ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

Authentication, Key Establishment & Cooperative Cache maintenance in Wireless P2P Environments Using GDC & Greedy Algorithm

[1]N.Sandeep Chaitanya
Research scholar
JNTUH

[2]P Sruthi
Assoc.Prof
CMRCET

[3]A Nandini
HITAM

[4] T.V Suresh Kumar
Research Scholar
JNTUH

[5]SP. Santhosh
HMITS

Abstract*— Some recent studies have shown that cooperative cache can improve the system performance in wireless P2P networks such as ad hoc networks and mesh networks. In this paper, we present our design and implementation of cooperative cache in wireless P2P networks, propose solutions to find the best place to cache the data and also provides the audit facilities by the owner. We propose a novel asymmetric cooperative cache approach, where the data requests are transmitted to the cache layer on every node, but the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data. This solution not only reduces the overhead of copying data between the user space and the kernel space, it also allows data pipelines to reduce the end-to-end delay. P2P has the potential for significant cost reduction and the increased operating efficiencies in computing. Although security issues are delaying its fast adoption, P2P computing is an unstoppable force and we need to provide security mechanisms to ensure its secure adoption. P2P security & auditing of the data in the cooperate cache is a key component in P2P computing.. This is achieved by applying greedy algorithm for placing the cached data and to provide security public-key cryptography, cryptographic protocols methodologies and the necessary infrastructural supporting services in which public-key authentication framework (PKI) is the main component. A desired distinction would be that security services for P2P security should manifest and facilitate the P2P feature of advanced resource sharing. In this article, we focus on P2P data storage security, which has always been an important aspect of quality of service. To ensure the correctness of users' data in the P2P we propose providing security using GDC. Public-key digital certificate has been widely used in public-key infrastructure (PKI) to provide user public key authentication. However, the public-key digital certificate itself cannot be used as a security factor to authenticate user. In this paper, we propose the concept of generalized digital certificate (GDC) that can be used to provide user authentication and key agreement. A GDC contains user's public information, such as the information of user's digital driver's license, the information of a digital birth certificate, etc., and a digital signature of the public information signed by a trusted certificate authority (CA). However, the GDC does not contain any user's public key. Since the user does not have any private and public key pair, key management in using GDC is much simpler than using public-key digital certificate. The digital signature of the GDC is used as a secret token of each user that will never be revealed to any verifier. Instead, the owner proves to the verifier that he has the knowledge of the signature by responding to the verifier's challenge. Based on this concept, we propose discrete logarithm (DL)-based protocol that can achieve user authentication and secret key establishment.

Index Terms—**P2P Computing, Public-Key Digital Certificate, Key Management, TPA, Greedy Algorithm.**

I. INTRODUCTION

P2P Computing has been envisioned as the next generation architecture of IT enterprise, due to its long list of unprecedented advantages in the IT history: on-demand self-service, ubiquitous network access, rapid resource elasticity, usage-based pricing and transference of risk [1]. As a disruptive technology with profound implications, From users' perspective, including both individuals and IT enterprises, storing data and transferring between peer terminals is a flexible on-demand manner brings appealing benefits: relief of the burden for storage management, and avoidance of capital expenditure on hardware, software, and personnel maintenances, etc. While P2P Computing makes these advantages more appealing than ever, it also brings new and challenging security threats towards users' outsourced data. Since Peer terminals are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data. As a result, the correctness of the data in the P2P is being put at risk due to the following reasons. Examples include P2P service providers, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed, or even hiding data loss incidents so as to maintain a reputation [6]–[8]. In short, although outsourcing data into the P2P is economically

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede the successful deployment of the P2P architecture.

Actually, cooperative caching [5], [6] which allows the sharing and coordination of cached data among multiple nodes, has been applied to improve the system performance in wireless P2P networks and has been widely used to improve the Web performance .Although cooperative caching and proxy techniques have been extensively studied in wired networks, little has been done to apply this technique to wireless networks. These protocols can be classified as message-based, directory-based, or router-based. This implementation was entirely in kernel and made extensive modifications in the kernel IP Stack. Although cooperative cache has been implemented by many researchers [6], [9], these implementations are in the Web environment, and all these implementations are at the system level. As a result, none of them deals with the multiple hop routing problems and cannot address the on-demand nature of the ad hoc routing protocols. To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data request. This certainly cannot be satisfied by the existing ad hoc routing protocols. In this paper, we present our design and implementation of cooperative cache in wireless P2P networks. Through real implementations, we identify important design issues and propose an asymmetric approach to reduce the overhead of copying data between the user space and the kernel space, and hence to reduce the data processing delay. Another major contribution of this paper is to identify and address the effects of data pipeline and MAC layer interference on the performance of caching. Although some researchers have addressed the effects of MAC layer interference on the performance of TCP [10] and network capacity, this is the first work to study this problem in the context of cache management. We also propose solutions for our asymmetric approach to identify the best nodes to cache the data. The proposed algorithm well considers the caching overhead and adapts the cache node selection strategy to maximize the caching benefit on different MAC layers.

