Leveraging Identity-based Cryptography for Node ID Assignment in Structured P2P Systems

Sunam Ryu, Kevin Butler, Patrick Traynor, and Patrick McDaniel
Systems and Internet Infrastructure Security Laboratory
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
Email: {ryu, butler, traynor, mcdaniel}@cse.psu.edu

***Abstract*— Structured peer-to-peer systems have grown enormously because of their scalability, efficiency and reliability. These systems assign a unique identifier to each user and object. However, current assignment schemes allow an adversary to carefully select user IDs and/or simultaneously obtain many pseudo-identities—leading ultimately to an ability to disrupt the P2P system in very targeted (and dangerous) ways. In this paper, we propose novel ID assignment protocols based on identity-based cryptography. This approach permits the acquisition of node IDs to be tightly regulated without many of the complexities and costs associated with traditional certificate solutions. We broadly consider the security requirements of ID assignment and present three protocols representing distinct threat and trust models. A detailed empirical study of the protocols is given. Our analysis shows that the cost of our identity-based protocols is nominal, and that the associated identity services can scale to millions of users using a limited number of servers.**

I. Introduction

Peer-to-peer networks are now ubiquitous. They provide a resilient media for the efficient storage and retrieval of file objects. Such models change the nature of storage and provide a vector toward dynamic and massively distributed global information sharing. However, while the object sharing techniques have advanced rapidly, security services protecting this media have yet to mature. This is largely due to a highly diverse, untrusted, and often anonymous user community.

Structured P2P systems assign a unique key identifier (ID) to every object and node. IDs associated with objects are mapped by P2P overlay protocols to the node responsible for that object. The assignment of node IDs is therefore critically important to the efficiency and security of the peer-to-peer system. However, current peer-to-peer systems use node ID assignment techniques that can be trivially manipulated by an adversary. For example, adversaries can strategically craft node IDs (directly or by forged addressing) to assert control over targeted objects. Further *Sybil* attacks allow an adversary to control large portions of the peer-to-peer network by simultaneously obtaining many identities [10]. Proposed solutions to these problems largely rely on the use of trusted certificate authorities and a structured public-key infrastructure (PKI) to assign and certify node IDs. These schemes, however, require maintenance of complex PKI systems, which can be difficult or infeasible to implement in practice.

In this paper we consider the use of identity-based cryptography to assist in the security and performance critical assignment of user identities in peer-to-peer systems. Identity-based cryptosystems use textual strings to derive public keys from cryptographic parameters advertised within a domain. This approach avoids many of the complexities of PKI usage (a user's public key is directly derivable from their identity), and reduces the overheads associated with authentication. We exploit these features in peer-to-peer systems by assigning an ID and providing the associated identity-based private key to each joining node. Nodes are weakly authenticated via callback: any node capable of *receiving* a TCP connection at an IP address is deemed the legitimate owner of that IP address. These mechanisms work in concert to provide for authenticated node identity and strongly limit damaging Sybil attacks.

We identify three protocols representing diverse trust models and performance profiles based on identity-based cryptography: a fully decentralized ID-based assignment scheme (protocol 1), a centralized scheme in which a single host plays dual roles as ID assigning authority and P2P bootstrap node (protocol 2), and an approach that retains the separation of duties of a decentralized model at a low cost by using a hybrid of identity-based and symmetric key cryptography (protocol 3). We have built functional ID client and server implementations and tested them in our laboratory environment.

Our empirical analysis considers the relative performance of the protocols and their scalability. We found that a fully decentralized scheme (protocol 1) induces delays of over twice that of the centralized scheme (protocol 2), i.e., average observed delay is 280 milliseconds in protocol 1 vs. 115 milliseconds in protocol 2. We also found that we could achieve protocol run-times similar to centralized solutions with a decentralized architecture using a hybrid symmetric and ID-based cryptographic approach (protocol 3, with observed average delay of 120 milliseconds). Further analysis shows that by applying MNT elliptic curve and random oracle optimizations to the identity-based cryptographic algorithms (whose operation dominates protocol costs), we can reduce protocol costs by as much as a factor of 5.

Any solution that limits the scalability of a peer-to-peer system is unlikely to be widely adopted. Our analysis found

that our protocols could scale easily, where 2 servers could conservatively sustain a community of over 600,000 nodes, and 50 ID servers could support over 15,000,000 nodes.

The rest of this paper is organized as follows. Section II gives a brief overview of structured P2P networks and identity-based cryptography, and identifies the broad goals and assumptions of this work. Section III describes our novel protocols in detail. Section IV describes an empirical evaluation of the proposed approach. Open problems and operational issues are discussed in Section V. Section VI discusses important related work and Section VII concludes.

II. Background

This section presents relevant background in peer-to-peer systems and identity-based encryption, and describes the security and performance goals of our approach.

