Pairing Based Cryptography For Distributed And Grid Computing

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

***Abstract*— Over recent years, bilinear pairings have emerged as a major research topic in cryptography. In this paper, we describe some applications of bilinear pairings in cryptography for distributed and Grid computing scenarios by presenting mechanisms for trust delegation and confidentiality on a distributed network like a Grid infrastructure.**

I. Introduction

The Weil and Tate Pairings on elliptic curves have been recently used to construct many new and elegant cryptographic schemes that are thought to be impossible to efficiently construct using non-pairing based techniques. A notable example is the Identity Based Encryption (IBE) scheme of Boneh and Franklin [1]. However, pairing based cryptography offers many other elegant solutions to the problem of Trust and Security in distributed networks. In this paper, we describe two mechanisms for security in distributed computing using pairing based cryptography. Our first method deals with the problem of trust delegation in distributed computing while our second method deals with achieving confidentiality over a large and distributed network.

II. Security In Distributed And Grid Computing

It is thought that future computing environments will be highly distributed and collaborative in nature. A typical example is the emergence of Grid technologies in the last decade.

The paradigm of grid computing offers a model for solving massive computational problems by making use of the unused resources (CPU cycles and/or disk storage) of large numbers of disparate, often desktop, computers treated as a virtual cluster embedded in a distributed telecommunications infrastructure.

The Global Grid Forum (GGF) has the purpose of defining specifications for grid computing. The Globus Alliance[1] implements these standards through the GLOBUS Toolkit, which has become the de facto standard for grid middleware. The security services are provides by a component called Grid Security Infrastructure (GSI). We describe GSI using the following terminology [2]:

- *Virtual Organization* (VO): A VO is a dynamic collection of users and resources that potentially span multiple administrative domains and therefore subject to varying security policies.
- *User*: A user is a subscriber to a Grid. One user can belong to multiple VOs and share his resources

[1]http://www.globus.org/

- *Resource*: Any sharable resource including hardware and software.
- *Community Authentication Service* (CAS): Each VO has its own CAS to maintain a set of policies and communicate those policies to the resources.
- *Community Policy*: A local database that stores policies imposed on each user of a VO.
- *Certification Authority* (CA): An independent trusted CA is responsible for certifying public keys of VO members. A CA can be shared by many VOs.

In a typical scenario, to send a job request to a resource Bob, the user Alice first authenticates herself to the CAS server. The CAS server issues Alice with a signed policy assertion (PA) declaring her identity and rights after authenticating her. Alice sends this PA to resource Bob along with her job request. On receiving this request, Bob validates Alice's PA (i.e. that Alice is indeed allowed to use resource Bob) then processes her job request. To complete this job, Bob may need to access other grid resources such as distributed databases or file servers, possibly in a different VO. In this case, each resource that Bob requests would authenticate and authorize access based on its local access control policy.

As in any networked computer system, it is necessary to authenticate incoming requests and authorize access to Grid resources based on a local security policy. We will assume that traditional mechanisms for authentication and access control can be used. In addition, some other security constraints for a distributed infrastructure are:

1) *Single Sign-On*: In the above example, Alice should be required to authenticate herself only once, when she initiates the session. Additional resource requests from Bob (and further resources down the chain) should be handled without Alice's involvement (and possibly without her explicit knowledge of the resources involved). This is achieved by generating temporary credentials for Alice in the form of a proxy public-private key-pair [3]. The proxy public key is certified by Alice with her long-term private key during her initial request phase. Further delegation on Alice's behalf is done via the proxy private key and is verified via the proxy public key.
2) *Delegation Of Trust*: A requirement similar to the above is that Alice (via her proxy private key) should be able to temporarily "delegate her trust" to Bob for the duration of her job session. Bob should additionally be able to further 'chain' his delegation to Alice's and be able

1-4244-0355-3/06/$20.00 (c) 2006 IEEE

to forward his request to the next needed resource in the chain. This is necessary in order to minimize the messages transferred between intermediate resources. In section IV we present an efficient method for delegation accumulation and verification using the idea of chain signatures.
3) *Support For Secure Group Communication*: As noted in [3], an efficient and scalable infrastructure for secure group communication forms an essential part of any distributed computing environment. In section V, we describe an efficient key agreement protocol for dynamic ad-hoc groups that can be adapted to any distributed environment. This protocol can be used to provide the underlying secure group communication infrastructure. Additionally, it can also be used in a Message Authentication Code (MAC) to achieve integrity.