II BASIC COOPERATIVE CACHE SCHEMES

Fig. 1 illustrates the Cache Path concept. Suppose node N1 requests a data item di from N0. When N3 forwards di to N1;N3 knows that N1 has a copy of the data. Later, if N2 requests di;N3 knows that the data source N0 is three hops away whereas N1 is only one hop away. Thus, N3 forwards the request to N1 instead of N4. Many routing algorithms (such as AODV [20] and DSR [12]) provide the hop count information between the source and destination. Caching the data path for each data item reduces bandwidth and power consumption because nodes can obtain the data using fewer hops. However, mapping data items and caching nodes increase routing overhead, and the following techniques are used to improve Cache Path's performance.

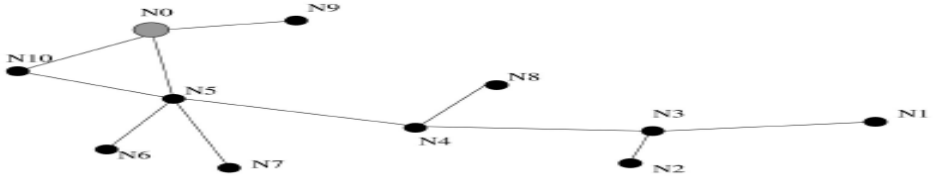

Fig. 1. A Wireless P2P Network

In Cache Path, a node need not record the path information of all passing data. Rather, it only records the data path when it is closer to the caching node than the data source. For example, when N0 forwards di to the destination node N1 along the path N5 _ N4 _ N3;N4 and N5 won't cache di path information because they are closer to the data source than the caching node N1. In general, a node caches the data path only when the caching node is very close. The closeness can be defined as a function of the node's distance to the data source, its distance to the caching node, route stability, and the data update rate. Intuitively, if the network is relatively stable, the data update rate is low, and its distance to the caching node is much shorter than its distance to the data source, then the routing node should cache the data path. In Cache Data, the intermediate node caches the data instead of the path when it finds that the data item is frequently accessed. For example, in Fig. 1, if both N6 and N7 request di through N5;N5 may think that di is popular and cache it locally. N5 can then serve N4's future requests directly. Because the Cache Data approach needs extra space to save the data, it should be used prudently. Suppose N3 forwards several requests for di to N0. The nodes along the path N3; N4, and N5 may want to cache di as a frequently accessed item. However, they will waste a large amount of cache space if they all cache di. To avoid this, Cache Data enforces another rule: A node does not cache the data if all requests for the data are from the same node. In this example, all the requests N5 received are from N4, and these requests in turn come from N3. With the new rule, N4 and N5

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

won't cache di. If N3 receives requests from different nodes, for example, N1 and N2, it caches the data. Certainly, if N5 later receives requests for di from N6 and N7, it can also cache the data. Cache Path and Cache Data can significantly improve system performance. Analytical results [8] show that Cache Path performs better when the cache is small or the data update rate is low, while Cache Data performs better in other situations. To further improve performance, we can use Hybrid Cache, a hybrid scheme that exploits the strengths of Cache Data and Cache Path while avoiding their weaknesses. Specifically, when a node forwards a data item, it caches the data or path based on several criteria.

A. Design Issues on Implementing Cooperative Cache

To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data request. This certainly cannot be satisfied by the existing ad hoc routing protocols. Next, we look at two design options.

B. Integrated Design

In this option, the cooperative cache functions are integrated into the network layer so that the intermediate node can check each passing-by packet to see if the requested data can be served. Although this design sounds straightforward, several major drawbacks make it impossible in real implementation. The network layer is usually implemented in kernel, and hence, the integrated design implies a kernel implementation of cooperative cache. However, it is well known that kernel implementation is difficult to customize and Second, kernel implementation will significantly increase the memory demand due to the use of CacheData. Finally, there is no de facto routing protocol for wireless P2P networks currently. Implementing cooperative cache at the network layer requires these cache and routing modules to be tightly coupled, and the routing module has to be modified to add caching functionality. However, to integrate cooperative cache with different routing protocols will involve tremendous amount of work.