A. Structured P2P overlay protocols

Structured overlays are designed to allow for scalable, efficient and reliable object placement within a dynamic virtual topology. To generalize, every node and object in a peer-to-peer system is assigned a unique identifier (ID).[1] A node locates an object by mapping the *object key* (the object's ID) to a node ID responsible for that object. The responsible node then supplies the object directly or indicates where/how it can be acquired. The P2P network is *structured* by arranging the nodes in such a way as to allow for efficient routing (searching) to the current responsible node for a given object. For example, $O(\log n)$ searching is achieved by arranging nodes in binary searchable topologies, e.g., rings.

In the representative systems Chord [23], Pastry [19] and Tapestry [25], node IDs are deterministically assigned by hashing the host's IP address. Conversely, in CAN [16], every node randomly picks its own node ID upon entering the system. In these systems an adversary can carefully select identities (either directly or by IP spoofing) such that they become the responsible node for sensitive objects. In a related technique, an adversary mounts a *Sybil* attack by obtaining a large number of simultaneous identities [10]. These identities probabilistically interpose the adversary in the routing paths for a great many objects, and thus permitting it to disrupt or manipulate the search process [5], [21], [24].

B. Identity-based cryptography

Public keys in identity-based public key cryptosystems are simple data objects [3], [7], [20], e.g., ASCII string email address. Associated with ID cryptosystems are a set of well-known public parameters used to generate the cryptographic material used for decryption or signature verification. A trusted third party, called the *private key generator* (PKG), generates the corresponding private key using secret information associated with the public parameters. Using this construct, anyone can encrypt messages or verify signatures without prior key distribution beyond the the dissemination of public parameters and the public key "strings". This is useful where the deployment of a traditional certificate authority-based public key infrastructure is inconvenient or infeasible.

A central operational consideration of ID-based cryptography is that the private keys must be obtained from the PKG. How one securely and efficiently obtains this private key is essential to the security of the supported system. For example, how the PKG decides who should be given the private key associated with an email address is crucial to maintaining the integrity of the system. Another consideration is cost: key generation can be computationally expensive (see Evaluation in Section IV). To ease the computation burdens of PKG operation, hierarchical identity-based encryption (HIBE) [2], [11], [12] can be used to reduce the overload of a root PKG by replicating private key generation to slave PKGs.

C. Protocol Setup

This work is focused on the secure assignment and authentication of pseudo-identities in peer-to-peer systems. As such, we define the following goals of the system:

- *Secure ID assignment* - each user must be given a unique pseudo-identity (or just "identity" throughout) to which he can later be authenticated. The user must not be able to influence the content of that ID in any way, e.g., she cannot select or predict the ID.
- *Sybil attack mitigation* - the number of simultaneous pseudo-identities a node can acquire should be bounded by the system.
- *Pseudo-identity authentication* - other participants should be able to authenticate all users (nodes) in the system.
- *Limited overheads* - the costs associated with use of the IDs should be nominal.
- *Simplicity* - the complexity of the creation, maintenance, and use of the system should be low.

We assume that the network is not secure: any IP address can be spoofed, and any message can be intercepted and altered by an adversary. Moreover, the network and all servers and participants in the system may fail in arbitrary ways. We place no restriction on the number of compromised or adversarial nodes in the system.

Discussed more fully in the protocols that follow, each joining node is weakly authenticated via callback: all responses to requests are transmitted through a server-initiated TCP connection (see protocols for details). Further, each such communication is protected by a secure channel established via a Diffie-Hellman exchange [9]. We further assume the existence of some loosely synchronized secure clock.

The following notation is used throughout:

[1] These identities are transient pseudonyms for the real users, and hence are often referred to as "pseudo-identities". We use the terms identity and pseudo-identity interchangeably throughout.

N : new node that joins the system
O : other nodes participating in the system
TP : third party
BN : bootstrap node
AS : node ID assignor
IP_A : node A's IP address
ID_A : node A's ID assigned by hashing the IP address
K_A^+, K_A^- : node A's public key and private key
$K_{A \cdot B}$: shared secret key between node A and node B
$E(m, k)$: encryption of message m using the key k
$HMAC(m, k)$: keyed-hash message authentication code of message m using the key k
$Sign(m, k)$: signature of message m using the key k
TS_i : time stamp
$a \parallel b$: concatenation of two strings, a and b

III. Protocol Specification

In the following subsections, we present three protocols that authenticate node IDs and protect structured peer-to-peer networks against Sybil attacks. Each protocol differs in functionality based on the architecture where the system is to be deployed. We describe each protocol's specification and operation, and briefly discuss the tradeoffs inherent to each approach.

A. Protocol 1 : Trusted Third Party

In the first protocol, the binding between a node's ID and its private key is performed by a *trusted third party* (TTP), as shown in Figure 1. This serves a similar function to a centralized authority in the traditional public key infrastructure. Here, the TTP[2] assigns random node IDs and generates the corresponding private keys. Unlike the traditional PKI model, however, we leverage identity-based cryptographic techniques to link user identities with keys.

1) System Setup: Before the system is brought online, the TTP generates a master key. The TTP then publishes the system parameters, which allows nodes to generate public keys from the identifying strings (e.g., an IP address) of other nodes. Private keys are generated at the TTP using the master key, public parameters, and the identifying string [20].

