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

Kevin R.B. Butler, *Student Member*, *IEEE*, Sunam Ryu, Patrick Traynor, *Member*, *IEEE*, and Patrick D. McDaniel, *Senior Member*, *IEEE*

**Abstract**—Structured peer-to-peer (P2P) 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—ultimately leading 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.

**Index Terms**—Network protocols, peer-to-peer, distributed systems, cryptographic controls.

## 1 Introduction

Peer-to-peer (P2P) 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 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 P2P system. Malicious entities that can control ID assignment can probabilistically or deterministically assert themselves as the source of selected content or routing messages, and can therefore subvert the routing protocols, pollute the delivered content, or prevent placement of legitimate content. Consider, for example, an entity attempting to deny service to a movie download service in an attempt to extort the distributors; such attacks are already common against other distribution services [1]. Further *Sybil* attacks allow an adversary to control large portions of the P2P network by simultaneously obtaining many identities [2]. 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 [3]. These schemes, however, require maintenance of complex PKI systems, which can be difficult or infeasible to implement in practice [4].

In this paper, we consider the use of identity-based cryptography (specifically identity-based encryption, or IBE) to assist in the security and performance critical assignment of user identities in P2P 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, as a user's public key is directly derivable from their identity, and reduces overheads associated with authentication. We exploit these features in P2P 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 to limit damaging Sybil attacks.

The use of IBE systems leads to a trust model different than those offered by previous centralized identity management approaches. In our proposed system, identities and keys are derived directly from the IP addresses of the participating entities—thus, there is no key-to-identity binding ambiguity for a trusted certification authority to resolve via signature. Users of the system compute the public keys of their peers directly. We explore these costs in a comparison with alternate designs in Section 4.3.

- *K.R.B. Butler and P.D. McDaniel are with the Department of Computer Science and Engineering, Pennsylvania State University, IST Building, University Park, PA 16802. E-mail: {butler, mcdaniel}@cse.psu.edu.*
- *S. Ryu is with the Defense Security Command, Juam-dong, Gwacheon-si, Gyeonggi-do, South Korea. E-mail: sunamryu@gmail.com.*
- *P. Traynor is with the School of Computer Science, Georgia Institute of Technology, Kluus Advanced Computing Building, Room 3138, 266 Ferst Drive, Atlanta, GA 30332-0765. E-mail: traynor@cc.gatech.edu.*

*Manuscript received 23 Feb. 2008; revised 1 July 2008; accepted 7 Nov. 2008; published online 1 Dec. 2008.*
*Recommended for acceptance by C. Shahabi.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-02-0074.*
*Digital Object Identifier no. 10.1109/TPDS.2008.249.*

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 ms in protocol 1 versus 115 ms in protocol 2. We also found that we could achieve protocol runtimes 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 ms). 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, rendering these solutions practical for current P2P systems and overlay networks such as Skype [5].

Any solution that limits the scalability of a P2P system is unlikely to be widely adopted. Our analysis found that our protocols could scale easily, where five servers could conservatively sustain a community of over 113,000 nodes, and 50 ID servers could support over 1,130,000 nodes.

In summary, we show that

- IBE solutions to prevent Sybil attacks are practical and scalable within current P2P systems,
- tradeoffs can be made to calibrate performance versus the amount of trust placed into the system, and
- using IBE in P2P systems presents some alternative challenges to a deployment based on a public-key infrastructure, but eliminates the complexities of certificate management inherent to PKI.

The rest of this paper is organized as follows: Section 2 gives a brief overview of structured P2P networks and identity-based cryptography, and identifies the broad goals and assumptions of this work. Section 3 describes our novel protocols in detail. Section 4 describes an empirical evaluation of the proposed approach. Open problems and operational issues are discussed in Section 5. Section 6 discusses important related work, and Section 7 concludes.

## 2 Background

This section presents relevant background in P2P systems and identity-based cryptography, and describes the security and performance goals of our approach.

### 2.1 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 P2P 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 and 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 [6], Pastry [7], and Tapestry [8], node IDs are deterministically assigned by hashing the host's IP address. Conversely, in CAN [9], 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 [2]. These identities probabilistically interpose the adversary in the routing paths for great many objects, thus permitting him to disrupt or manipulate the search process [10], [11], [12].

### 2.2 Identity-Based Cryptography

Public keys in identity-based public key cryptosystems are simple data objects [13], [14], [15], e.g., ASCII string email addresses. Associated with ID cryptosystems is a set of well-known public parameters for generating the cryptographic material used for decryption or signature verification. A trusted third party (TTP), 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 dissemination of public parameters and the public key "strings." This is useful where the deployment of a traditional certificate authority-based PKI is inconvenient or infeasible, as IBE-based systems do not require certificate management, eliminating the need for certificate lookups and complex certificate revocation schemes.

A central operational consideration of ID-based cryptography is that 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 Section 4). To ease the computation burdens of PKG operation, hierarchical IBE (HIBE) [16], [17], [18] can be used to reduce the overload of a root PKG by replicating private key generation to slave PKGs.

### 2.3 Protocol Setup

This work is focused on the secure assignment and authentication of pseudo-identities in P2P 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

[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 $b$ |

Fig. 1. Notation used throughout this paper.