C. Layered Design

In this a dedicated cooperative cache layer resided in the user space; i.e., cooperative cache is designed as a middleware lying right below the application layer and on top of the network layer (including the transport layer). There are two options for the layered design. One naïve solution uses cross-layer information, where the application passes data request (search key) to the routing layer, which can be used to match the local cached data. However, this solution not only violates the layered design, but also adds significant complexity to the routing protocol which now needs to maintain a local cache table. Further, if an intermediate node needs to cache the data based on the cooperative cache protocol, it has to deal with fragmentation issues since some fragments of the data may not go through this node. Thus, this naive solution does not work in practice. Although this solution can solve the problems of the naïve solution, it has significant overhead. For example, to avoid caching corrupted data, reliable protocols such as TCP are needed. However, this will significantly increase the overhead, since the data packets have to move to the TCP layer at each hop. Note that the data packets only need to go to the routing layer if cooperative cache is not used. Further, this solution has a very high context switching overhead. At each intermediate node, the packets have to be copied from the kernel to the user space for cache operations, and then they are injected back to kernel to be routed to the next hop.

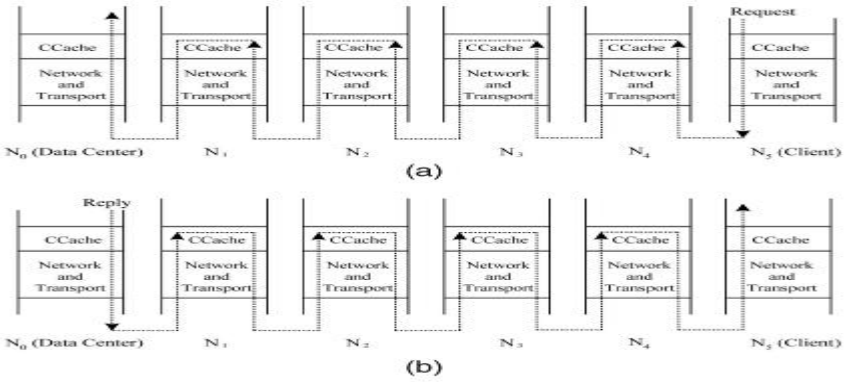

Fig. 2. Layered design. (a) The request packet flow and (b) the reply packet flow.

The **pipeline effect is a**nother problem of the layered design. The lack of data pipeline normally, the transport layer can fragment a large data item into many small data packets, which are sent one by one to the next hop. If there are multihops between the sender and the receiver, these small packets can be pipelined and the end-to-end delay can be reduced. In cooperative cache, the caching granularity is at the data item level. Although a large data item is still

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

fragmented by the transport layer, there is no pipeline due to the layered design. This is because the cache layer is on top of the transport layer, which will reassemble the fragmented packets. Since all packets have to go up to the cache layer hop by hop, the network runs like "stop and wait" instead of "sliding window." This will significantly increase the end to- end delay, especially for data with large size.

The Basic Idea the Asymmetric Cooperative Cache Approach

To address the problem of the layered design, we propose an asymmetric approach. We first give the basic idea and then present the details of the scheme. In this solution, data requests and data replies are treated differently. The request message still follows the path shown in Fig. 2a; however, the reply message follows a different path. If no intermediate node needs to cache the data, N0 sends the data directly to N5 without going up to the cache layer. Suppose N3 needs to cache the data based on the cooperative cache protocol, as shown in Fig. 3. After N3 receives the request message, it modifies the message and notifies N0 that the data should be sent to N3. As a result, the data are sent from N0 to N3 through the cache layer, and then sent to N5. Note that the data will not go to the cache layer in intermediate nodes such as N1;N2, and N4 in this example.

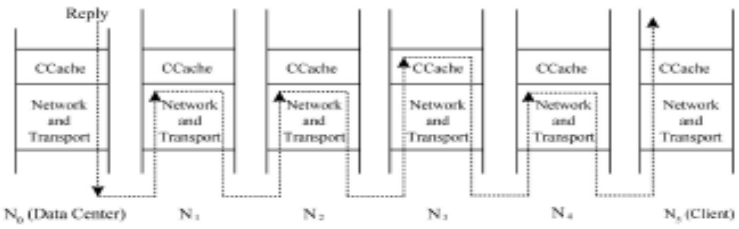

Fig. 3 in the asymmetric approach

In this way, the data only reach the routing layer for most intermediate nodes; the data reply only goes up to the cache layer at the intermediate nodes that need to cache the data. Intermediate node needs to cache the data. Although the request message always needs to go up to the cache layer, it has a small size, and the added overhead is limited.

The Asymmetric Approach

Our asymmetric caching approach has three phases:

Phase 1: Forwarding the request message. After a request message is generated by the application, it is passed down to the cache layer. To send the request message to the next hop, the cache layer wraps the original request message with a new destination address, which is the next hop to reach the data server (real destination). Here, we assume that the cache layer can access the routing table and find out the next hop to reach the data center. This can be easily accomplished if the routing protocol is based on DSR or AODV. In this way, the packet is received and processed hop by hop by all nodes on the path from the requester to the data server. In this the cache manager performs two tasks: First, it checks if it has the requested data in its local cache; if not, it adds its local information to the request packet. The local information includes the access frequency (number of access requests per time unit) of the requested data, channel used, and throughput of the link where the request is received. Its node id will also be added to Path List, which is a linked list encapsulated in the cache layer header. When the request message reaches the node who has the data, Path List in the message will include all the intermediate nodes along the forwarding path.