2) Node Join: In order for a node N to join an overlay network, it first contacts the trusted third party TTP and provides its IP address. After weakly authenticating its identity via callback, TTP gives the node a randomly-generated ID and the corresponding private key. N then contacts the bootstrap node BN[3] and provides BN with its ID and a timestamp, both signed with N's private key. Upon receiving and verifying a join request from N, BN returns a signed copy of the timestamp and node ID.

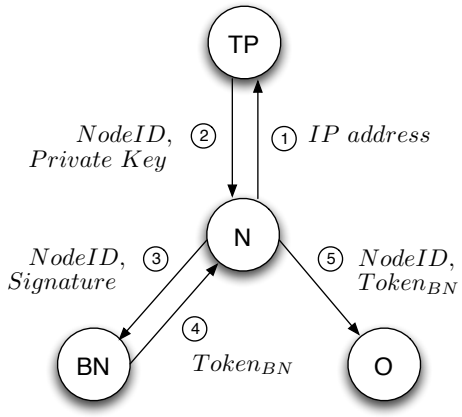

Fig. 1. Node join in TTP protocol (protocol 1)

A more formal expression of the protocol is as follows:

1) $N \rightarrow TTP : IP_N$
2) $TTP \rightarrow N : ID_N,\ E\ (K_N^-, K_{TP \cdot N})$
3) $N \rightarrow BN : ID_N,\ TS_1,\ Sign\ (ID_N \parallel TS_1, K_N^-)$
4) $BN \rightarrow N : Sign\ (ID_N \parallel TS_1, K_{BN}^-)$
5) $N \rightarrow O : ID_N,\ TS_1,\ Sign\ (ID_N \parallel TS_1, K_{BN}^-)$

Because BN has signed the response, it can be used as a token of authenticity when N contacts another node O to join the overlay. O can verify the signature, which acts as a proof that the node ID and IP address of N are correlated, without the need for a certificate from BN. Because O knows BN's identity, it can generate BN's ID-based public key and validate the token, allowing N into the overlay.

To update its private key, N contacts the trusted third party TTP at some later time and provides the signature generated by using its current private key and the previous issue-date. After checking these values, TTP issues the updated private key including new issue-date. To rejoin the system, N contacts BN with the signature using its updated private key and, if verified, receives the updated token.

The Sybil attack is prevented because of the callback behavior: only if the node can be reached at the IP address given will it receive a response from the bootstrap node. Note that if the node generates a spoofed IP address, but the adversary is able to route the response back to it, the adversary already has effective control of the spoofed IP address, and for all purposes can act as the owner of that address. This is not an example of a simple spoofing attack, which is possible to implement against other P2P network protection schemes. The node can verify the authenticity of BN, as BN's identity is

[2]As with traditional centralized authorities, the procedure of requesting and transmitting private keys can be offline to reduce the possibility of revealing private keys generated by the third party.

[3]Finding the bootstrap node is application-specific. We assume that a new node joining the network knows initially about the bootstrap node that is already part of the system.

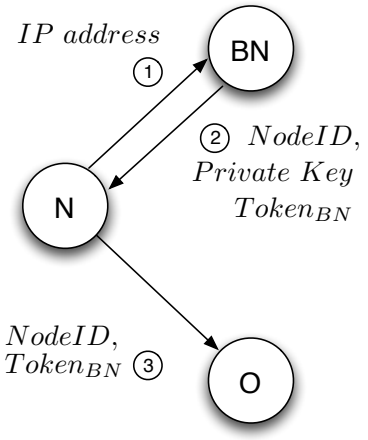

Fig. 2. Node join in Trusted Bootstrap Node protocol (protocol 2)

known and hence, its associated public key is also known, due to the use of identity-based cryptography. The node N can then validate the signature using BN's public key.

This protocol can noticeably reduce cost and system complexity compared to a traditional public key infrastructure, as it requires neither prior key distribution nor certificates. The decentralized nature of this architecture also provides for the separation of duties for policy and enforcement in the system. Moreover, it not only guarantees that node IDs are assigned at random but can also control the available period of node IDs through simple key expiration.

B. Protocol 2 : Trusted Bootstrap Node

In contrast with the previous scheme, the Trusted Bootstrap Node protocol shown in Figure 2 implements a centralized system. Specifically, instead of relying upon a trusted third party to perform the duties of key distribution, the bootstrap node becomes the arbiter of network membership and trusted information. In so doing, this approach attempts to minimize the overhead and complications associated with a decentralized architecture.

1) Setup: In a similar fashion to a trusted third party, the trusted bootstrap node publishes the system parameters and keeps a secret master key. The bootstrap node uses its master key to create the corresponding private keys and to generate random node IDs.

2) Node Join: When N attempts to join the network, it sends its IP address to BN. BN weakly authenticates N's identity through callback. Should N successfully demonstrate control over its claimed IP address, BN generates and assigns a node ID, a corresponding private key and a token to be used for authentication with member nodes in the network. N then contacts O with the token received from BN. Using the public key of BN, O checks the validity of the token. Note that this token is only valid from the IP address bound to the token itself, making its use by other nodes insufficient for gaining network membership.