In the next section, we introduce the cryptographic primitives for the protocols described in this paper.

III. Pairing Based Cryptography

Pairing based cryptography is based on the existence of efficiently computable non-degenerate bilinear maps which can be abstractly described as follows. Let G_1 and G_2 be two cyclic multiplicative groups both of the same order n such that computing discrete logarithms in G_1 and G_2 is intractable. A bilinear pairing is a map $\hat{e} : G_1 \times G_1 \mapsto G_2$ that satisfies the following properties:

1) *Bilinearity*: $\hat{e}(a^x, b^y) = \hat{e}(a,b)^{xy}\ \forall a,b \in G_1$ and $x,y \in \mathbb{Z}_n$.
2) *Non-degeneracy*: If g is a generator of G_1 then $\hat{e}(g,g)$ is a generator of G_2.
3) *Computability*: The map $\hat{e}$ is efficiently computable.

Typically the map $\hat{e}$ will be derived from either the modified Weil or the Tate Pairing over elliptic curves and a finite field [4]. See, for example [5] for details on generating bilinear maps of any given order n that is square free. Depending on the type of application, the value of n will either be a prime or composite. Whenever, n is composite, we will assume that $n = pq$ where p, q are distinct odd primes such that given the product $n = pq$, factoring n is intractable. Let g be some **fixed** generator of G_1. We define the following problems.

A. *Diffie-Hellman Problem (DHP$_{(g,G_1)}$)*: Given $g^x, g^y \in G_1$ output $g^{xy} \in G_1$.
B. *Decision Diffie-Hellman Problem (DDHP$_{(g,G_1)}$)*: Given $g^x, g^y, g^z \in G_1$ output 1 if $z = xy \in \mathbb{Z}_n$; otherwise output 0.
C. *Inverse Diffie-Hellman Problem (IDHP$_{(g,G_1)}$)*: Given $g^x \in G_1$ for some $x \in \mathbb{Z}_n^*$, output $g^{1/x} \in G_1$.

Lemma III.1. *DDHP$_{(g,G_1)}$ (the Decision Diffie-Hellman Problem) is easy*

Proof. Clearly, from the properties of the mapping, $z = xy \in \mathbb{Z}_n$ if and only if $\hat{e}(g, g^z) = \hat{e}(g^x, g^y)$. Thus, solving DDHP$_{(g,G_1)}$ is equivalent to computing the mapping $\hat{e}$ twice. □

Whenever n is composite, we will also make the following hypothesis.

Conjuncture III.2. *IDHP$_{(g,G_1)}$ $\not\Rightarrow$ DHP$_{(g,G_1)}$ unless $\phi(n)$ is known.*

IV. Chain Signatures

Chain signatures are derived from the aggregate signatures of Boneh et al. [6] and were first mentioned in [7] for e-commerce applications. In this section, we demonstrate some interesting properties of chain signatures that make them suitable for trust delegation in Grid computing. Our approach is similar to the idea of "role based cascaded delegation" using Hierarchal Certificate Based Encryption (HCBE) [8].

Consider a grid computing scenario with j (distinct) ordered users/resources $\{I_1, I_2 \dots I_j\}$ that participate for the completion of some job, which is represented using the contract m. The transfer of delegation is carried out from I_1 to I_k sequentially via the intermediate nodes. Consider the incremental delegations below (here, a delegation implies a short "proof"):

$$\begin{array}{ll} D_0: & I_1 \\ D_1: & I_1 \rightarrow I_2 \\ D_2: & I_1 \rightarrow I_2 \rightarrow I_3 \\ D_3: & I_1 \rightarrow I_2 \rightarrow I_3 \rightarrow I_4 \\ & \text{etc} \dots \end{array}$$

For instance, D_3 can be considered as I_4's proof of delegation (of some rights) from I_1 via intermediate nodes I_2 and I_3. We want to enforce the following conditions for this delegation transfer.