Phase 2: Determining the caching nodes. When a request message reaches the data server (the real data center or the intermediate node that has cached the requested data), the cache manager decides the caching nodes on the forwarding path and Then, the ids of these caching nodes are added to a list called Cache List, which is encapsulated in the cache layer header.

Phase 3: Forwarding the data reply. Unlike the data request, the data reply only needs to be processed by those nodes that need to cache the data. To deliver the data only to those that will cache the data, tunneling techniques [8] are used. The data reply is encapsulated by the cache manager and tunneled only to those nodes appearing in Cache List As shown in Fig. 2, suppose the intermediate node N3 needs to cache data di. Then, N3 and N5 are the nodes to process the data at the cache layer. N0 includes N3 and N5 in the cache header of the data item di, and first sets the destination address of di to be N3. When N3 receives any fragmented packet of di, the routing layer of N3 will deliver the packet upward to the transport layer and then to the cache layer. After the whole data item di has been received by N3, it caches the data item, sets the next destination using the next entry in Cache List, which is N5, and then passes the data down to the routing layer. After N5 receives the data, it delivers the data to the application

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

layer Tunneling delay (di,j(S)): the delay to forward a data item with size S from the cache layer of Ni to the cache layer of Nj, without handing the data up to the cache layer of any intermediate nodes. di,j(S) is hard to measure because it is affected by many factors, such as transmission interference, channel diversities, node mobility, etc. We assume di,j(s) is known at this point to introduce our optimal placement model. We will present heuristics to calculate it later.

Cache Placement Algorithms

Optimal cache placement: We first define a new term called aggregate delay, which includes the time to serve the current client request and the delay to serve future data requests coming from the same path. We can also assign different weights to these two parts of the delay based on the optimization objective, e.g., giving less weight to the future access delay if the current request has strict delay requirement. For simplicity, we assign equal weight for the current and future access delay in this paper. Below, we formally define the optimal cache placement problem.

A greedy cache placement algorithm

To get the optimal cache placement, the data server needs to compute the aggregate delay for every possible cache placement set. Since there are 2n (n is the length of the forwarding path) possible ways to choose cache placement set, which is too expensive. Therefore, we propose a greedy heuristic to efficiently compute the optimal cache placement. Let P*(k) be the optimal cache placement for a forwarding path when only the nearest k hops from the data server are considered as possible cache nodes. With the same condition, let L*(k) be the aggregate delay of the optimal n placement P*(k), and p*(k) be the hop distance of the farthest cache node from the data server in P*(k). When k =0, no cache node is between the data server and the client, and then the data server N0 transmits the data directly to the client Nn without reassembling at any intermediate node. All future requests from Ni need to get data from the data server. Therefore P*(o)=0,p*(k)=0 and

$$L^*(0) = d_{0,n}(S_D) + \sum_{i=1}^{n-1} f_i \cdot (d_{0,i}(S_D) + d_{0,i}(S_R)) \cdot \Delta t$$

Given l*(k),p*(k)and p*(k), we check whether to select the intermediate node Nk+1 as a cache node.

P: cache placement set
L: aggregate delay
Pos: hop distance of the farthest node from the data server
F[i]: Excluded data acess frequency on Ni
dD[i,j]: delay of forwarding the data item from Ni to Nj
dR[i,j]: delay of forwarding the data item from Nj to Ni
SD: size of the data item
SR: size of the data request
R[i]: Link throughput nodes Ni and Ni+1
T[I, j]: Link throughput nodes Ni and Nj
l[i]: Channel used for the link between nodes Ni and Ni+1
h(S): Chace processing cost for the data size of S
Δt: the expiration time of the data item

Greedy Cache Placement Algorithm

Get data access frequency on nodes—f[ ]
Get distance between node I and node i+1 store It into R [ ] (Same Cell)
Get distance between node I and node j store it into L[ ] (other Cell)
Get data Expiration time stored into t
Get size of data request stored into Sr (number of nodes in the cell)
Get size of the data item stored into Sd (size of the node)
Begin
For i=0 to number of nodes (n) -1 do
Distance between nodes in the same cell are stored into t[I,i+1]
For i=0 to n do
For j=i+1 to n do
Calculate delay times from node I to j, calculate delay time from node j to I stored into dD[I,j], dS[I,j] respectively, and store the distance between nodes from node I to j stored into t[I,j]
End for
End for
Set delay time (L) as first cell nth node data forwarding time