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 source IP address can be spoofed, and any packet can be eavesdropped by an attacker. We explicitly do not assume that an adversary can hijack arbitrary IP addresses (e.g., force the network to route packets to himself, rather than to the host assigned that address). The network and all servers and participants in the system may crash without notification. The servers in the system are assumed not to be compromised and remain faithful to the operation of the protocol throughout (we defer an analysis of server compromise to future work). We make no assumptions about the organization or ownership of the system servers. Experience suggests that communities of users who see value in the service will provide functionality, typically through donations or via subscription service.

We place no restrictions on the number of compromised or adversarial client nodes in the system. An individual adversary is assumed to have limited access to available IP addresses and computational resources.[2]

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).

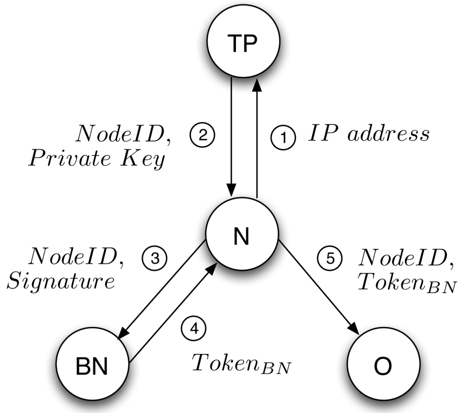


Fig. 2. Node join in TTP protocol (protocol 1).

Each such communication is protected by a secure channel established via a Diffie-Hellman key exchange [19]. Unauthenticated Diffie-Hellman key exchanges are subject to man-in-the-middle attacks. If deemed a concern, additional mechanisms such as server-side authenticated Diffie-Hellman can be used to combat this problem (as used in SSL/TLS [20]). This would incur additional overheads, and for brevity, we leave the investigation of authenticated Diffie-Hellman to future work. We further assume the existence of some loosely synchronized secure clock.

Fig. 1 describes the notation used throughout this paper.

## 3 Protocol Specification

In the following sections, we present three protocols that authenticate node IDs and protect structured P2P 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.

### 3.1 Protocol 1: Trusted Third Party

In the first protocol, the binding between a node's ID and its private key is performed by a TTP, as shown in Fig. 2. This serves a similar function to a centralized authority in the traditional PKI. Here, the TTP[3] 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.

#### 3.1.1 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 [15].

#### 3.1.2 Node Join

In order for a node $N$ to join an overlay network, it first contacts the $TTP$ and provides its IP address. After weakly

[2] In practice, adversaries may have access to many hosts or IP addresses, e.g., via botnets. In all of the protocols defined throughout, the number of malicious P2P client identities available to this adversary would be linearly bounded by the number the IP addresses that the adversary can force the network to route to them.

[3] 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.

authenticating its identity via callback, $TTP$ gives the node a randomly generated ID and the corresponding private key. Note that that one-time key $K_{TP,N}$ used to encrypt and integrity protect the ID and private key passed to the node is negotiated by the pair by performing a Diffie-Hellman exchange as part of the callback process. After the exchange is received, $N$ contacts the bootstrap node $BN$[4] and provides $BN$ with its ID and a timestamp, both signed with $N$'s private key.

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. $N \rightarrow BN : ID_N$, $TS_1$, $Sign(ID_N \| TS_1, K_N^-)$.
4. $BN \rightarrow N : Sign(ID_N \| TS_1, K_{BN}^-)$.
5. $N \rightarrow O : ID_N$, $TS_1$, $Sign(ID_N \| TS_1, K_{BN}^-)$.

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 $TTP$ 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. Because $BN$ has signed the response, it can be used as a token of authenticity. $O$ can verify the signature, as $BN$'s identity is 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 PKI, 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.

## 3.2 Protocol 2: Trusted Bootstrap Node

In contrast with the previous scheme, the Trusted Bootstrap Node protocol shown in Fig. 3 implements a centralized system. Specifically, instead of relying upon a TTP to perform the duties of key distribution, the bootstrap node becomes the arbiter of network membership and trusted information. This approach thus attempts to minimize the overhead and complications associated with a decentralized architecture.

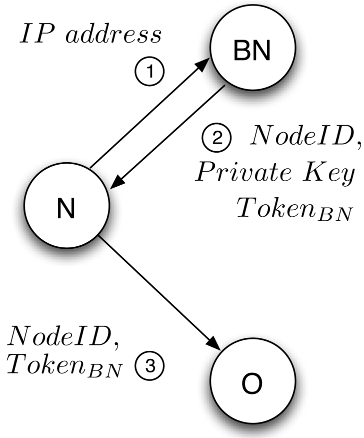


Fig. 3. Node join in Trusted Bootstrap Node protocol (protocol 2).

### 3.2.1 Setup

In a similar fashion to a TTP, 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.

### 3.2.2 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:

1. $N \rightarrow BN : IP_N$.
2. $BN \rightarrow N : ID_N$, $E(K_N^-, K_{BN\cdot N})$, $TS_1$, $Sign(ID_N \| TS_1, K_{BN}^-)$.
3. $N \rightarrow O : ID_N$, $TS_1$, $Sign(ID_N \| TS_1, K_{BN}^-)$.

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, $N$ 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

[4] 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.

