ldots h_n\}$ be a set of hosts for some $n \geq 1$. We say a *chained* path exists from h_n to h_0 if and only if there exists a direct path from h_x to h_{x-1} for each x from 1 to n.

Assume that i is the initiator of the message, a is any sending host and b is the receiving host. Also, excepting the act of message transfer no other interaction is allowed between any hosts. Using this scenario, authentication can be defined as follows: a must prove to b that a chained path exists from a to i.

B. Definitions

(a) Fix the alphabet $\Sigma = \{0, 1\}$. The set of strings over Σ is denoted by Σ^* and the set of all strings over Σ of length $\leq l$ are denoted by Σ^l. For any $x, y \in \Sigma^*$ the symbol $x\|y$ denotes the concatenation of x and y. The empty string is denoted by the symbol ϵ.

(b) If a, b are two variables of the same type, then $a \leftarrow b$ denotes that a is set to the value of b. If $\mathbb{A}$ is a non-empty set, then $x \leftarrow \mathbb{A}$ denotes that x has been uniformly chosen in $\mathbb{A}$. Throughout this paper we will use the symbol $\mathbb{Z}$ to denote the set of integers and the symbol $\mathbb{I}$ to denote the set of all identities $\{I_1, I_2, \ldots\}$.

(c) A sequence of i elements $\alpha_1, \alpha_2, \ldots \alpha_i$ is denoted by $\langle \alpha_1, \alpha_2, \ldots, \alpha_i \rangle$. The *empty* sequence is a sequence without any elements and is denoted by $\langle \rangle$. If $S = \langle \alpha_1, \alpha_2, \ldots, \alpha_i \rangle$ is some finite sequence then $\langle S, \alpha \rangle = \langle \alpha_1, \alpha_2, \ldots, \alpha_i, \alpha \rangle$ is also a finite sequence. For any two sequences $S = \langle \alpha_1, \alpha_2, \ldots, \alpha_i \rangle$ and $T = \langle \beta_1, \beta_2, \ldots, \beta_j \rangle$, $S = T \Leftrightarrow (i = j \;\&\; \alpha_i = \beta_i \; \forall i)$. For any finite set $\mathbb{A}$, the symbol $\langle \mathbb{A} \rangle$ denotes the set of all sequences having (non-repeating) elements from $\mathbb{A}$.

(d) Fixed Strings: Let $\mathbb{L}_1$ and $\mathbb{L}_2$ be any two languages. For some $x \in \mathbb{L}_1$ and some $y \in \mathbb{L}_2$, the pair $\langle x, y \rangle$ is said to be *fixed* if and only if there exists a (polynomial-time computable) binary function $\sigma : \mathbb{L}_1 \times \mathbb{L}_2 \mapsto \{0, 1\}$ such that $\sigma(x, y) = 1$ and it is computationally intractable to find another string $\hat{y} \in \mathbb{L}_2$ such that $\sigma(x, \hat{y}) = 1$.

C. Bilinear Pairings

The fundamental building blocks of our protocol are a class of primitives known as *bilinear pairings*[1] defined as follows: Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of the same order. Assume that computing the discrete logarithm in both $\mathbb{G}_1$ and $\mathbb{G}_2$ is hard. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ that satisfies the following properties [10], [4]:

1) *Bilinearity*: $e(aP, bQ) = e(P, Q)^{ab} \; \forall P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$
2) *Non-degeneracy*: $P \neq 0 \Rightarrow e(P, P) \neq 1$
3) *Computability*: e is efficiently computable
   The above also imply: $e(P + Q, R) = e(P, R) \cdot e(Q, R) \; \forall P, Q, R \in \mathbb{G}_1$.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. Despite the complex mathematics involved in constructing such maps, cryptographic protocols based on pairings can be described entirely without ever referring to the actual implementation. We refer the reader to [10], [4], [11] for more details. Pairings and other parameters should be selected in proactive for efficiency and security. For appropriately selected parameters, the following problems are computationally intractable:

1) Discrete Logarithm Problem (DLP): Given $P, aP \in \mathbb{G}_1$, compute $a \in \mathbb{Z}_q$
2) Diffie Hellman Problem (DHP): Given $P, aP, bP \in \mathbb{G}_1$, compute $abP \in \mathbb{G}_1$.
3) Bilinear Diffie Hellman Problem (BDHP): Given $P, aP, bP, cP \in \mathbb{G}_1$, compute $e(P, P)^{abc} \in \mathbb{G}_2$.