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

Set cache placement as null
Set pos as 0
For i=0 to n-1 do
Calculate the delay time and add it into L
End for
For k=0 to n-2 do
Calculate the aggregate delay time of transferring the data to another node and calculate the number of nodes the data will be moved and that mean value is stored into L' If l' value is less than l then l' is stored into l
Pos value as k+1;
End if
End for
End

III AUTHENTICATION & KEY EXCHANGE

The primary requirements by P2P security are:

- Need of secure communications for the P2P setting of virtual organizations (VOs). A VO typically is composed of users and resource providers beyond conventional organizational boundaries. Thus a centrally-managed security solution won't suit .
- Ease of use by users. An important element in this requirement is the need for provisioning "single sign-on" for users, including delegation of credentials for computations that involve multiple resources and/or sites.
- Applications of standard technologies. This not only facilitates fast and ready deployment of the P2P technologies, but also helps to ensure correct applications of security techniques. GSI actually meets these requirements very well.

As users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted. Thus, how to efficiently verify the correctness of outsourced P2P data without the local copy of data files becomes a big challenge for data storage security in P2P Computing. Note that simply downloading the data for its integrity verification is not a practical solution due to the expensiveness in I/O cost and transmitting the file across the network. Besides, it is often insufficient to detect the data corruption when accessing the data, as it might be too late for recover the data loss or damage. Considering the large size of the outsourced data and the user's constrained resource capability, the ability to audit the correctness of the data in a P2P environment can be formidable and expensive for the P2P users [8], [9]. Therefore, to fully ensure the data security and save the P2P users' computation resources, it is of critical importance to enable public auditability for P2P data storage so that the users may resort to a verifier , who has expertise and capabilities that the users do not, to audit the outsourced data when needed. Based on the audit result, verifier could release an audit report, which would not only help users to evaluate the risk of their subscribed P2P data services, but also be beneficial for the P2P service provider to improve their P2P based service platform. A digital certificate is the combination of a statement and a digital signature of the statement. The well-known digital certificate is the "X.509 public-key digital certificate" [1]. The statement generally contains the user's public key as well as some other information. The signer of the digital signature is normally a trusted certificate authority (CA). The X.509 public-key digital certificate has been widely used in public-key infrastructure (PKI) to provide authentication on the user's public key contained in the certificate. The user is authenticated if he is able to prove that he has the knowledge of the private key corresponding to the public key specified in the X.509 public-key digital certificate. However, the public- key digital certificate itself cannot be used to authenticate a user since a public-key digital certificate contains only public information and can be easily recorded and played back once it has been revealed to a verifier or Third party Auditor (TPA) In this paper, we propose an innovative approach which enables a user to be authenticated and a shared secret session key be established with his communication partner using any general form of digital certificates, such as a digital driver's license, a digital birth certificate or a digital ID, etc. We call this kind of digital certificate as a generalized digital certificate (GDC). A GDC contains user's public information and a digital signature of this public information signed by a trusted CA. However, in GDC, the public information does not contain any user's public key. Since user does not have any private and public key pair, this type of digital certificate is much easier to manage than the X.509 public-key digital certificates. The digital signature of the GDC is used as a secret token of each user. The owner of a GDC never reveals signature of GDC to a verifier in plaintext. Instead, the owner computes a response to the verifier's challenge to prove that he has the knowledge of the digital signature. Thus, owning a GDC can provide user authentication in a digital world.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

In addition, a secret session key can be established between the verifier and the certificate owner during this interaction. There are three entities in a digital certificate application. They are the following:

a) Certificate Authority (CA): CA is the person or organization that digitally signs a statement with its private key. In PKI applications, the X.509 public-key digital certificate contains a statement, including the user's public key, and a digital signature of the statement. The difference between the GDC and the existing public-key digital certificate is that in a GDC, the public information does not contain any user's public key.

b) Owner of a GDC: The owner of the GDC is the person who receives the GDC from a trusted CA over a secure channel. The owner needs to compute a valid "answer" in response to the verifier's challenged "question" in order to be authenticated and establish a secret session key.

c) Verifier: The verifier is the person who challenges the owner of a GDC and validates the answer using the owner's public information and CA's public key.

