Secure Resolution of End-Host Identifiers for Mobile Clients

Samu Varjonen[*], Tobias Heer[†], Ken Rimey[‡], Andrei Gurtov[‡§]
[*]Helsinki Institute for Information Technology HIIT / University of Helsinki
[†]RWTH Aachen University, Distributed Systems Group
[‡]Helsinki Institute for Information Technology HIIT / Aalto University
[§]CWC, University of Oulu

***Abstract*—Many efforts of the network research community focus on the introduction of a new identifier to relieve the IP address from its dual role of end-host identifier and routable locator. This identifier-locator split introduces a new identifier between human readable domain names and routable IP addresses. Mapping between identifiers and locators requires additional name mapping mechanisms because their relation is not trivial. Despite its popularity and efficiency, the DNS system is not a perfect choice for performing this mapping because identifiers are not hierarchically structured and mappings are frequently updated by users. In this paper we discuss the features needed to resolve flat identifiers to locators in a secure manner. In particular, we focus on the features and the performance that identifier-locator split protocols require from a mapping system. To this end, we consider a mapping system for an identifier-locator split based mobility solution and evaluate its performance.**

***Index Terms*—Identifier-locator split, Name resolution, Host Identity Protocol, Security, DNS, DHT, OpenDHT**

I. Introduction

In the evolution of the Internet, IP addresses initially served hosts as their identifiers *and* their routable locators. Although this dual role simplified many design decisions in the communication stack, it has been called into question because it hampers dynamics and flexibility in today's networks. Identifier-locator split protocols address this problem by limiting IP addresses to being pure locators and by introducing a new identifier above the network layer. This new identifier is often not routable and serves purely for host identification. With this split, these protocols support current requirements in the Internet, including security, mobility, and multihoming.

While the identifier-locator split has clear benefits, it also introduces its own problems. In addition to the existing Domain Name System (DNS) mappings, identifiers must be resolved to one or more locators. Despite the similarity of both resolution steps, the identifier-locator split introduces requirements that the current name resolution architecture cannot handle in practice. First, the pattern of the requests changes from *name-to-locator* to *name-to-identifier* and *identifier-to-locator*, where the identifier may belong to a flat namespace and the locator may change frequently. Second, the system must support fast mapping updates for mobile hosts. Third, secure user-generated updates must be supported.

The current DNS was designed for an Internet that consisted of stationary nodes. As such, the DNS was built for frequent reads and occasional updates. In contrast, mobile hosts need to update their location in the identifier mapping system quickly to stay reachable. Such updates pose new challenges to performance and security since the DNS is mainly an administered environment in which end hosts typically do not have direct write access to their DNS records.

In this paper we present an architecture that maps fully qualified domain names (FQDNs) to end-host identifiers (EIDs) using the DNS, and that maps the EIDs of a host to its routable locators (RLOCs) using a distributed hash table (DHT). Our contribution consists of an in-depth analysis of the problem domain and the design of a secure resolution architecture for identifiers and locators for mobile users. As a proof of concept, we present practical experience with an implementation of the resolution architecture.

The rest of this paper is structured as follows: In Section II we give a brief introduction to the identifier-locator split, and in Section III we discuss the problems that a resolution system has to face. We also offer a general solution to these problems and highlight its properties. In Section IV we introduce the details of the resolution system. In Section V we present a qualitative analysis of the feasibility of our proposal using the Host Identity Protocol as an example. We also provide a high level analysis of the processing times in comparison with the observed processing times of live systems. Section VI discusses related work and Section VII concludes the paper.

II. Introduction to the Identifier-locator Split

Currently discussed identifier-locator split protocols follow one of two principles: address rewriting, or mapping and encapsulating. In the address rewriting method, an IPv6 address is divided into a front and a back half. The front half of the IPv6 address represents the locator of the host, and the back half represents its identity. The Identifier-Locator Network Protocol (ILNP) [1] is a protocol that implements the address rewriting method. The deployment of address rewriting schemes requires major renumbering in the network and compulsory support for IPv6 because of the insufficient address length of IPv4. The current Internet is in a transition phase in which IPv6 connectivity cannot be guaranteed everywhere yet. This hampers the immediate deployment of protocols that rely on IPv6.