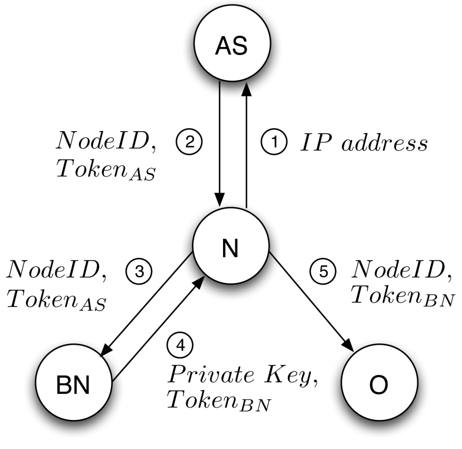


Fig. 4. Node join in Trusted Assignor Node protocol (protocol 3).

key expiration. An exploration of the security, performance, and functional tradeoffs of this scheme is given in Section 5.

### 3.3 Protocol 3: Trusted Assignor Node

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 Trusted Assignor Node 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.

#### *3.3.1 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.

#### *3.3.2 Node Join*

When $N$ attempts to join the network, as shown in Fig. 4, it transmits its IP address to a trusted assignor node $AS$. After verifying the identity, $AS$ generates the node ID and issues a time-stamped 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:

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

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 $BN$. We further discuss the benefits and security, performance, and functional tradeoffs of this scheme in Section 5.

## 4 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 of all three protocols in C. We use the GNU GMP library for all standard cryptographic algorithms, 128-bit AES for non-IBE, and SHA-1 for hashing. All identity-based cryptographic algorithms use the pairing-based cryptography (PBC) library [21]. We parameterized the library to use supersingular elliptic curves over a nonrandom oracle construction [22] and Cha-Cheon signatures [23]. Experiments were performed on two Dell Optiplex 745 workstations, each with dual core 1.86-GHz Intel Core Duo 2 processors and 1-Gbyte RAM. The hosts ran the Ubuntu 7.04 distribution using version 2.6.20 of the Linux kernel and were connected via a Gigabit Ethernet switch. All reported results represent an 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).

We present metrics for "old" and "new" implementations. The PBC library upon which our implementation of identity-based cryptography is based has been under active development over the last several years [24]. These developments have seen increases in performance of most operations, as well as bug fixes and code restructuring. The following tests were performed using both version 0.2.16 (old, circa July 2006) and version 0.4.11 (new, circa July 2007) of the PBC library, and serve as a commentary on the effectiveness of the library improvements. Modulo changes to accommodate new library APIs, the application code in both old and new experiments is identical. Our experimental results centrally present the following findings:

- Microbenchmarks of cryptographic operations show that the cost of IBE operations can be significantly reduced as improvements to the implementation libraries continue.
- Computational costs of each of our protocols improve due to underlying library improvements, and provide a tradeoff between performance and trust in the system entities.
- IBE systems are competitive with PKI-based systems, which share many of the same computational costs.
- Our approach is scalable to very large P2P systems and support server scale-up, such that a system using only 50 servers can support over 1 million users with any of our protocols.

We begin our evaluation with a discussion of cryptographic microbenchmarking.

### 4.1 Cryptographic Microbenchmarks

There are five significant cryptographic operations used by the protocols defined in this paper: the creation of the

TABLE 1
Cryptographic Microbenchmarks (in Milliseconds)

| | Old | | New | | |
|---|---|---|---|---|---|
| | Cost | $\sigma$ | Cost | $\sigma$ | % Improv |
| Key Creation | 31.3836 | 0.2700 | 28.5704 | 0.1920 | 8.963 |
| Node Signature | 48.2608 | 0.9239 | 18.4583 | 0.2125 | 61.753 |
| Request Verification | 62.0760 | 1.7160 | 49.9466 | 1.7160 | 19.539 |
| ID Token Creation | 14.4557 | 0.1461 | 15.8517 | 0.1224 | -9.657 |
| Symmetric Key Token Creation | 0.0415 | 0.0048 | 0.0415 | 0.0046 | 0.1035 |

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 that 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 1.

The key creation, 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). Note the significant performance gains due to improvements to the PBC library, which reduce signing costs by as much as 50 percent. Slightly increased token creation costs are also observable. These differences are due both to various pairing and other improvements, and changes in the code to make certain operations more robust, e.g., better memory management, other bug fixes (see PBC library README files [21]). Our main observation from these benchmarking experiments is that *significant room for improvement in the underlying implementation means that IBE primitives are capable of even greater performance.*

### 4.2 Protocol Benchmarks

We now break down the per-flow and total costs for each of the protocols. Table 2 presents the results for the four messages composing the TTP protocol (protocol 1). The first two messages implement an exchange between the node and TTP to obtain the private key and node ID. Messages 3 and 4 are used to authenticate the joining node to the bootstrap node, and to obtain the token used to prove ownership of the ID to other P2P members (see Fig. 2 and associated text in Section 3.1 for further detail). Note that for each exchange with a foreign server, the client performs a Diffie-Hellman key exchange. For simplicity, we report the costs associated with this exchange within the first message exchanged with the server.