The message exchange is as follows:

$$
\begin{aligned}
&1)\ N \rightarrow BN : IP_N \\
&2)\ BN \rightarrow N : ID_N,\ E\ (K_N^-, K_{BN \cdot N}),\ TS_1, \\
&\qquad\qquad Sign\ (ID_N \parallel TS_1, K_{BN}^-) \\
&3)\ N \rightarrow O : ID_N,\ TS_1,\ Sign\ (ID_N \parallel TS_1, K_{BN}^-)
\end{aligned}
$$

To renew the private key or rejoin the network, N contacts BN with the signature using its current private key and the previous issue-date. If verified, N receives the new private key or the new token from BN.

The major advantage of this protocol is the reduction in overhead associated with the interaction of a third party. This can simplify the procedure of joining a node, as the bootstrap node deals with both assigning node IDs and generating private keys. In a similar fashion to the TTP protocol, it can also guarantee random node ID assignment and control the available period of node IDs through simple key expiration. A deeper analysis of the tradeoffs inherent to this scheme is performed in Section V.

C. Protocol 3 : Multiple Node ID Assignors

The previous two protocols trade off the separation of duties inherent to a decentralized architecture with the overall performance of a centralized scheme. Ideally, a hybrid of these two approaches could be created to provide the strengths of both systems while minimizing their implementation-related drawbacks. This section examines such a construction in the Multiple Node ID Assignors protocol. Specifically, a single bootstrap node generates only the private keys and delegates the authority of assigning node IDs to one of many trusted nodes. To reduce the cost of this operation, we leverage the inherent trust between the bootstrap and assignor nodes. In this, we assume that the bootstrap and assignor nodes privately share a symmetric cryptographic key that is used to provide efficient token generation.

1) Setup: Prior to operation, the bootstrap node selects the trusted nodes for assigning node IDs and establishes secret keys with them. The bootstrap node generates the system parameters to be published and provides those nodes with the parameters for node ID assignment. Like the previous two protocols, this scheme also guarantees random node ID assignment by preventing a node from choosing its own node ID.

2) Node Join: When N attempts to join the network, as shown in Figure 3, it transmits its IP address to a trusted assignor node AS. After verifying the identity, AS generates the node ID and issues a timestamped token as proof of authentication. Upon verification of a token sent from N, BN provides both a private key and a second token to be used for proving N's authenticity to O. Formally, the message flow is as follows:

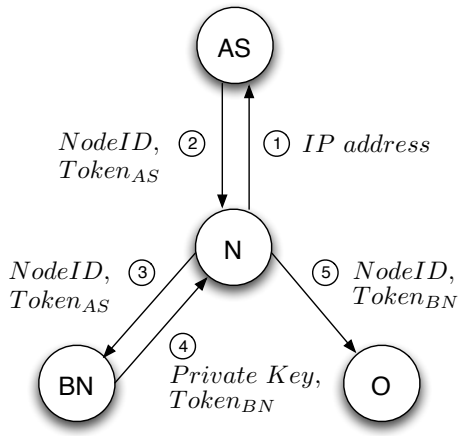

Fig. 3. Node join in Multiple Node ID Assignors protocol (protocol 3)

1) $N \rightarrow AS : IP_N$
2) $AS \rightarrow N : ID_N,\ TS_1, HMAC\ (ID_N \parallel TS_1, K_{AS \cdot BN})$
3) $N \rightarrow BN : ID_N,\ TS_1, HMAC\ (ID_N \parallel TS_1, K_{AS \cdot BN})$
4) $BN \rightarrow N : E\ (K_N^-, K_{BN \cdot N}),\ TS_2,\ Sign\ (ID_N \parallel TS_2, K_{BN}^-)$
5) $N \rightarrow O : ID_N,\ TS_2,\ Sign\ (ID_N \parallel TS_2, K_{BN}^-)$

To renew the private key or rejoin the network, N contacts BN with the signature using its current private key and the previous issue-date. If verified by BN, N receives the new private key or the new token from BN. A detailed analysis of the benefits and tradeoffs inherent to this scheme is discussed in Section V.

IV. Evaluation

In this section we consider the cost of the three protocols presented in the preceding section. Used in this analysis, we have built an initial implementation written of all three protocols in C. We use the GNU GMP library for all standard cryptographic algorithms, 128-bit AES for non-identity-based encryption, and SHA-1 for hashing. All identity-based cryptographic algorithms use the pairing-based cryptography (PBC) library [13]. We parameterized the library to use supersingular elliptic curves over a non-random oracle construction [15] and Cha-Cheon signatures (see library documentation for details). All experiments were executed using a dual processor G5 (server) and a Mac mini 1.5Ghz G4 (client), both of which were running the Apple OS X 10.4.7 operating system. All results reported below represent the average of 1,000 executions of the protocol or other measured function. We report numbers based on the cost of each step as well as the cumulative runtime (in milliseconds).