In this paper, our goal is to propose a similar solution in electronic-world applications. We call it the generalized digital certificate (GDC). A GDC contains public information of the user and a digital signature of the public information signed by a trusted certificate authority. The digital signature will never be revealed to the verifier. Therefore, the digital signature of a GDC becomes a security factor that can be used for user authentication. Our proposed scheme is closely related to the ID-based cryptography. In an ID-based cryptographic algorithms, each user needs to register at a private key generator (PKG) and identify himself before joining the network. Once a user is accepted, the PKG will generate a private key for the user. The user's identity (e.g. user's name or email address) becomes the corresponding public key. In this way, in order to verify a digital signature of a message, the sender sends an encrypted message to a receiver; a user only needs to know the "identity" of his communication partner and the public key of the PKG, which is extremely useful in cases like wireless communication where pre-distribution of authenticated public keys is infeasible. However, in an ID-based cryptographic algorithm, it is assumed that each user already knows the identity of his communication partner. Based on this assumption, there is no need, nor have feasible ways, to authenticate the identity. This is the main advantage of ID-based cryptography. Due to this assumption, ID-based cryptography is only limited to applications that communication entities know each other prior to communication. While in our proposed GDC scheme, the user does not need to know any information of his/her communication partner. The public information of a GDC, such as user's identity, can be transmitted and verified by each communication entity. Furthermore, this information is used to authenticate each other. In other words, our proposed schemes support general PKI applications, such as Internet e-commence, that communication entities do not need to know each other prior to the communication. Our proposed solution is based on the combination of a conventional DL based digital signature scheme and the well-known (generalized) Diffie- Hellman assumption.

ElGamal Digital Signature

In the ElGamal scheme [25], a large prime p and a generator g in the order of $p-1$ are assumed to be shared by all users. The signer selects a random private key $x \in [1, p-2]$ and computes the corresponding public key $y = gx \bmod p$. The signer first randomly selects a secret parameter $k \in [1, p-1]$ with $\gcd(k, p-1) = 1$ and computes $r = g^k \bmod p$. Then, s is solved by knowing the signer's secrets, x and k, as

$$m = ks + rx \bmod p - 1,$$

where m represents the message digest of the message m'. (r, s) is defined as the digital signature of the message m'. The signature (r, s) can be verified by checking whether the equation is correct

$$g^m = y^r r^s \bmod p,$$

In an ElGamal signature scheme, the parameter r of the signature can be computed off-line as $r = gk \bmod p$. The signature component s is computed on-line. Readers can refer to [26] for more discussion on the design of DLbased signature schemes. Without loss of generality, we can represent the generalized signing equation for all DL-based signature schemes as $ax = bk + c \bmod p - 1$ where (a, b, c) are three parameters from the set of values (m, r, s). More specifically, each parameter can be a mathematical combination of (m, r, s). For example, the parameter a can be m, r or s. The verification equation is determined accordingly as $y^a = r^b g^c \bmod p$. There are 18 generalized ElGamal-type signature variants.

Assume A and B have their private keys, xA and xB, and their corresponding public keys, $yA = g^{xA} \bmod p$ and $yB = g^{xB} \bmod p$, respectively, where p is a large prime integer and g is a primitive element of the multiplicative group modulo p. Only A and B can compute a shared secret $K_{AB} = y_B{}^{xA} = y_A{}^{xB} = K_{BA} \bmod p$. DHA refers to the assumption that it is computationally infeasible to determine KA, without knowing the private key x_A or x_B. However, solving the private key x_A or x_B from the corresponding public key y_A or y_B is equivalent to solving the discrete logarithm problem.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

User Authentication and Key Establishment Protocol

1) Registration at CA: Let A be the certificate owner and B be the verifier. A needs to register at a CA to obtain a GDC. The CA generates an ElGamal signature (rA, sA) for user A's statement $m'A$ according to equation (1), where mA is the message digest of the statement $m'A$. Since the signature component rA is a random integer and does not depend on mA, it does not need to be kept secret. However, the signature component sA is a function of the statement. Each owner needs to keep it secret from the verifier in the authentication process.

2) Protocol: The authentication and key establishment protocol contains the following four steps:

a) The user A passes his user information $m'A$ and parameters(r_A, S_A) to the verifier B, where $S_A = r_A{}^{sA} \bmod p$.

b) After receiving m'_A and (r_A, S_A), the verifier checks whether $g^{mA} = y_{SA}{}^{rA} \bmod p$,

where y is the public key of the CA. If this equality holds true, the verifier B first randomly selects an integer $vB \in [1, p-2]$, then computes a challenge $c_B = r_A{}^{vB} \bmod p$ and send c_B to the user A. Otherwise, the user authentication fails and the protocol is stopped.

3) The user A first uses his secret sA to compute the Diffie-Hellman secret key $K_{A,B} = c_B{}^{sA} \bmod p$, $K'_{A,B} = D(KA,B)$, where $D(K_{A,B})$ represents a key derivation procedure with $K_{A,B}$ as an input. Then user A randomly selects an integer $vA \in [1, p-2]$, computes $c_A = r_A{}^{vA} \bmod p$ and the response $Ack = h(K'_{A,B}, c_B \| c_A)$, where $h(K'_{A,B}, c_B \| c_A)$ represents a one-way keyed hash function under the key $K'_{A,B}$. The user A sends Ack and c_A back to B.