While the following problem is always easy (see [10] for a proof):

1) Decisional Diffie Hellman Problem (DDHP): Given $P, aP, bP, abP, Q \in \mathbb{G}_1$, differentiate between abP and Q.

Our motivation to use pairings is due to the fact that the DDHP in $\mathbb{G}_1$ is easy while both the DHP and the DLP are hard in $\mathbb{G}_1$. Such groups (where DDHP is easy but DHP is hard) are generally referred to as as *Gap* Diffie Hellman (GDH) groups [12].

III. One-Way Signature Chaining

We will describe here a protocol that enables one-way chaining using chained signatures. A one-time initial setup is necessary during which our participants create a public-key directory. Once this setup is complete, Any registered member can participate in unlimited rounds of the protocol.

A. Initial Setup (Create PKI)

A public directory (or PKI) will be used to authenticate messages (and if necessary to encrypt them). The PKI we describe is based on bilinear pairings and a central authority is responsible for generating the security parameters. A trusted CA is responsible for certifying the public keys. To participate in the protocol each user must have a certified public key. The setup proceeds as follows:

1) Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ be a bilinear mapping as defined in section II-C. Let $P \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$. Also let $\mathcal{H} : \langle \Sigma^* \rangle \mapsto \mathbb{G}_1$ be a cryptographic hash functions. The parameters $\langle e, q, \mathbb{G}_1, P, \mathcal{H} \rangle$ are generated by the trusted authority and made public in an authentic way.
2) Each participant I_i generates $x_i \leftarrow \mathbb{Z}_q$ as the private key. The corresponding public key is $Y_i = x_i P$
3) Each participant who wants to sign messages obtains a certificate from the CA linking the identity I_i and the public key Y_i. In other words, the CA fixes the pairs $\langle Y_i, I_i \rangle$.

B. A Zero Knowledge Proof

We first describe a simple interactive zero knowledge proof and then use it to construct a chained signature scheme. In this system there are two participants, I_1 and I_2. The aim of the protocol is for I_1 to identify itself to I_2. This protocol for example could be used by the CA to ascertain that a user indeed knows the private key corresponding to the given public

[1] Bilinear pairings are probably best known for their use in Identity Based Encryption (IBE) by Boneh and Franklin in 2001 [4]. Other notable applications of pairings are Identity Based Signatures (IBS) [5], [6], [7], tripartite one-round key agreement [8] and Certificate-Less Public Key Cryptography (CL-PKC) [9].

key before handing out the certificate. Once a certified key exists, it can be used for identification.

1) The prover I_1 claims to know the discrete logarithm $x_1 \in \mathbb{Z}_q$ of $Y_1 \in \mathbb{G}_1$
2) The verifier I_2 generates a random $Q \leftarrow \mathbb{G}_1$ and sends it to I_1
3) I_1 responds with $R = x_1 Q \in \mathbb{G}_1$
4) I_2 accepts if $e(R, P) = e(Q, Y_1)$

The proof is zero knowledge based on the following arguments:

(a) Correctness: The properties of bilinear maps ensure that the verification is always successful if I_1 does not cheat.
(b) Soundness: If Q is truly selected uniformly from $\mathbb{G}$, the probability of computing R such that $e(R, P) = e(Q, x_i P)$ without knowledge of x_i is negligible assuming that the DLP in $\mathbb{G}_1$ is hard [10].
(c) Zero-Knowledge: The protocol is zero-knowledge if I_2 can generate the transcript (R, Q) without interaction with I_1 (that is, the protocol can be simulated). I_2 simply computes $Q = rP$ for some $r \leftarrow \mathbb{Z}_q$ and $R = r x_i P$.