Operation	**Cost**	σ
Key creation	37.340	2.302
Node signature	80.722	3.548
Request verification	74.649	5.157
ID token creation	20.095	1.036
Symmetric-key token creation	0.131	0.052

TABLE I
Cryptographic Microbenchmarks (in msecs)

	TTP Exchange		BN Exchange	
Message	One	Two	Three	Four
Server (step)	3.376	46.510	166.243	63.781
Server (cumulative)	3.376	49.886	216.129	279.910
Client (step)	0.266	53.188	161.697	63.570
Client (cumulative)	0.268	53.456	215.153	278.723

TABLE II
TTP protocol performance (in msecs)

A. Cryptographic Microbenchmarks

There are five significant cryptographic operations used in the protocols defined in this paper: the creation of the identity-based key (all protocols), the signing of the ID request (protocol 1), the verification of the node request (protocol 1), the creation of the ID-token (all protocols), and the creation of a symmetric key-based token (protocol 3). Note we omit verification of the token by the other nodes in the P2P system as it does not relate to the bootstrap process (these costs, however, are largely identical to those of ID request verification). The measured costs are detailed in Table I.

The signature and subsequent verification operations are appreciably more expensive than the other operations. Such results are not unexpected, as they represent the most computationally intensive operations in identity-based cryptography (for the supersingular curve). Additionally, because the client is notably less powerful than the server in these experiments, signing is approximately equal to verification. Note an important corollary: in client and server hosts with similar computational resources, verification would be significantly faster than signing.

Message	One	Two
Server (step)	1.750	115.319
Server (cumulative)	1.750	117.069
Client (step)	0.266	115.640
Client (cumulative)	0.266	115.906

TABLE III
Trusted Bootstrap Node protocol performance

	ID Exchange		BN Exchange	
Message	One	Two	Three	Four
Server (step)	2.813	0.132	2.378	115.965
Server (cumulative)	2.813	2.945	5.323	121.288
Client (step)	0.265	3.355	0.029	116.539
Client (cumulative)	0.265	3.620	3.649	120.188

TABLE IV
MULTIPLE NODE ID ASSIGNORS PROTOCOL PERFORMANCE

B. Protocol Benchmarks

We now break down the per-flow and total costs for each of the protocols. Table II presents the results for the four messages composing the TTP protocol (protocol 1). The first two messages implement an exchange between the node and trusted third party to obtain the private key and node ID. Messages 3 and 4 are used to authenticate the joining node to the bootstrap node and obtain the token used to prove ownership of the ID to other P2P members (see Figure 1 and associated text in Section III-A for further detail).

There are several aspects of the performance analysis of protocol 1 that warrant comment. First, the exchange between the node and the TTP is relatively fast. As noted in the previous section, an ID-based key takes about 37 milliseconds to create. This accounts for approximately 80% of the time required for this exchange, with the remaining 20% attributed to network delay, software initialization, etc. The third message (first message of the bootstrap node exchange) consumes about 60% of the total delay per protocol iteration - a result of both the client signature and subsequent signature verification. The last message cost can be attributed to signature costs associated with token generation.

The Trusted Bootstrap Node protocol (protocol 2) combines all of the server functions into a single flow, where the user obtains the ID, the private key, and the token in the same exchange. This leads to a simplified performance analysis shown in Table III. Note that the average execution time is less than half that of protocol 1. This is due to the fact that the single exchange eliminates a signature creation and verification, and reduces the communications overhead by eliminating additional messages between the client and server. However, this efficiency has a cost: all server functions (and hence all trust) must be placed in a single authority. This may not be appropriate (or even feasible–see scalability below) in many environments.

In the first exchange of protocol 3 (messages 1 and 2), the node obtains an ID and (symmetric key) token from an ID assignment server. The node obtains the private key and secondary (identity-based) token from the bootstrap node in the second exchange (messages 3 and 4). A cost analysis of this protocol is presented in Table IV.

Protocol 3 retains the separation of duties between the different servers while retaining low cost. For example, the ID exchange fulfills the same purpose as the the TTP exchange in protocol 1 at $1/20$th the cost. This is achieved by applying symmetric key cryptography: the ID authority and the bootstrap node exploit a shared secret to secure communication between the two.

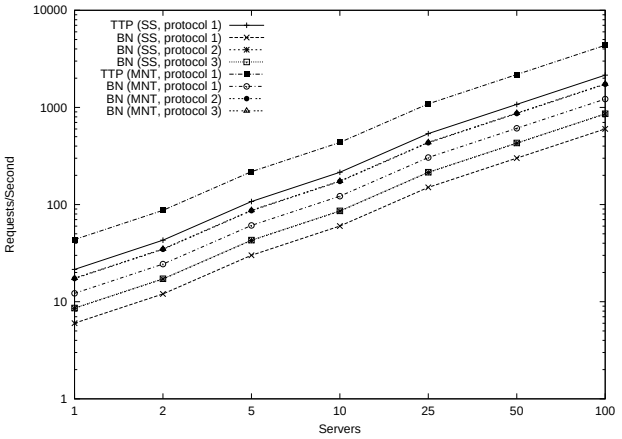

Fig. 4. Server scalability by base construction (request/second)

Note that the token value returned to the node in the first exchange of protocol 3 no longer has the quality that it can independently be validated by the node before being passed to the bootstrap node. This represents a small window for DoS attack where an adversary could corrupt the token being passed to the joining node. Such corruption would not be caught until it is given to the bootstrap node. It is unclear how much a problem this represents because the signed token could just as easily be corrupted on the path between the joining node and the bootstrap node.