1) It should be possible to represent an arbitrary long chain of delegations using only a short cryptographic credential. Preferably each value D_i should be of constant size.
2) For any $i > 0$, it should be possible to compute D_i directly from D_{i-1} without interaction with I_i.
That is, we assume that participant I_i may not be available for interaction once a resource request to I_{i+1} has been submitted.
3) It should be possible to "accumulate" delegation credentials in the following manner. Given just the delegation D_i, it should not be possible for any member I_l where $l \notin \{1, 2, \dots i\}$ to extract any sub-delegation D_k for some $k < i$.
4) Given a delegation D_i, it should not be possible for any member I_l (where possibly $l \in \{1, 2, \dots i\}$) to change the order of hosts mentioned in D_i.

In a scheme such as above, any delegation D_i non-repudiably authenticates all intermediate nodes in the chain. The aggregate signature scheme of [6] will be the basis of our delegation scheme.

A. Setup PKI

This construction makes use of a bilinear map $\hat{e} : G_1 \times G_1 \mapsto G_2$ as described in section III between cyclic multiplicative groups G_1, G_2 of prime order n. To participate in the protocol each user must have a certified public key. The setup proceeds as follows:

1) Pick a random generator g of G_1 and let $\mathcal{H} : 2^{\Sigma^*} \mapsto G_1$ be a cryptographic hash function constructed using the technique described in [4]. The parameters $(\hat{e}, G_1, G_2, n, g, \mathcal{H})$ are generated by a trusted authority and made public in an authentic way.
2) Each participant I_i generates $x_i \overset{R}{\leftarrow} \mathbb{Z}_n^*$ as the private key. The corresponding public key is $y_i = g^{x_i} \in G_1$.

B. Trust Delegation Using Chain Signatures

In this scenario there are j ordered distinct participants $\langle I_1, I_2 \dots I_j \rangle$. The original user that initiated the request is I_1. The request m is passed from I_i to I_{i+1} along with a chain-signature (delegation credential) as described below.

1) Signing: Let $L_r = \langle I_1, I_2, \dots I_r \rangle$ for $r \geq 1$. Define $D_0 = 1 \in \mathbb{G}_1$. Define recursively

$$D_i = \mathcal{H}(m, L_i)^{x_i} \cdot D_{i-1}$$

$$= \prod_{r=1}^{i} \mathcal{H}(m, L_i)_i^x$$

The chain signature (delegation credential) of I_i on the message m is (D_i, L_i)

2) Verification: I_{i+1} accepts the above signature of I_i as valid if the following equality holds

$$\hat{e}(D_i, g) = \prod_{r=1}^{i} \hat{e}(\mathcal{H}(m, L_r), y_r)$$

Clearly, the size of delegation D_i remains constant irrespective of the value of i.

C. Security

The protocol is secure against aggregate delegation forgery if the Diffie-Hellman problem in G_1 is hard [6] assuming that $\mathcal{H}$ is a random oracle. Additionally, given any delegation D_i, extracting any sub-delegation is also provably equivalent to the Diffie-Hellman Problem [9]. Finally, changing the order of identities contained in L_i is hard if $\mathcal{H}$ is assumed to be collision resistant.

V. Distributed Key Agreement

The problem of achieving confidentiality over distributed and broadcast networks is a challenging problem. In most scenarios, a message must be encrypted to a selected subset of users. To achieve scalability in this model, it must be possible to dynamically share secret keys without interaction with other group members. A new cryptosystem based on composite order bilinear maps for key agreement in large distributed environments is proposed in [10] that allows this using a distributed third party. Here we give a brief overview of their scheme which is based on Rabi and Sherman's protocols for one-round group key agreement using *Strong Associative One Way Functions* (SAOWFs) [11].

A. Strong Associative One Way Functions

Let $\mathbb{G}$ be a finite abelian group with respect to the operation $\star$. The mapping $f : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}$ defined by $f(A, B) = A \star B$ for $A, B \in \mathbb{G}$ has the following properties:

1) $f(f(A, B), C) = f(A, f(B, C)) \forall A, B, C \in \mathbb{G}$ [Associativity]
2) $f(A, B) = f(B, A) \forall A, B \in \mathbb{G}$ [Commutativity]
3) There exists a unique identity element $I \in \mathbb{G}$ such that $f(A, I) = A \; \forall A \in \mathbb{G}$.
4) For each $A \in \mathbb{G}$, there exists a unique element $B \in \mathbb{G}$ such that $f(A, B) = I$. We say B is the inverse of A and denote it by A^{-1} when there is no ambiguity in the notation.