We will now extend this zero knowledge proof to include a third participant I_3 in this scenario. The goal of the protocol is that I_1 and I_2 will simultaneously prove to I_3 that they know the discrete logarithm of Y_1 and Y_2 such that I_3 cannot be convinced about either of the two statements separately. That is, the proof is valid only on the two statements together: "I_1 knows x_1 and I_2 knows x_2" but not on any of the individual statements "I_1 knows x_1" or "I_2 knows x_2" independently of the other. Such a proof is called an *additive zero knowledge* proof [1].

1) The provers I_1, I_2 start by claiming to know the discrete logarithms $x_1, x_2 \in \mathbb{Z}_q$ of $Y_1, Y_2 \in \mathbb{G}_1$ respectively.
2) The verifier I_3 generates two random elements $Q_1, Q_2 \leftarrow \mathbb{G}_1$ and sends Q_1 to I_1 and Q_2 to I_2.
3) I_1, I_2 compute separately $R_1 = x_1 Q_1$ and $R_2 = x_2 Q_2$ respectively. Then they compute together the value $R = R_1 + R_2$ and send R to I_3.
4) I_3 accepts if $e(R, P) = e(Q_1, Y_1).e(Q_2, Y_2)$. We claim that this test will pass if and only if I_1 knows x_1 *and* I_2 knows x_2.

The correctness can be easily verified: if none of I_1, I_2 cheat, the verification process always passes. The soundness property holds because computing individual proofs $x_1 Q_1$, $x_2 Q_2$ is still infeasible without knowledge of x_1 or x_2 due to the hardness of the DHP in $\mathbb{G}_1$ as shown in theorem 4.4 of [13] (cf. aggregate extraction). If the verification process fails it is not possible to tell which of I_1 or I_2 (or both) cheated. The protocol is still zero knowledge because the transcripts (Q_1, Q_2, R) can be generated by I_3 as follows: first generate $r_1, r_2 \leftarrow \mathbb{Z}_q$. Then compute $Q_1 = r_1 P$, $Q_2 = r_2 P$ and $R = r_1 Y_1 + r_2 Y_2$.

C. *The Authentication Protocol*

We will extend the above zero knowledge scheme to an arbitrary number of users and use the Fiat-Shamir heuristic [14] to turn the interactive identification protocol to a non-interactive chain-signature scheme. We assume that:

1) The message (or contract) to be signed is m. The identities of the first n participants is the ordered sequence $\langle I_1, I_2, \ldots I_n \rangle$ where $I_i \in \mathbb{I}$ for each positive integer i. A further restriction is that all the identities must be unique. The users signing the contract have to follow the order specified. The verification process must succeed if and only if the correct order is given as input. Such a signature is called a chain-signature.
2) At the time of signing, the participants are not allowed to interact except with their immediate neighbours in the list. Moreover there is only one interaction allowed, that of passing the message on to the next recipient. The chain-signature must be attached with the message during transfer. Participants can add more links to the chain but cannot remove any previously added links.
3) The order of the participants is 'ad-hoc'. It is not possible for any user I_i to precisely determine the identity of the next user I_{i+1} during signing which implies that the contract can involve entities that need not be aware of the other entities.

An arbitrary participant I_i will process the message as follows: On receiving it from I_{i-1}, it first follows the verification procedure. Before passing the message to I_{i+1}, it follows the signing procedure. The first participant I_1, however, only follows the signing procedure. Before describing the process, we give some additional notation:

A valid chain-signature consists of an *identifier-list* which is a list of identifiers and a *aggregation* of individual signatures. The signing procedure takes three inputs: a valid message, a valid chain-signature and an identifier. It either outputs a new valid chain-signature or an error. The verification procedure takes two inputs: a message and a chain-signature and outputs true or false. Let $i \geq 1$, $j \geq 0$ be integers. Define $L_j \in \langle \mathbb{I} \rangle$ and $U_j \in \mathbb{G}_1$ as follows:

1) $L_0 = \langle \rangle$, the empty sequence and $U_0 = 0 \in \mathbb{G}_1$
2) $L_i = \langle L_{i-1}, I_i \rangle = \langle I_1, I_2, \ldots, I_i \rangle$
3) $U_i = U_{i-1} + x_i \mathcal{H}(M, L_i) = \sum_{r=1}^{r=i} x_r \mathcal{H}(M, L_r)$

1) Signing: For any participant I_i, this procedure takes as input the message m, the identifier I_i, the chain-signature $\langle L_{i-1}, U_{i-1} \rangle$ and outputs a new chain-signature $\langle L_i, U_i \rangle$ where the values L_{i-1}, L_i, U_{i-1} and U_i are as defined above.