The current implementation has a number of opportunities for optimization. For example, a number of additional protocol exchanges exist that simplify programming, but incur non-trivial overheads. Similarly, the implementation of the cryptographic functions analyzed in the protocol and preceding subsection can be made more efficient: the cryptographic materials (e.g., keys, book-keeping structures) are created for each protocol run at the server, which increases the cost of the operations significantly. We are actively exploring, improving, and evaluating the implementation.

There are also more efficient parameters for encryption under an ID-based cryptosystem. MNT elliptic curves, for example, are more than 102.7% faster than supersingular curves for encryption operations. Another promising optimization explored by Pirretti et al. [15] is the use of a random oracle construction [1], [4]. To *vastly* simplify, this approach allows us to replace complex cryptographic algorithm elements within the ID algorithms with a simple hash function. Such an approach is formally weaker than "standard" cryptographic models, but is often essential to making practical cryptosystems. As measured by Pirretti et al., this approach results in

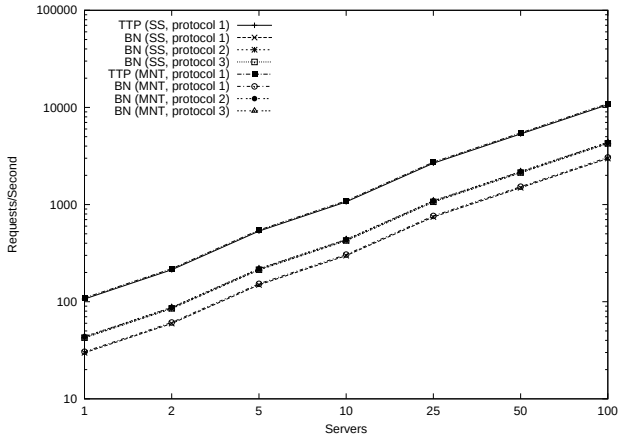

Fig. 5. Server scalability w/random oracle construction (request/second)

395.9% faster encryption for supersingular and 408.4% for MNT curves.

C. Scalability

One of the chief measures of the feasibility of this approach is its ability to scale to large numbers of users. Peer-to-peer systems often contain thousands or millions of concurrent users. Failure to support these huge workloads will severely limit the applicability of our approach.

In order to support scalability to very large peer-to-peer systems, we consider protocol cost under replicated operation. In this evaluation, we assume that all server functions can be replicated (as briefly discussed in the preceding section), and that such replication leads to linear or near-linear speedup (a reasonable assumption). Figure 4 shows the scaleup behavior for servers in all three protocols under both the SS and MNT elliptic curves. Note that systems containing one or a few servers can sustain a limited load (supporting a few to tens of requests per/second), but scale quickly to hundreds or thousands of requests per second in large installations. Further, the use of the MNT construction increases the number of requests supported in all cases by a factor of two. Note that the use of MNT curves is not without cost: the keys, signatures, and ciphertexts associated with MNT curve ID-cryptography are significantly larger that those in SS curves. However, as storage and bandwidth are plentiful in peer-to-peer systems, this may not represent a serious problem for this application.

Figure 5 shows the effect of random oracles on cost. The random oracle construction increases the supported workload almost five-fold for all protocols. This is a reflection of the results presented above, where cost is dominated by ID-based cryptographic operations. In the most efficient construction and protocol, a 100-server environment can handle over 3,000 requests per second. Note that the differences in cost between MNT and SS curves is less pronounced in the random-oracle model: the SS curves are only slightly slower than MNT (within a few percentage points). Hence, because of lower storage costs, SS curves may indeed be optimal in random oracle systems.

Instantaneous requests-per-second measurements do not tell the whole story of scalability. What one needs is a characterization of the sustainable size of the supported community. Put another way, how many users can these systems continually support? We formulate the size of the community based on the protocol and optimizations as follows: Assume that a base server exchange takes k microseconds under an SS curve (from above tables). Each construction has a optimization factor o that represents the protocol speedup factor (MNT=2.027, random oracle SS=4.959, and random oracle MNT=5.084). Further, assume an average occupancy of a user is s (in hours). Then, the supported community size $\mathcal{C}$ would be:

$$\mathcal{C} = \frac{10^7}{k * o} * (s * 60^2) \quad (1)$$

Applying this formula to real environments, assume that users have an average occupancy in the peer-to-peer system of 2 hours (a conservative estimate), and that the node joining/rejoining is uniformly distributed in time. In this case, a system of two servers in protocol 1 could support a user community of 439,000 users (two servers support 61 requests per second * 7,200 seconds). The same two servers could support 626,000 and 619,000 users in protocols 2 and 3. Larger systems can support larger communities: a system of only 50 servers could support over 11,000,000 users in protocol 1, and 15,800,000, and 15,600,000 users in protocols 2 and 3, respectively. Hence, these protocols scale to even the largest peer-to-peer networks by replicating server functions over a modest number of servers.