The above properties are ensured in any abelian group. We now additionally want to enforce the following properties:

5. Computability: For all $A, B \in \mathbb{G}$, $f(A, B)$ must be efficiently computable.
6. Strong Non-Invertibility: Let $A, B \overset{R}{\leftarrow} \mathbb{G}$ and $C = f(A, B) \in \mathbb{G}$. Given A, C, computing $B = f(C, A^{-1})$ must be infeasible.

We say that f is a *Strong Associative One-Way Function* (SAOWF) if all of the above six properties are satisfied. As shown in [11], one-round key agreement is possible using SAOWFs.

Although a practical construction of an SAOWF is not known, Saxena et al. [10] propose a novel alternative to this by presenting a practical implementation of a "black-box" SAOWF using a technique called *Verifiable Oracles* which is informally described as follows.

B. Oracle Based Black-box SAOWF

A central authority (the oracle) is responsible for setting up the parameters of $\mathbb{G}$. In their implementation, only the oracle has the ability to compute f. Thus, we give all instances of pairs (A, B) to the oracle to compute $f(A, B)$. However, when at least one of $\{A, B\}$ has been sampled by us, we can compute f directly without help from the oracle. Additionally, we can blind any pairs (A, B) by creating new pairs (A', B') and ask the oracle to compute $f(A', B')$. From the oracle's output, $f(A, B)$ can be directly computed such that even the oracle does not know the value of $f(A, B)$. Finally, the output of the oracle can be verified.

The idea of [10] is to use a bilinear group G_1 of composite order n (such that n is hard to factor), use the oracle as a "Diffie-Hellman problem" solver in G_1 and exploit the gap between the Diffie-Hellman Problem and the Inverse Diffie-Hellman Problem whenever $\phi(n)$ is unknown (conjuncture III.2, section III).

Since the only known way to solve the Diffie-Hellman problem is to compute discrete logarithms, we provide the discrete logarithm to the oracle encrypted using with an asymmetric encryption algorithm $\mathbf{E}$ having the following multiplicative homomorphic property; for any messages $m_1, m_2 \in \mathbb{Z}_n$, given $\{\mathbf{E}(m_1), m_2\}$ or $\{m_1, \mathbf{E}(m_2)\}$, it must be possible to compute $\mathbf{E}(m_1 m_2 \bmod n)$ directly without knowing the corresponding decryption algorithm $\mathbf{D}$. The Paillier cryptosystem [12] has this property and we will use it in our construction.

C. Parameter Generation

Without going into the details of Paillier encryption method [12] or the black-box SAOWF construction of [10], we briefly describe the group key agreement protocol. The reader is referred to the appropriate papers for a detailed discussion.

Let G_1, G_2 be cyclic multiplicative bilinear groups of composite order n as discussed in section III, such that G_1 is generated by g. Let λ be the private key of the Paillier cryptosystem [12] corresponding to the public key (t, n).

In this model, elements of $\mathbb{G}$ are of the form (u, v) where $u = g^x \in G_1$ and $v = \mathbf{E}(x)$ using the Paillier public key (t, n) for some $x \in \mathbb{Z}_n^*$. We will assume that unless the factors of n are known, it is infeasible to select x from $\mathbb{Z}_n \backslash \mathbb{Z}_n^*$. Denote the oracle knowing the factorization n by $\mathcal{O}$. We define a binary operation $\star$ in $\mathbb{G}$ as follows:

Let $A = (u_a, v_a)$ and $B = (u_b, v_b)$ such that $A, B \in \mathbb{G}$. Then $C = A \star B = (u_c, v_c) \in \mathbb{G}$ where

$$u_c = u_a^{\mathbf{D}(v_b)} = u_b^{\mathbf{D}(v_a)} = g^{\mathbf{D}(v_a)\mathbf{D}(v_b)} \in G_1 \tag{1}$$

and

$$\mathbf{D}(v_c) = \mathbf{D}(v_a)\mathbf{D}(v_b) \bmod n \tag{2}$$

$$= \mathbf{D}(v_a^{\mathbf{D}(v_b)} \bmod n^2) = \mathbf{D}(v_b^{\mathbf{D}(v_a)} \bmod n^2) \tag{3}$$

where equation 3 follows from the multiplicative properties of the Paillier cryptosystem. We state without proof the following facts [10].