2) Verification: For clarity, we describe the verification procedure to be followed by I_{i+1}. This procedure takes as input the message m, the signature $\langle L_i, U_i \rangle$ and outputs true or false. The process can be described as follows:

1) Output false if the sequence L_i contains any duplicate elements.
2) Output true if $e(U_i, P) = \prod_{r=1}^{r=i} e(\mathcal{H}(M, L_r), Y_r)$ otherwise output false.

If the output of the verification process is true, it can be ascertained that the order proclaimed in the list of signers is indeed correct. Notice that our signatures are very similar to the aggregate signatures of Boneh et al. [15] where signatures of many users (on different messages) are verified in one single step. In our scheme, however, an aggregation of signatures of many users on the same message is verified at once and the exact order of the signers is preserved.

D. Analysis Of The Protocol

We will now give a brief analysis of the protocol and show that it achieves the necessary objectives of correctness and soundness. Roughly speaking, correctness requires that if all participants behave correctly, then the verification process should always output true. On the other hand, soundness requires that the verification process should output false with overwhelming probability if even one participant misbehaves. Recall that we want to ensure that the ordered list of participants specified in each L_i should correctly and uniquely identify the path of the received message. The objectives of the protocol are summarized below:

1) Using their private keys, participants can add their names to the end of the list (contained in the signature) without interaction. Arbitrary names cannot be added to a list without access to the corresponding private keys.
2) Once added, a name cannot be removed from the list without access to the corresponding private key. However, the authenticity of the list can be verified (without interaction) using only the public keys of the participants involved.
3) The authenticity of the list can be verified if and only if the correct order of all the participants is supplied.
4) Non-repudiation must be provided. A holder of a chained signature should be able to convince a judge about the identities of all participants in the chain (in the correct order). The above properties automatically imply non-repudiation.

1) Correctness: We must show that if all the participants behave correctly, then the verification process will always succeed. The correctness of the verification process (of section III-C.2) follows directly from the property of bilinear maps: LHS of step 2 of the verification process (section III-C.2) $= e(U_i, P) = e(\sum_{r=1}^{r=i} x_r\mathcal{H}(M, L_r), P) = \prod_{r=1}^{r=i} e(\mathcal{H}(M, L_r), x_rP)$ = RHS

2) Soundness: To prove soundness of our scheme we need to show the verification process fails with a high probability if the protocol is not followed correctly (that is, if even one user misbehaves). The standard accepted notion of soundness is *chosen ciphertext* security which assumes the most powerful type of adversary with access to all communication channels [16], [17]. The adversary can corrupt any party and is capable of removing arbitrary messages from any communication channel and injecting new messages into the channel. A scheme is secure if no such adversary is able to defeat the protocol's objectives. Usually, to prove the soundness of a signature scheme, it is enough to show that existential forgery on any message is not possible [18]. To achieve chosen ciphertext security for chained signatures, however, two types of forgeries must be considered independently of each other:

- Forgery of signatures on any previously unsigned messages
- Forgery of signatures on any previously unsigned sequence of identities

In other words, we must show that the signatures are unforgeable with respect to any chosen message M or any chosen sequence L_i (which uniquely and correctly identifies the order of signers). This follows from the following theorem.