V. Discussion

ID-based cryptosystems have many advantages over certificate-based systems, such as obviating the need for a public key infrastructure and the resultingly vast simplification of key management. However, as discussed in this section, the operational requirements of ID-based cryptosystems present other challenges.

A. Key Escrow

One of the limitations of ID-based cryptography is an unavoidable presence of key escrow. This problem is particularly manifest in protocols 1 and 2 (which we describe in detail in sections III-A and III-B, respectively). In these scenarios, a dependence exists on the trusted private key generator (PKG), which has full knowledge of all private keys in the system and a master key that aids in their generation. However, the server represents a single point of failure in the system: if the PKG is compromised, all of the private keys can be exposed.

Several schemes have been proposed to limit the effect of server compromise in ID-based cryptosystems. One such scheme uses multiple authorities to store and use the master key [3], [6], where no single authority ever possesses enough

information to autonomously generate a private key. However, these solutions can add significant complexity to the system, e.g., complex failure modes, required additional protocol exchanges, etc.

B. Key Revocation

Certificate Revocation Lists (CRLs) are used in traditional certificate-based systems to determine whether a public key continues to be valid, i.e., has not been revoked. However, particularly where many certificates are issued or in highly dynamic environments, the overheads associated with maintaining CRLs can be prohibitive [14]. ID-based schemes do not need to manage CRLs or verify the validity of public keys through a certificate chain. It is, however, inherently difficult to support proper key revocation in the system when a node's public key is synonymous with its ID.

One particular situation where key revocation may be necessary is in networks where DHCP is used. With DHCP, a client on a local network is assigned an IP address from a pool. When that client leaves the network, the IP address they were assigned becomes available for reuse. An adversary can obtain an IP address through DHCP and register an ID with the P2P network, then release their IP address and obtain a new one, and gain a new ID with this address. In this manner, a limited variant of the Sybil attack may be possible.

Key expiry explicitly defines when a key is created and the period over which it should be deemed valid. Expiry can be incorporated in an ID-based system by including the current date or time as part of the public key, along with the node ID (i.e., appending a timestamp to the IP address) resulting in the following ID:

```
192.168.0.1-Monday-July-21st-8:00am-10:10am
```

This ID explicitly indicates the time over which the associated node can participate in the network. The key lifetimes limit the vulnerability of a compromised node to only a short window. Hence, because the damage of a compromised key is limited, revocation is unnecessary [17]. However, the validity period affects the security of the system; if the time period is too short, updating the corresponding private key may introduce unnecessary computation at the PKG. Conversely, longer time periods can result in more exposure to compromise. It is incumbent on the system to set system parameters to make this tradeoff between security and cost.

C. Denial of Service Attacks

P2P systems are vulnerable to *Denial of Service* (DoS) attacks in which an adversary causes resource exhaustion by executing many seemingly legitimate operations. The PKG in an ID-based cryptosystem may be attacked in this way by sending a flood of forged or spoofed requests, overwhelming it with false requests for private keys. As shown in section IV, this key generation is computationally expensive, and a flood of false requests may result in the PKG ceasing to meaningfully function.

To mitigate this attack, we defer private key generation until the initial phase of the weak authentication callback is complete, i.e., the key is generated after the three-way TCP handshake from PKG to the requesting node finishes. To wit, only when the authenticity of the requesting node is verified will a new private key be generated. Note that if an adversary controls a zombie network of tens of thousands of hosts, the P2P system will be susceptible to attack; the callback mechanism is a weak form of authentication. Possible solutions to these more sophisticated distributed DoS attacks include implementing load balancing [8] or computational puzzles [18]. Ultimately, server resources are finite and achieving resilience to thousands or millions of malicious hosts is, to say the least, challenging. Defending against these attacks is beyond the scope of this paper.

VI. Related Work

Douceur [10] identifies *Sybil attacks* as adversaries simultaneously obtaining many pseudo-identities in P2P systems. He shows that without a centralized certification authority, it is very difficult to prevent nodes from gaining many pseudo-identities, and asserts that requiring all nodes to obtain a certificate is too expensive to be practical. He suggests methods for imposing computational cost on creating an identity and system conditions to mitigate the attack. However, Douceur limits much of his discussion to the attack, and it is not clear how one would implement these approaches in P2P overlay networks.

In addition to a centralized authority, Castro et al. [5] suggest either charging money for certificates or binding node IDs to real-world identities in order to mitigate the Sybil attack. While this can ensure that node IDs are unique and, to some extent, moderate the rate at which node IDs can be obtained, it is often impractical to require that all nodes spend money or prove their real-world identity in P2P systems.