There are several aspects of the performance analysis of protocol 1 that are notable. First, the exchange between the node and the TTP is relatively fast. As noted in the previous section, an ID-based key takes about 31 (or 28) ms to create. This accounts for approximately 40 percent of the time required for this exchange, with the remaining 60 percent attributed to the DH exchange, network delay, and software initialization. The third message (first message of the bootstrap node exchange) consumes about 40 percent 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 library improvements result in about a 7 percent protocol performance improvement.

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 3. Note that the average execution time is *65 percent 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. As is true of

TABLE 2
TTP Protocol Performance (in Milliseconds)

| | One | Two | Three | Four | % Improve |
|---|---|---|---|---|---|
| Client (old step) | 0.0323 | 237.9227 | 231.1510 | 122.1773 | |
| Client (old cumulative) | 0.0323 | 237.9550 | 469.1060 | 591.2833 | |
| Client (new step) | 0.0384 | 234.6281 | 189.4667 | 121.3408 | |
| Client (new cumulative) | 0.0384 | 234.6664 | 424.1331 | 545.4740 | 7.7474 |
| Server (old step) | 80.4402 | 117.4226 | 271.3293 | 122.1762 | |
| Server (old cumulative) | 80.4424 | 197.8628 | 469.1921 | 591.3683 | |
| Server (new step) | 80.4315 | 114.1990 | 229.5874 | 121.3401 | |
| Server (new cumulative) | 80.4336 | 194.6305 | 424.2179 | 545.5580 | 7.7465 |

TABLE 3
Trusted Bootstrap Node Protocol Performance

| | One | Two | % Improve |
|---|---|---|---|
| Client (old step) | 0.0311 | 353.2157 | |
| Client (old cumulative) | 0.0311 | 353.2469 | |
| Client (new step) | 0.0403 | 357.5951 | |
| Client (new cumulative) | 0.0403 | 357.6354 | -1.2423 |
| Server (old step) | 39.9581 | 313.3774 | |
| Server (old cumulative) | 39.9581 | 353.3355 | |
| Server (new step) | 40.1817 | 317.5576 | |
| Server (new cumulative) | 40.1817 | 357.7393 | -1.2463 |

protocol 3 (see below), the PBC improvements have little effect on the experiment—the substantially faster cryptographic operations do not occur within this protocol.

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). Shown in Table 4, 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 TTP exchange in protocol 1 at a fraction of 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. The difference between the reported improvements of protocol 3 over protocol 1 is due almost entirely to this enhancement.

Note that the token value returned to the node in the first exchange of protocol 3 no longer has the quality that it can be independently validated by the node before being passed to the bootstrap node. This represents a small window for a denial-of-service (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 of a problem this represents, as 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 nontrivial overheads. Similarly, the implementation of the cryptographic functions analyzed in the protocol and preceding section can be made more efficient: the cryptographic materials (e.g., keys, bookkeeping 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 percent faster than supersingular curves for encryption operations. Another promising optimization explored by Pirretti et al. [22] is the use of a random oracle construction [25], [26]. 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 395.9 percent faster encryption for supersingular and 408.4 percent for MNT curves.

ne may be tempted to conclude based on these experiments that the PBC library improvements provide little, if any, advantage. This is an incorrect conclusion. The advantages of the optimization lie centrally in the improvements to the IBE signature generation and verification—operations not occurring in protocols 2 and 3. Of course, these operations will occur frequently as the user proves its identity to others in the P2P system. Hence, such performance improvements may observably increase the performance of the entire P2P

TABLE 4
Multiple Node ID Assignors Protocol Performance

| | One | Two | Three | Four | % Improv |
|---|---|---|---|---|---|
| Client (old step) | 0.0338 | 78.5879 | 0.0073 | 275.1289 | |
| Client (old cumulative) | 0.0338 | 78.6217 | 78.6290 | 432.4037 | |
| Client (new step) | 0.0380 | 80.3612 | 0.0074 | 277.5529 | |
| Client (new cumulative) | 0.0380 | 80.3992 | 80.4066 | 438.3940 | -1.385 |
| Server (old step) | 40.1262 | 0.0718 | 78.6487 | 313.6445 | |
| Server (old cumulative) | 40.1262 | 40.1980 | 118.8468 | 432.4912 | |
| Server (new step) | 40.1452 | 0.0710 | 80.4924 | 317.7765 | |
| Server (new cumulative) | 40.1452 | 40.2161 | 120.7085 | 438.4850 | -1.385 |

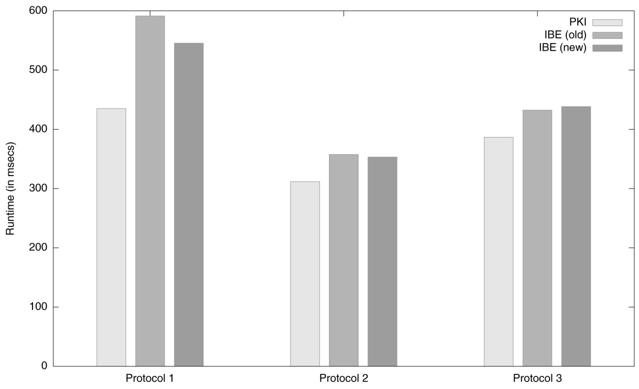


Fig. 5. Comparison of IBE with public key node management.

network, even though those improvements are not always observable in node ID acquisition.