Theorem 1: *Assume that the DHP in $\mathbb{G}_1$ is hard. Then our scheme is secure against existential forgery on messages or sequence of identities.*

Proof: We assume that the hash function $\mathcal{H}$ is a random oracle. Refer to the definitions in section III-C. U_i can be considered as an aggregation (or sum) of individual signatures of I_r on $\langle M, L_r\rangle\ \forall r \leq i$ using the aggregate signature scheme of [15]. To see this let $S_r \in \mathbb{G}_1$ be the individual signature of I_r on $\langle M, L_r\rangle$ where $S_r = x_r\mathcal{H}(M, L_r)$. We can then rewrite:

$$U_i = \sum_{r=1}^{r=i} x_r\mathcal{H}(M, L_r) = \sum_{r=1}^{r=i} x_r\mathcal{H}(M, L_r) = \sum_{r=1}^{r=i} S_r$$

It is shown in theorem 3.2 of [10] that our scheme is secure against existential forgery of *individual* signatures if $\mathcal{H}$ is a random oracle. Assuming that $I_r \neq I_j$ whenever $r \neq j$ and $h(I_j) \neq \epsilon\ \forall j$, it is also ensured that $L_r \neq L_j$ whenever $r \neq j$ (in other words, all L_r are distinct). It is shown in theorem 3.2 of [15] that this scheme is secure against existential forgery of *aggregate* signatures if the messages $\langle M, L_r\rangle$ are all distinct.

Consider any aggregation U_i of individual signatures $\{S_1, S_2 \dots S_i\}$ such that none of the individual signatures are known. It is shown in theorem 4.5 of [15] that extracting any individual signature (or any sub-aggregation of individual signatures) from U_i is not feasible if the DHP in $\mathbb{G}_1$ is hard.

To prove that our chained signature scheme is secure, it remains to be shown that an adversary cannot even change the order of the identies used in the verification. First note that existential forgery on individual signatures is not possible. If $\mathcal{H}$ is a random oracle, the probability of finding collisions of the type $\mathcal{H}(M, L_i) = \mathcal{H}(M', L'_i)$ where $\langle M, L_i\rangle \neq \langle M', L'_i\rangle$ is negligible. This ensures that the pairs $\langle U_i, M\rangle$ and $\langle U_i, L_i\rangle$ are both fixed from the adversary's point of view. In other words, each individual signature S_i corresponds to the unique sequence $L_i \in \langle\mathbb{I}\rangle$ and a unique message $M \in \Sigma^*$. This completes the proof of security of our scheme. To summarize, this scheme is secure against the following attacks:

1) Existential forgery of individual and aggregate signatures
2) Existential forgery of the order of signers in the signature
3) Extraction of individual signatures from an aggregation

IV. Overview Of The Protocol

The above protocol is an example of a one-way signature chaining scheme. To understand this, see that step 2 of the verification process (section III-C.2) involves the public keys of all participating users (in the right order). We see that the signatures have an "additive" property, demonstrated by the fact that I_{i+1} can 'add' more information to the signature U_i of I_i by computing U_{i+1}. Non-repudiation is provided as follows: I_{i+1} can prove in a court that the message was indeed received from I_i by producing this signature as a *witness* and invoking the verification procedure. A few points about this protocol are noteworthy:

1) Any user I_i who passes the message can add its name in the list of the signature. Once added, users cannot remove names of other users from the list (without completely making the list empty), nor can they change the order or add new names.
2) The signing and verification procedures are completely non-interactive. Moreover, it is possible to combine the signing and verifying procedures into a single *sign-verify* procedure to increase efficiency.
3) The signature size is constant ignoring the payload of the identifier list (which cannot be avoided). The performance of the scheme can be summarized as follows (assuming n users in the chain):
    a) Signing: one multiplication in $\mathbb{G}_1$, one addition in $\mathbb{G}_1$ and one computation of $\mathcal{H}$
    b) Verification: n pairing computations and multiplications in $\mathbb{G}_2$, and n computations of $\mathcal{H}$