1) Clearly, if either of $\mathbf{D}(v_a)$ or $\mathbf{D}(v_b)$ is known then we can compute $A \star B$ directly.
2) On the other hand if neither of $\mathbf{D}(v_a)$ or $\mathbf{D}(v_b)$ is known, we can use the oracle to compute $A \star B = \mathcal{O}(A, B)$ using equations 1 and 2.
3) If it is known that $A, B \in \mathbb{G}$, it is possibly to verify the first element u_c of $A \star B$ because $u_c = g^{\mathbf{D}(v_a)\mathbf{D}(v_b)}$ if and only if $\hat{e}(g, u_c) = \hat{e}(u_a, u_b)$ (see lemma III.1).
4) $\mathbb{G}$ is efficiently samplable; first we sample random $x \xleftarrow{R} \mathbb{Z}_n^*$ then compute $u = g^x$ and $v = \mathbf{E}(x)$. Thus, $(u, v) \in \mathbb{G}$. In this case we call $x = \mathbf{D}(v)$ the *sampling information* for (u, v).
5) Let $A = (u_a, v_a) \in \mathbb{G}$ and let $x_a = \mathbf{D}(v_a)$ be the sampling information for A. Anyone who knows x_a can also compute $A^{-1} \star B$ for any $B \in \mathbb{G}$.
6) Partition $\mathbb{G}$ into equivalence classes using the relation $\sim$ as follows. Let $A, B \in \mathbb{G}$ such that $A = (u_a, v_a)$ and $B = (u_b, v_b)$. We say $A \sim B$ if and only if $u_a = u_b$. The relation $\sim$ transforms the equivalence classes of $\mathbb{G}$ into an abelian group with respect to the operation $\star$. Additionally, if we assume that the oracle is used for computing $\star$ (see item 2 above), then this group satisfies the strong non-invertibility and computability conditions mentioned earlier.

For notational convenience in what follows, we will use the symbol $\tilde{A}$ to denote the equivalence class of A for any $A \in \mathbb{G}$. We will also use the "$=$" operator instead of "$\sim$" when working with equivalence classes whenever there is no confusion.

For any $\tilde{A} \in \mathbb{G}$, we let the symbol $\tilde{A}^i$ denote $\tilde{A} \star \tilde{A} \star \ldots \tilde{A}$ (i times). The inverse of $\tilde{A}$ is denoted by $\tilde{A}^{-1}$. It can be trivially verified that the following are also true: $\tilde{A}^i \star \tilde{A}^j = \tilde{A}^{i+j}$; $(\tilde{A}^i)^j = \tilde{A}^{ij}$; $\tilde{A} \star \tilde{A}^{-1} = \tilde{A}^0 = \tilde{I}$ (where $\tilde{I}$ is the identity equivalence class such that $(g, \mathbf{E}(1)) \in \tilde{I}$) and; $(\tilde{A}^i \star \tilde{B}^j)^k = \tilde{A}^{ij} \star \tilde{B}^{jk}$, for all $\tilde{A}, \tilde{B} \subsetneq \mathbb{G}$ and all i, j, k in $\mathbb{Z}$.

D. Group Key Agreement

We will use the protocol of Rabi and Sherman [11] for group key agreement.

1) Setup PKI: Before the system can be used, a public key infrastructure must be set up as follows.

1) The oracle $\mathcal{O}$ generates a random element $\tilde{P} \xleftarrow{R} \mathbb{G}$, which will serve as a common public value in our key agreement protocol. We will assume that the sampling information of $\tilde{P}$ is kept secret by the oracle.
2) Each user i generates a private key $\tilde{X}_i \xleftarrow{R} \mathbb{G}$ using the sampling method described above. The sampling information is also kept as part of the private key.
3) Each user i computes the public key $\tilde{Y}_i = \tilde{X}_i \star \tilde{P}$. This computation is possible because each private key $\tilde{X}_i$ has been sampled by user i. The public keys are made available in an authentic way.