### 4.3 IBE Overheads

A central cost of the system relates to the use of identity-based cryptography. In an effort to characterize the effect of this cost on the system, we study an alternate design that factors out the overheads associated with IBE. Fig. 5 shows a histogram of the runtime comparison of the proposed IBE system with a theoretical "PKI" system whose cryptographic costs are marginal,[5] as might be the case with a server implementing any number of a node management scheme using public key cryptography, e.g., [27].

Fig. 5 shows a cost comparison of the IBE and PKI solutions. In this analysis, we find a modest speedup in protocol 1 when using a PKI approach, where the runtime decreases by 20 percent to 27 percent as compared to the new and old IBE implementations, respectively. The PKI also shows a lesser improvement of 10 percent to 12 percent for protocols 2 and 3. This stands to reason, as PKI solutions pay many of the same costs as IBE-based systems, such as key generation and the costs of signing and validation. In addition, most of the computation overheads associated with the IBE system are dominated by network and operating system costs. Such performance advantages will also diminish over time, as IBE and its implementations mature.

### 4.4 Scalability

One of the chief measures of the feasibility of this approach is its ability to scale to large numbers of users. P2P 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 P2P 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). Fig. 6 shows the scale-up 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. Furthermore, the use of the MNT construction increases the number of requests supported in all cases by a factor of 2. 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 P2P systems, this may not represent a serious problem for this application.

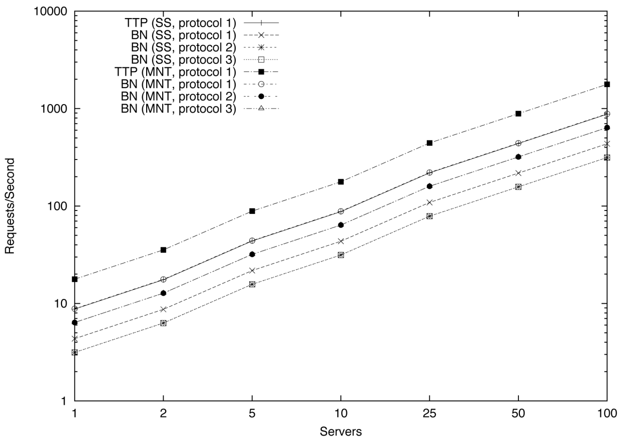


Fig. 6. Server scalability by base construction (request/second), new library.

Fig. 7 shows the effect of random oracles on cost. The random oracle construction increases the supported workload almost fivefold 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 2,000 requests per second. Note that the differences in cost between MNT and SS curves are less pronounced in the random-oracle model: the SS curves are only slightly slower than MNT (within a few percentage

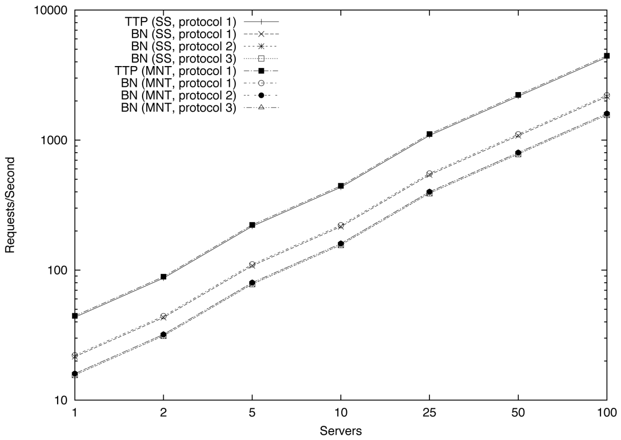


Fig. 7. Server scalability with random oracle construction (request/second).

[5] For the purposes of this analysis, we assume that the cryptographic costs are zero. In practice, the costs would be nonzero, and our analysis is therefore conservative in favor of the PKI approach.

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 ($\mathrm{MNT} = 2.027$, random oracle $\mathrm{SS} = 4.959$, and random oracle $\mathrm{MNT} = 5.084$). Further, assume an average occupancy of a user is $s$ (in hours). Then, the supported community size $\mathcal{C}$ would be

$$\mathcal{C} = \frac{10^6 * o}{k} * (s * 60^2). \qquad (1)$$

Applying this formula to real environments, assume that users have an average occupancy in the P2P system of 2 hours (based on empirical measurements [28]), and that the node joining/rejoining is uniformly distributed in time. In this case, a system of five servers in protocol 1 could support a user community of 62,690 users (five servers support ≈15 requests per second ∗ 7,200 seconds). The same five servers could support 113,000 users in both protocols 2 and 3. Larger systems can support larger communities: a system of only 50 servers could support over 1,040,000 users in protocol 1, and 1,130,000 users in protocols 2 and 3. Hence, these protocols scale to even the largest P2P networks by replicating server functions over a modest number of servers. Note that while the protocol runtime performance comparison above shows modest improvements when using a PKI approach over IBE, PKI solutions are, at least for now, likely to scale much better because of the reduced computational overheads.