Srivatsa and Liu [22] espouse a variant of the traditional approach. In this, the bootstrap node assigns a random identifier and issues an associated certificate with a short lifetime. This can guarantee unique node ID assignment and also control the number of node IDs that are generated in the system. However, it can be cumbersome for all nodes to obtain and update a certificate.

A variety of cryptographic puzzle mechanisms have been proposed to address Sybil attacks. Castro et al. [5] describe one method for node ID generation by requiring new nodes to generate a unique key pair such that the hash of the public key has the first p zero bits. Rowaihy et al. [18] present an admission control system using a hierarchy of participating peers and a chain of puzzles. Its effectiveness depends on the cost and the degree of hardness of solving puzzles. However, it is limited by a complex structure and requires a potentially large number of exchanges with varying servers to obtain a single ID.

VII. Conclusion

In this paper we have considered the use of identity-based cryptography to assist in the security and performance critical

assignment of user identities in peer-to-peer systems. Identity-based cryptosystems use textual strings to derive public keys from cryptographic parameters advertised within a domain. This approach avoids many of the complexities of PKI usage (a user's public key is directly derivable from their identity), and reduces the overheads associated with authentication. We exploit these advantages in peer-to-peer systems by assigning an ID and providing the associated identity-based private key ID to each joining node. Nodes are loosely authenticated via callback: any node capable of *receiving* an in-bound TCP connection for an IP address is deemed authentic.

We developed three protocols representing diverse trust models and performance profiles based on identity-based cryptography: a fully decentralized ID-based assignment scheme (protocol 1), a centralized scheme in which a single host plays the role of both ID authority and bootstrap node (protocol 2), and an approach that retains the separation of duties in a decentralized model at a low cost by using a hybrid of identity-based and symmetric key cryptography (protocol 3). Our evaluation of the performance of these protocols shows that their costs vary widely by model and type of cryptography used. We further show that systems using these protocols can scale to massive P2P networks through the proper use of cryptography and server replication.

Peer-to-peer systems often face conflicting requirements for autonomy, robustness, and security. These systems fill an important niche by providing highly-available, massively-distributed storage. However, their continued growth is dependent on the technical community's ability to introduce further infrastructure to secure the media. This work and others like it will solve the challenges of this media by exploiting emerging technologies such as identity-based cryptography.

REFERENCES

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, pages 62–73, 1993.

[2] D. Boneh, X. Boyen, and E. J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of Eurocrypt 2005*, volume LNCS 3494, pages 440–456. Springer-Verlag, 2005.

[3] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.

[4] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited (preliminary version). In *STOC*, pages 209–218, 1998.

[5] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proceedings of OSDI 2002*, Boston, MA, Dec. 2002.

[6] L. Chen, K. Harrison, N. Smart, and D. Soldera. Applications of multiple trust authorities in pairing based cryptosystems. In *Proccedings of the Infrastructure Security Conference 2002*, volume LNCS 2437, pages 260–275, 2002.

[7] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[8] N. Daswani and H. Garcia-Molina. Query-flood DoS attacks in Gnutella. In *Proceedings of ACM CCS'02*, pages 181–192, Washington, DC, 2002.

[9] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[10] J. Douceur. The Sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.

[11] C. Gentry and A. Silberberg. Hierarchical ID-based cryptography. In *Proceedings of Asiacrypt 2002, LNCS 2501*, pages 548–566. Springer-Verlag, 2002.

[12] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In *Proceedings of Asiacrypt 2002, LNCS 2501*, pages 466–481. Springer-Verlag, 2002.

[13] B. Lynn. PBC library. http://rooster.stanford.edu/˜ben/pbc/, 2006.

[14] P. McDaniel and A. Rubin. A Response to 'Can We Eliminate Certificate Revocation Lists?'. In *Proceedings of Financial Cryptography 2000*. International Financial Cryptography Association (IFCA), February 2000. Anguilla, British West Indies.

[15] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-Based Systems. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA, November 2006.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, , and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001*, pages 161–172, San Diego, CA, 2001.

[17] R. L. Rivest and B. Lampson. SDSI – A simple distributed security infrastructure. Presented at CRYPTO'96 Rump session, 1996.

[18] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta. Limiting Sybil attacks in structured peer-to-peer networks. Technical Report NAS-TR-0017-2005, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, July 2005.

[19] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of Middleware*, pages 329–350, Heidelberg, Germany, 2001.

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

[21] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *Proceedings of ACM SIGOPS European Workshop*, Leuven, Belgium, 2004.

[22] M. Srivatsa and L. Liu. Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In *Proceedings of ACSAC 2004*, pages 252–261, Cambridge, MA, 2004.

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, San Diego, CA, 2001.

[24] D. S. Wallach. A survey of peer-to-peer security issues. In *Proceedings of ISSS*, pages 42–57, Tokyo, Japan, 2002.

[25] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.