2) Key Agreement Protocol: Without loss of generality, we demonstrate how four users can compute a shared key. The four users (1, 2, 3, 4) can use the oracle to compute a secret key as follows:

1) User 1 uses the oracle $\mathcal{O}$ to compute the partial public key $\tilde{Y}_{234} = \tilde{Y}_2 \star \tilde{Y}_3 \star \tilde{Y}_4 = \tilde{X}_2 \star \tilde{X}_3 \star \tilde{X}_4 \star \tilde{P}^3$ by making 2 calls; i.e. by computing $\tilde{Y}_{234} = \mathcal{O}(\mathcal{O}(\tilde{Y}_2, \tilde{Y}_3), \tilde{Y}_4)$. We note that everyone is allowed to compute this partial public key $\tilde{Y}_{234}$.
2) The group private key $\tilde{K}_{1234} = \tilde{X}_1 \star \tilde{Y}_{234} = \tilde{X}_1 \star \tilde{X}_2 \star \tilde{X}_3 \star \tilde{X}_4 \star \tilde{P}^3$ can then be directly computed by user 1 without the help of the oracle using the secret sampling information for generating $\tilde{X}_1$. User 2 computes $\tilde{K}_{1234}$ independently of user 1 as $\tilde{K}_{1234} = \tilde{X}_2 \star \tilde{Y}_{134} = \tilde{X}_2 \star \mathcal{O}(\mathcal{O}(\tilde{Y}_1, \tilde{Y}_3), \tilde{Y}_4)$.
3) Likewise, the remaining two users can compute the same group private key $\tilde{K}_{1234}$ using the oracle and their secret sampling information. No other user except the oracle $\mathcal{O}$ has the ability to compute this key.

In general, for a set of m users $\{1, 2, 3 \ldots m\}$ the group private key is

$$\tilde{K}_{123\ldots m} = (\tilde{P}^{m-1} \star \tilde{X}_1 \star \tilde{X}_2 \star \ldots \tilde{X}_m) \tag{4}$$

E. Overview Of The Scheme

In this section we discuss some interesting features of the above scheme.

1) *Secret Key Computation*: Although the secret key in equation 4 is described in terms of equivalence classes, in reality, we only need to know any one element in the equivalence class to agree on the secret key. To see this, let $(u_k, v_k) \in \tilde{K}_{123\ldots m}$. Then the secret key for this group is derived from u_k and the value v_k is ignored.
2) *Communication Complexity*: For a group of m users, a total of $m - 2$ oracle calls are required for each user to

compute the shared key. Thus a total of $m(m-2)$ calls are required for all the m users. However, no specific ordering is required between the users. A user i may choose to compute the shared key *after* a ciphertext is received. Additionally, oracle calls can be sent in a batch.

3) *Universal Key Escrow*: The oracle has universal escrow capability. Given a public key $\tilde{Y}_i = \tilde{X}_i \star \tilde{P}$ for some private key $\tilde{X}_i$, the oracle can invert $\star$ and compute $\tilde{X}_i$.
4) *Non-interactivity*: Assuming that all the public keys $\tilde{Y}_i$ are known in advance, any user can compute the shared key without interacting with the other users.
5) *Decentralizing the Oracle*: An arbitrary number of "copies" of the oracle can be run without any compromise in security. The ability to do more computations of $\star$ does not give any additional advantage.
6) *Protection From Passive Adversaries*: It is possible to use the oracle $\mathcal{O}$ to compute $\tilde{A} \star \tilde{B}$ for any $\tilde{A}, \tilde{B} \in \mathbb{G}$ such that the oracle does not know either of $\tilde{A}$ or $\tilde{B}$ using the following blinding technique [10].
   a) The input is $\tilde{A}, \tilde{B} \in \mathbb{G}$ such that none of $\{\tilde{A}, \tilde{B}, \tilde{A}^{-1}, \tilde{B}^{-1}\}$ have been sampled by us.
   b) Sample $\tilde{A}_1, \tilde{B}_1 \xleftarrow{R} \mathbb{G}$. Thus we can compute $\tilde{A}' = \tilde{A} \star \tilde{A}_1$ and $\tilde{B}' = \tilde{B} \star \tilde{B}_1$.
   c) Use the oracle to output $\tilde{C}' = \tilde{A}' \star \tilde{B}'$.
   d) Compute $\tilde{C} = \tilde{A}_1^{-1} \star \tilde{B}_1^{-1} \star \tilde{C}'$ and output $\tilde{C} = \tilde{A} \star \tilde{B}$.
7) *Protection From Active Adversaries*: For any A, B, is it possible to verify the first element of $A \star B$ due to the properties of bilinear maps. It is however, also possible to verify the second element of $A \star B$ due to the above blinding technique [10].
8) Other Extensions: It is shown in [10] how to extend the above key agreement protocol to enable efficient join and merge operations and to creating ring signatures.