Recent empirical analyses of structured P2P networks show increasing variance in the session hold times [29], [28]. These studies show client session times that range from about 1 to 2.4 hours, depending on the network and the experimental data collected. Thus, the above analysis may overestimate the scalability of the current system by a factor of 2. However, advances in algorithm efficiency and hardware accelerators for PBC [30] are likely to vastly increase the scalability of servers—thus, largely mitigating the effect of higher churn rates on server throughput.

## 5 Discussion

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

### 5.1 Runtime Costs

The computational overheads of IBE extend beyond the admission control process. Each control message needs to be signed by the sender and verified by the recipient. While over time, Moore's law and improvements in the underlying cryptographic algorithms may reduce the burden on the clients, the increased demands on the clients may present additional delays. To illustrate, consider the computational costs associated with a naive IBE-based content search (using the benchmarks in Section 4.1). The system would incur a 70-ms delay for each message-approximately 20 ms for signing and 50 ms for validation. Such overheads may lead to delays of upward of a second to find a single piece of content, e.g., assuming a DHT with $10^7$ nodes, the search overhead would be $\log(10^7) = 10 * 70 = 700$ ms.

There are several ways to mitigate these costs. First, a responder could cache common responses that it already has signed, and return those in the response to relevant peer requests (thus, amortizing the costly repetitive signing process). Second, peers in the network could simply negotiate pairwise shared symmetric keys in an initial exchange (by encrypting the one-time pair key with the responder's public key). All subsequent communication could be inexpensively authenticated via the negotiated shared key. Note that these strategies rely on content or node reference locality. Locality has been observed in P2P networks [31], but further analysis is required to understand the degree to which these strategies will be successful in mitigating IBE costs.

### 5.2 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 3.1 and 3.2, respectively). In these scenarios, a dependence exists on the trusted 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 [13], [32], where no single authority ever possesses enough information to autonomously generate a private key. However, these solutions can add significant complexity to the system, such as complex failure modes, required additional protocol exchanges, etc.

### 5.3 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 [33]. 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, gaining 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:

$$\texttt{192.168.0.1-Monday-July-21st-8}:\texttt{00 a.m.-10}:\texttt{10 a.m.}.$$

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 [34]. 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. Note that these costs are due to allowing expiration, a feature not found in current P2P systems. The costs associated with expiration will also be found in certificate-based systems where the certificates expire. These are necessary costs as they allow revocation, which would be extremely difficult otherwise.

### 5.4 Denial of Service Attacks

P2P systems are vulnerable to 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 4, 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. Note, however, that such networks are explicitly outside of the Sybil attack as multiple machines are actually under the control of a single administrator. Possible solutions to these more sophisticated distributed DoS attacks include implementing load balancing [35] or computational puzzles [36]. 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.

### 5.5 Anonymity

While the use of IBE would seem to suggest otherwise, the concept of identity in this work is one of loose association. When a node presents its token to $O$, the NodeID inside can carry any range of semantic values. For systems in which a user's identity is not sensitive (e.g., a torrent system offering Linux distributions), $ID_N$ in the final message of the protocol can simply be the IP address of the node. In other networks, where anonymity is necessary, the value of $ID_N$ returned by $BN$ can be arbitrary (e.g., the result of an HMAC). Because $ID_N$ in the final token is not tied directly to an IP address, a node can use anonymized routing techniques to further protect their identity [37]. The protocols discussed in the previous sections will all complete correctly regardless of the $ID_N$ used in the final message as long as the signature from $BN$ can be verified.

Defending against Sybil attacks requires some form of identification to be present in a network [2]. Accordingly, if a network and its user value complete anonymity over the ability to prevent nodes from claiming multiple identities, no defense against the Sybil attack can be effective. Such a tradeoff is acceptable given the expectations of a network. For instance, very small and private P2P networks may not judge Sybil attacks to be a potential threat. Larger and public networks, however, will likely need to protect against malicious users.

### 5.6 Address Translation

Operating multiple computers behind a single IP address through the use of network address translation (NAT) is problematic for many P2P systems. The difficulties are exacerbated when the IP address serves as an identifier as well for the node. The use of NAT is discouraged due to its violation of the end-to-end principle in network design [38], and may be rendered obsolete with the impending widespread deployment of IPv6. Some legacy networks, however, may continue using NAT. IBE can handle the issues associated with NAT in different ways, depending on how NAT is implemented.

In the case of basic NAT where strictly address translation occurs, each host will be associated with a corresponding nonprivate IP address that may be registered with other hosts or the TTP; that is, the external IP address will correspond with $IP_N$ and $ID_N$ will reflect this address. In the case of NAT based on port reassignment, the problem becomes more complex. In a P2P environment, hosts will both act as senders and recipients of data; however, with port-based address translation, there is little facility for an machine outside the NAT perimeter to specifically contact a host inside a NAT without a priori knowledge of the port associated with the router in question. In this case, the machine inside the NAT will need to register both the external IP address and the port number it is reachable on with the registration node. Because these port numbers will be different for each host connecting behind the NAT, $ID_N$ will be unique. The NAT server will need to support traversal in order to facilitate handling of connections originating externally; also, NAT clients supporting STUN [39] will be able to easily determine their external IP address and port number, facilitating registration. These connections should be long lasting and static because of the requirements of associating identities with those particular nodes; more dynamic environments will require additional information such as time within its ID, as discussed in Section 5.3.