4) After receiving the Ack and c_A from the user A, the verifier B uses his secret v_B to compute the Diffie-Hellman shared secret key $K_{B,A} = S_A{}^{vB} \bmod p$, $K'_{B,A} = D(K_{B,A})$, and checks whether $h(K'_{B,A}, cB \| cA) = Ack$ is true. If this verification is successful, the certificate owner A is authenticated by the verifier B and a onetime secret session key ${}_B{}^{vA} = r_A{}^{vAvB} = c_A{}^{vB} \bmod p$ is shared between A and B. This shared key can provide perfect forward security.

In order to be authenticated successfully by the verifier, in our protocol, the certificate owner needs to compute and send a valid pair (r_A, S_A) and Ack to the verifier in steps 1) and 3). The parameters (r_A, SA) need to satisfy $g^{mA} = y^{rA} S_A \bmod p$. This pair of integers can be easily solved by anyone. However, we want to show that only the certificate owner A who knows the secret exponent of S_A can compute a valid Ack. This is because the verifier B can compute the one-time secret key $K_{B,A}$ used in generating the Ack as $K_{B,A} = S\,A^{vB} = r_A{}^{sAvB} \bmod p$.

According to the DHA, the certificate owner A who knows the secret exponent of S_A can also compute $K_{A,B}$ as $K_{A,B} = c_B{}^{sA} = r_A{}^{sAvB} = K_{B,A} \bmod p$. Thus, the certificate owner can interact with the verifier and be authenticated successfully.

Security Analysis and Discussion

In this section, we will analyze the security of the proposed user authentication and key establishment protocol for the unforgeability, one-wayness and no transferability.

a**) Unforgeability**: In order to perform a forgery attack, the attacker needs to present a valid pair (r_A, S_A) in step 1) and the corresponding Ack in step 3) in order to impersonate the certificate owner successfully. A valid pair (r_A, S_A) alone in step 1) cannot be used to authenticate the certificate owner since this pair of parameters can be solved easily by the attacker from equation (3). However, it is computationally infeasible for the attacker to find the discrete logarithm of SA because the security of the ElGamal signature scheme.Therefore, it is computationally infeasible for the attacker to get a pair (r_A, S_A) to satisfy $g^{mA} = y^{rA} r^{sA}{}_A \bmod p$. Due to the DHA, without knowing the secret exponent of S_A, it would be infeasible for the attacker to compute $K_{A,}$ and forge a valid Ack in step (3). On the other hand, the certificate owner obtains the secret exponent of S_A from CA during the registration and the certificate owner can be authenticated successfully in step 3). In summary, the security of the unforgeability of our proposed protocol is provided through combination of the security of the ElGamal signature scheme and the DHA. Therefore, the proposed user authentication and key establishment protocol is secure against forgery attacks.

b) **One-wayness**: In step 1), the certificate owner presents SA to the verifier. The computation of secret s_A from S_A is infeasible since computation of s_A from the S_A is a discrete logarithm problem. Also, in step 3), the certificate owner uses the secret s_A to compute the Diffie-Hellman key $K_{A,B}$. Although the verifier knows the Diffie-Hellman key $K_{A,,}$ but due to the DHA, the verifier cannot obtain the secret s_A. Therefore, our proposed protocol satisfies the one wayness property.

c) **No transferability:** Due to the DHA, a valid response Ack can only be generated by a certificate owner who knows the secret digital signature component s_A such that $r_A{}^{sA} = S_A \bmod p$, or by a verifier who knows the random secret of a random challenge selected by the verifier. As the verifier selects a random challenge each time, the

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

response is only valid for a one-time authentication. Since the digital signature of a GDC is never passed to the verifier, the verifier cannot pass the complete GDC to any third party. There is no privacy intrusion problem in our protocol.

herefore, a valid response Ack cannot be transferred into a response of another verifier's challenge. Our protocol enables a certificate owner to be authenticated and two one-time shared secret keys K_A, and $c_B{}^{vA} = r_A{}^{vAvB} = c\,A^{vB}$ mod be established between A, the certificate owner, who knows s_A such that $r_A{}^{sA} = S_A$ mod p, and the verifier B through the authentication protocol. The former is used to generate the Ack, and the latter is established shared secret key between A and B. In addition, it enables the owner to send a confirmation Ack to the verifier. Since the Diffie- Hellman secret shared key can be generated by either A or B, the certificate owner A can deny participating in the protocol. In the original DHA, it is assumed that the generator g is a primitive element of the multiplicative group modulo p; while the parameter $rA = gk$ mod p in Theorem 1 is not necessarily a generator. However, we can ensure that rA is a primitive element of the multiplicative group modulo p by requiring $(k, p-1) = 1$ [27]. Particularly, when $p = 2p'+1$ is a safe prime, where p' is also a prime, we can ensure rA is a primitive element of the multiplicative group modulo p if k is an odd number. Similar to the ID-based cryptographic algorithms, our proposed protocol also has the key escrow problem, that is the CA knows the one-time secret session key shared between the users. Some cryptographic algorithms have been proposed to solve the key escrow problem of the ID based signature (IBS) while enjoying the benefits of the IBS, such as certificate less digital signature (CDS).