F. Security

The security proof of the above key agreement protocol assumes that the following problem is intractable.

Group Inversion Problem. *(GIP$_{\mathbb{G}}$). Let $P = (h, \beta) \xleftarrow{R} \mathbb{G}$ be uniformly sampled using secret $\alpha \xleftarrow{R} \mathbb{Z}_n^*$. In other words, let $h = g^\alpha \in G_1$ and $\beta = \mathbf{E}(\alpha) \in \mathbb{Z}_{n^2}^*$. Given P, compute $P^{-1} = (h', \beta') \in \mathbb{G}$ where $h' = g^{1/\alpha} \in G_1$ and $\beta' = \mathbf{E}(1/\alpha) \in \mathbb{Z}_{n^2}^*$, possibly by using the oracle $\mathcal{O}$.*

It appears that GIP$_{\mathbb{G}}$ is hard if the following three problems are intractable:

1) Breaking the Paillier cryptosystem with public key (t, n).
2) Computing discrete logarithms in G_1 to base g.
3) Reducing IDHP$_{(g,G_1)}$ to DHP$_{(g,G_1)}$ without knowing $\phi(n)$.

VI. Conclusion

In this paper we described some applications of Pairing Based Cryptography in distributed and Grid computing by presenting methods for trust delegation and key agreement. The method of credential accumulation described in section IV allows intermediate nodes to accumulate delegation credentials and combine them into a single constant size delegation credential that authenticates the entire chain such that it is infeasible to extract any sub-delegations just from intermediate delegation. The protocol of section V-D provides a new method of key agreement in large and distributed groups. Although the method eliminates the interaction between individual members, it still requires the use of a distributed trusted third party (with universal key escrow capability) for key computation. The advantage here is that no state information about any groups or individual members needs to be maintained by the third party, which just acts as a "verifiable computing device". The membership of a group can be decided by senders.

References

[1] 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.

[2] H. W. Lim and Matthew J. B. Robshaw. On identity-based cryptography and grid computing. In Marian Bubak, G. Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, *International Conference on Computational Science*, volume 3036 of *Lecture Notes in Computer Science*, pages 474–477. Springer, 2004.

[3] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[4] 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.

[5] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[6] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[7] Amitabh Saxena and Ben Soh. One-way signature chaining: A new paradigm for group cryptosystems and e-commerce. Cryptology ePrint Archive, Report 2005/335, 2005.

[8] Roberto Tamassia, Danfeng Yao, and William H. Winsborough. Role-based cascaded delegation. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 146–155, New York, NY, USA, 2004. ACM Press.

[9] Jean-Sébastien Coron and David Naccache. Boneh et al.'s k-element aggregate extraction assumption is equivalent to the diffie-hellman assumption. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 392–397. Springer, 2003.

[10] Amitabh Saxena and Ben Soh. A new cryptosystem based on hidden order groups. 2006. Unpublished manuscript. Available from http://homepage.cs.latrobe.edu.au/asaxena/saxena06key.pdf.

[11] Muhammad Rabi and Alan T. Sherman. An observation on associative one-way functions in complexity theory. *Inf. Process. Lett.*, 64(5):239–244, 1997.

[12] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.