A limitation of this approach is that it introduces yet another vector for Sybil attacks: a single user can use port numbers as virtual IP addresses to generate tens of thousands of identities—in effect subverting the very mechanism the system is intended to address. To address this limitation, a TTP/BN/AS can "rate-limit" the number of identities that can be given to a specific IP for any number of ports. The authority would cache the identities provided to each IP address, and eject them from the cache when their validity period expires. All requests for additional identifiers are rejected after a threshold of cached IDs is reached.

## 6 Related Work

Douceur [2] 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 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. Early approach began by building reputation system upon which P2P participants could make decisions about whether to interact with suspicious or unknown clients [40].

In addition to a centralized authority, Castro et al. [10] 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 [27] espouse a variant of the traditional approach. Here, 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. [10] 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. Based on original work by Borisov [41], Rowaihy et al. [36] 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.

More recently, several systems have begun to fight the effect of Sybil attacks by understanding the social context in which they occur [42], [43], [44]. These systems employ decentralized approach that exploits the human-scale trust relationships to probabilistically limit the number of Sybil hosts acting as peers. However, because these systems are aimed at establishing peering relationships in an unstructured environment, they are inappropriate for the kinds of networks our system is designed to protect.

## 7 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 P2P 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 P2P 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 inbound 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.

P2P 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 address these challenges by exploiting emerging technologies such as identity-based cryptography.

## References

[1] D. Pappalardo and E. Messmer, *Extortion via DDoS on the Rise,* http://www.networkworld.com/news/2005/051605-ddos-extortion.html, May 2005.

[2] J. Douceur, "The Sybil Attack," *Proc. First Int'l Workshop Peer-to-Peer Systems (IPTPS '02),* Mar. 2002.

[3] B. Levine, C. Shields, and N. Margolin, *A Survey of Solutions to the Sybil Attack.* Univ. of Massachusetts Amherst, 2006.

[4] C. Ellison and B. Schneier, "Ten Risks of PKI: What You're Not Being Told about Public Key Infrastructure," *Computer Security J.,* vol. 16, no. 1, 2000.

[5] S.A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," *Proc. IEEE INFOCOM '06,* Apr. 2006.

[6] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM '01*, pp. 149-160, 2001.

[7] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. 18th IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01)*, pp. 329-350, 2001.

[8] B.Y. Zhao et al., "Tapestry: A Resilient Global-Scale Overlay for Service Deployment," *IEEE J. Selected Areas in Comm.*, vol. 22, no. 1, pp. 41-53, 2004.

[9] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM '01*, pp. 161-172, 2001.

[10] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure Routing for Structured Peer-to-Peer Overlay Networks," *Proc. Symp. Operating Systems Design and Implementation (OSDI '02)*, Dec. 2002.

[11] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against Eclipse Attacks on Overlay Networks," *Proc. ACM SIGOPS European Workshop*, 2004.

[12] D.S. Wallach, "A Survey of Peer-to-Peer Security Issues," *Proc. Second Int'l Symp. Steel Structures (ISSS '02)*, pp. 42-57, 2002.

[13] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. 21st Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '01)*, pp. 213-229, 2001.

[14] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues," *Proc. Eighth IMA Int'l Conf. Cryptography and Coding*, pp. 360-363, 2001.

[15] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Proc. Advances in Cryptology (CRYPTO '84)*, pp. 47-53, 1984.

[16] D. Boneh, X. Boyen, and E.J. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," *Proc. Advances in Cryptology (Eurocrypt '05)*, pp. 440-456, 2005.

[17] C. Gentry and A. Silberberg, "Hierarchical ID-Based Cryptography," *Proc. Eighth Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (Asiacrypt '02)*, pp. 548-566, 2002.

[18] J. Horwitz and B. Lynn, "Towards Hierarchical Identity-Based Encryption," *Proc. Eighth Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (Asiacrypt '02)*, pp. 466-481, 2002.

[19] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. IT-22, no. 6, pp. 644-654, Nov. 1976.

[20] T. Dierks and C. Allen, *The TLS Protocol Version 1.0*, RFC 2246, Jan. 1999.

[21] B. Lynn, *PBC Library*, http://rooster.stanford.edu/~ben/pbc/, 2007.

[22] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, "Secure Attribute-Based Systems," *Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06)*, Nov. 2006.

[23] J.C. Cha and J.H. Cheon, "An Identity-Based Signature from Gap Diffie-Hellman Groups," *Proc. Sixth Int'l Workshop Theory and Practice in Public Key Cryptography (PKC '03)*, Jan. 2003.

[24] S. Ryu, K. Butler, P. Traynor, and P. McDaniel, "Leveraging Identity-Based Cryptography for Node ID Assignment in Structured P2P Systems," *Proc. IEEE Int'l Symp. Security in Networks and Distributed Systems (SSNDS)*, 2007.

[25] M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," *Proc. First ACM Conf. Computer and Comm. Security (CCS '93)*, pp. 62-73, 1993.

[26] R. Canetti, O. Goldreich, and S. Halevi, "The Random Oracle Methodology, Revisited," (Preliminary Version), *Proc. ACM Symp. Theory of Computing (STOC '98)*, pp. 209-218, 1998.