IV .CONCLUSION

In this paper, we propose a Cooperative cache & privacy-preserving owner auditing system for data caching security in p2p Environments. We utilize Greedy Algorithm, GDC & the Hash key generated through Elgamal Digital Signature & Deffie Hellman Key exchange Protocol to guarantee that Verifier or TPA would not learn any knowledge about the data content stored in the intermediate systems during the efficient auditing process, which not only eliminates the burden of Owner from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. And also the user does not have any private and public key pair, this type of digital certificate is much easier to manage than the X.509 public-key digital certificates Considering TPA may concurrently handle multiple audit sessions from different Owners for their outsourced data files, where TPA can perform the multiple auditing tasks in a batch manner.

REFERENCES

[1] Network Working Group, "Internet X.509 public key infrastructure certificate and crl profile, RFC: 2459," Jan. 1999.

[2] C. Tang and D. Wu, "An efficient mobile authentication scheme for wireless networks," IEEE Trans. Wireless Commun., vol. 7, pp. 1408-1416, Apr. 2008.

[3] G. Yang, Q. Huang, D. Wong, and X. Deng, "An efficient mobile authentication scheme for wireless networks," IEEE Trans. Wireless Commun., vol. 9, pp. 168-174, Jan. 2010.

[4] J. Chun, J. Hwang, and D. Lee, "A note on leakage-resilient authenticated key exchange," IEEE Trans. Wireless Commun., vol. 8, pp. 2274-2279, May 2009.

[5] D. Chaum and H. van Antwerpen, "Undeniable signatures," Advances in Cryptology - Crypto'89, Lecture Notes in Computer Science, vol. 435, pp. 212-217, 1989.

[6] M. Bohøj and M. Kjeldsen, "Cryptography report: undeniable signature schemes," Tech. Rep., Dec. 15, 2006.

[7] X. Huang, Y. Mu, W. Susilo, and W. Wu, "Provably secure pairing-based convertible undeniable signature with short signature length," Pairing- Based Cryptography -C Pairing 2007, vol. 4575/2007 of Lecture Notes in Computer Science, pp. 367-391, Springer Berlin / Heidelberg, 2007.

[8] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," Advances in Cryptology - EUROCRYPT, pp. 143-154, 1996. LNCS Vol 1070.

[9] Network Working Group, "Internet X.509 public key infrastructure certificate and crl profile, RFC: 2459," Jan. 1999.

[10] R. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," Advances in Cryptology-ASIACRYPT, Lecture Notes in Computer Science, vol. 2248/2001, Springer Berlin / Heidelberg, 2001.

[11] J. Ren and L. Harn, "Generalized ring signatures," IEEE Trans. Dependable Secure Comput., vol. 5, no. 4, Oct.-Dec., pp. 155-163, 2008.

ISSN: 2319-5967
ISO 9001:2008 Certified
International Journal of Engineering Science and Innovative Technology (IJESIT)
Volume 2, Issue 1, January 2013

[12] S. Saeednia, S. Kremer, and O. Markowitch, “An efficient strong designated verifier signature scheme," ICISC 2003, vol. 2836 of Springer Lecture Notes in Computer Science, pp. 40-54, 2003. HARN and REN:

AUTHOR BIOGRAPHY

1. N.Sandeep Chaitanya, doing his research in JNTUH, received his B.Tech from DVRCET, Hyd, AP in Computer Science & Information Technology and M.Tech in Information Technology from Sathyabama University, Chennai. Presently he is working as Associate Professor at CMRCET. His research interests include Cloud Computing, Grid Computing, Mobile Networking, Computer Networks and Network Security.

2. P Sruthi received M.Tech & B.Tech from JNTUH. Presently she is working as Associate Professor at CMRCET. Her research interests include Cloud Computing, Grid Computing & P2P Computing.

3. A Nandini received B.Tech from JNTUH. Presently she is working as Process Executive in Cognizant Technology Solutions. Her research interests include Mobile ad-hoc networks, Cloud Computing, Grid Computing & P2P Computing**.**

4. T V Suresh Kumar doing his research received B.Tech & M.Tech from JNTUH. He is working as Assoc. Professor at VEC... His research interests include Wireless Networks, VLSI, Image & Speech Processing.

5. S P Santhosh, received his B. Tech from JNTUH. His research interests include Grid computing & Computer Networks P2P Computing & Internet