Our protocol demonstrates a type of chaining called *backward* chaining where each receiver of the message is responsible for "adding" a link to the chain. Likewise, we can also consider *forward* chaining where the senders of the message are responsible for creating the chain. In this variant, each sender is aware of the next receiver during the signing process. Forward chaining has the advantage that the order of participants can be strictly specified by senders. However, such a scheme also restricts the flexibility of the system because the message will have to be signed multiple times if sent to many receivers in parallel. Moreover in a backward chaining scheme, multiple senders within a 'trust zone' can use a single signing gateway without revealing the identity of the recipients. Due to these reasons, we only considered backward chaining in our work.[2] It is easy to convert our backward chaining scheme to a forward chaining one simply by redefining L_j in section III-C as follows: $L_0 = \langle I_1 \rangle$ and $L_k = \langle L_{k-1}, I_{k+1} \rangle$ if $k > 0$. We note that forward chaining schemes can be trivially made without pairings as described in [19]. The resulting signatures, however, are inefficient in size. We are not aware of any simple constructions for backward chaining schemes without pairings. In this regard, our scheme is unique because it enables ad-hoc backward chaining without involving any third parties.

V. Conclusion and Future Directions

In this paper, we proposed a one-way signature chaining scheme based on bilinear pairings over elliptic curves. Our method is based on the notion of additive zero knowledge [1] which enables trust to propagate between different provers. We demonstrated that signature chaining can be used to form ad-hoc trust relationships between multiple participants in a dynamic and non-interactive manner. Our protocol uses a standard certificate-based PKI and it is worth researching if a certificate-less or an identity based scheme can be derived from the certificate based one presented in this paper.

Considering that one-way signature chaining enables us to correctly validate the path of any received message using very short signatures and provides non-repudiation, we can consider several applications: mobile agent authentication [19], [1], electronic auctions, payment systems [20], group e-commerce (e-commerce transactions where multiple entities are involved such that direct interaction is not possible between many of them), electronic work-flow enforcement (ensuring the order in which participants should be involved), 'secret-passing' protocols, secure routing, authenticated mail relaying and spam tracing, token based authentication, IP tracing, mobile IP, intrusion detection, GRID computing, battlefield modeling, Supply Chain Management, distributed systems and wireless roaming.

[2]Note that forward chaining is equivalent to 'designated' proxy chain signatures while backward chaining is equivalent to universal or undesignated proxy chain signatures.

References

[1] Amitabh Saxena and Ben Soh. Authenticating mobile agent platforms using signature chaining without trusted third parties. In *Proceedings of The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05)*, pages 282–285, Hong kong, 2005. IEEE computer press.

[2] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security*, pages 48–57, New York, NY, USA, 1996. ACM Press.

[3] Amitabh Saxena and Ben Soh. A novel method for authenticating mobile agents with one-way signature chaining. In *Proceedings of The 7th International Symposium on Autonomous Decentralized Systems (ISADS 05)*, pages 187–193, China, 2005. IEEE Computer Press.

[4] Dan Boneh and Matthew 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. Springer-Verlag, 2001.

[5] Kenneth G. Paterson. Id-based signatures from pairings on elliptic curves. Cryptology ePrint Archive, Report 2002/004, 2002.

[6] Song Han, Winson K.Y. Yeung, and Jie Wang. Identity-based confirmer signatures from pairings over elliptic curves. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 262–263, New York, NY, USA, 2003. ACM Press.

[7] Amit K Awasthi and Sunder Lal. Id-based ring signature and proxy ring signature schemes from bilinear pairings. Cryptology ePrint Archive, Report 2004/184, 2004.

[8] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[9] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003.

[10] Dan Boneh, Ben Lynn, and Hovav 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.

[11] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[12] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 104–118, London, UK, 2001. Springer-Verlag.

[13] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Biham [21], pages 416–432.

[14] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[15] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Biham [21], pages 416–432.

[16] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO ’91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 433–444, London, UK, 1992. Springer-Verlag.

[17] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC ’90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, New York, NY, USA, 1990. ACM Press.

[18] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[19] Amitabh Saxena and Ben Soh. A mobile agent authentication protocol using signature chaining with bilinear pairings. Cryptology ePrint Archive, Report 2005/272, 2005.

[20] Amitabh Saxena, Ben Soh, and Dimitri Zantidis. A digital cash protocol based on additive zero knowledge. In *Proceedings of The 3rd International Workshop on Internet Communications Security (ICCSA 05)*, volume 3482 of *Lecture Notes in Computer Science*, pages 672–680, Singapore, 2005. Springer-Verlag.

[21] Eli Biham, editor. *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*. Springer, 2003.