[27] M. Srivatsa and L. Liu, "Vulnerabilities and Security Threats in Structured Overlay Networks: A Quantitative Analysis," *Proc. Ann. Computer Security Applications Conf. (ACSAC)*, 2004.

[28] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," *Proc. USENIX Ann. Technical Conf.*, June 2004.

[29] D. Stutzbach and R. Rejaie, "Understanding Churn in Peer-to-Peer Networks," *Proc. ACM Internet Measurement Conf.*, Oct. 2006.

[30] T. Kerins, W. Marnane, E. Popovici, and P. Barreto, "Hardware Accelerators for Pairing Based Cryptosystems," *IEE Proc. Information Security*, vol. 152, pp. 47-56, Oct. 2005.

[31] J. Chu, K. Labonte, and B.N. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems," *Proc. SPIE ITCom: Scalability and Traffic Control in IP Networks*, vol. 4868, July 2002.

[32] L. Chen, K. Harrison, N. Smart, and D. Soldera, "Applications of Multiple Trust Authorities in Pairing Based Cryptosystems," *Proc. Infrastructure Security Conf. (InfraSec '02)*, pp. 260-275, 2002.

[33] P. McDaniel and A. Rubin, "A Response to "Can We Eliminate Certificate Revocation Lists?"," *Proc. Financial Cryptography Conf. (FC '00)*, Int'l Financial Cryptography Assoc. (IFCA), Feb. 2000.

[34] R.L. Rivest and B. Lampson, "SDSI—A Simple Distributed Security Infrastructure," *Proc. Int'l Cryptology Conf. (CRYPTO '96)*, Rump Session, 1996.

[35] N. Daswani and H. Garcia-Molina, "Query-Flood DoS Attacks in Gnutella," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02)*, pp. 181-192, 2002.

[36] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta, "Limiting Sybil Attacks in Structured Peer-to-Peer Networks," *Proc. IEEE INFOCOM '07*, May 2007.

[37] *Tor: Anonymity Online*, Electronic Freedom Foundation, http://tor.eff.org/, 2007.

[38] J.H. Saltzer, D.P. Reed, and D.D. Clark, "End-to-End Arguments in System Design," *IEEE Trans. Computer Systems*, vol. 2, no. 4, pp. 277-288, Nov. 1984.

[39] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, *STUN—Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)*, RFC 3489, 2003.

[40] E. Damiani, S.D.C. di Vimercati, S. Paraboschi, P. Samarati, and F. Violante, "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02)*, Nov. 2002.

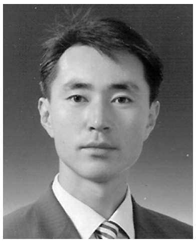
[41] N. Borisov, "Computational Puzzles as Sybil Defenses," *Peer-to-Peer Computing*, pp. 171-176, 2006.

[42] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman, "SybilGuard: Defending against Sybil Attacks via Social Networks," *Proc. ACM SIGCOMM '06*, Aug. 2006.

[43] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," *Proc. IEEE Symp. Security and Privacy*, May 2008.

[44] C. Lesniewski-Laas, "A Sybil-Proof One-Hop DHT," *Proc. Workshop Social Network Systems*, Apr. 2008.

**Kevin R.B. Butler** received the BSc degree in electrical engineering from Queen's University, Kingston, Ontario, in 1999 and the MS degree in electrical engineering from Columbia University in 2003. He is a PhD candidate in computer science and engineering at the Pennsylvania State University, University Park. His research interests include systems and storage security. He has also closely examined security and policy considerations for interdomain routing, and has investigated issues in secure hardware, privacy, and worm propagation across the Internet and in wireless networks. He is a student member of the IEEE and the IEEE Computer Society.



**Sunam Ryu** received the BS degree in computer science from Korea Military Academy, South Korea, in 1996 and the MS degree in computer science and engineering from Pennsylvania State University in 2007. He is a military officer in the Defense Security Command, Gwacheon-si, South Korea. His research interests include computer and network security issues in peer-to-peer systems and distributed systems.

**Patrick Traynor** received the BS degree in computer science from the University of Richmond in 2002 and the MS and PhD degrees from Pennsylvania State University in 2004 and 2008, respectively. He is an assistant professor in the School of Computer Science, Georgia Institute of Technology, Atlanta and a member of the Georgia Tech Information Security Center (GTISC). His research interests include the security of telecommunications networks and their interconnections with the Internet and the system challenges of applied cryptography. He is a member of the IEEE.

**Patrick D. McDaniel** is an associate professor in the Department of Computer Science and Engineering, Pennsylvania State University, University Park and a codirector of the Systems and Internet Infrastructure Security Laboratory. His research interests include network, telecommunications, and systems security, language-based security, and technical and public policy issues in digital media. He was awarded the National Science Foundation CAREER Award and has chaired several top conferences in security including, among others, the 2007 and 2008 IEEE Symposium on Security and Privacy and the 2005 USENIX Security Symposium. He is the editor-in-chief of the *ACM Journal Transactions on Internet Technology* (TOIT), and serves as an associate editor of the journals *ACM Transactions on Information and System Security* and the *IEEE Transactions on Software Engineering.* Prior to pursuing his PhD in 1996 at the University of Michigan, he was a software architect and a program manager in the telecommunications industry. He is a senior member of the IEEE.