In mapping and encapsulating schemes, additional identifiers are mapped to locators, and packets are encapsulated so that locators are only used in the packet headers at the network layer, while higher layers see the identifiers in the encapsulated packets. Mapping and encapsulating approaches can be grouped into two categories: network-based and host-based encapsulation. The Locator/Identifier Separation Protocol (LISP) [2] is an example of a network-based approach. The Host Identity Protocol (HIP) [3] is an example of a host-based protocol. Mapping and encapsulating-based approaches have the benefit that they work on top of IPv4 as well as on top of IPv6. In addition, mapping and encapsulating schemes do not require changes to the core routing of the Internet. In this paper we focus on identifier-locator split protocols that implement the mapping and encapsulating scheme, and more specifically, on host-based approaches, which use additional identifiers in and above the transport layer. These approaches show two distinct requirements that set them apart from other identifier-locator split protocols: host-based locator updates and support for flat namespaces.

III. System Requirements

The introduction of end-host identifiers changes the way names are resolved at the beginning of a communication session. With the identifier-locator split, hosts have to resolve FQDNs to EIDs and EIDs to RLOCs. An essential question is whether the existing DNS name resolution infrastructure can cope with this task and how an alternative system should function. There are four problem areas that a name resolution structure for locator-identifier split mappings must tackle: a) In most cases the EIDs are based on a flat and often cryptographic namespace (e.g., the EID can be the hash of a public key of an asymmetric key pair identifying the host). It is known that DNS does not cope well with data that has no hierarchical structure. b) The architecture has to support user-generated and user-updated mappings. In the current DNS, names are mapped to a relevant authority controlling a subspace of the namespace. In some cases there is no authority for the user to turn to. For example, in HIP, the identifiers are self-created by the users, and in most cases the users do not belong to any organization that grants them modification rights to a DNS sub-domain, such that they could store and update their mappings. c) The architecture has to be secure; for example, it has to prevent attackers from forging identities and mappings of clients. Additionally, the system must prevent attackers from flooding the resolution system with bogus mappings to drown valid mappings. d) Finally, the system has to operate in an efficient manner.

A. Support for Flat Namespaces

The nature and structure of identifiers depends on the chosen identifier-locator split protocol. Identifiers can be divided into two categories. The first category represents human-readable identifiers at the application layer. Domain names, the most prominent human-readable identifier, are managed and structured in a hierarchical way, reflecting the hierarchy prevalent in the management of networked systems. The second category represents binary identifiers that may or may not be organized in a hierarchical way. These identifiers can consist of any sequence of bits without taking human readability into account. In the network community, there is a trend towards cryptographic identifiers to provide inherent security when addressing a host or service. An example of such a cryptographic identifier is HIP's Host Identity Tag (HIT), a form of public key fingerprint. Such cryptographic names have little or no hierarchical structure, making it difficult to assign the management of the identity to an organization as the DNS does for human-readable names.

B. Rapid Mapping of User-generated Updates

Using DNS to store all the required mapping information (domain name, identifier, and locator) would suffice for stationary nodes under administrative management, but would not be a good choice for mobile nodes and for users who do not have modification rights to the DNS. To allow fast mapping updates for mobile hosts, which need to change their IP address mapping rapidly to stay reachable, the DNS Resource Records (RRs) would have to use low time-to-live (TTL) values, or caching would have to be disallowed. However, high TTLs and caching are cornerstones of the scalability and performance of the DNS. Abandoning them for a considerable proportion of entries would seriously degrade the performance of the system as a whole.

This paper considers the proposition that mobile nodes with access to the DNS should use it to map FQDNs to EIDs, and as the research community has adopted distributed hash tables (DHT) to handle flat namespaces, the paper furthermore assumes that EIDs should be resolved using DHTs. DHTs do not employ hierarchical caching and thus allow for immediate mapping updates. However, using a DHT results in a higher communication overhead within the name mapping system (c.f. Section V).

C. Securing Mapping Updates

The DNS, as a hierarchical and administered name resolution system, is widely regarded as secure. Even without cryptographic protections like DNSsec [4], fraudulent behavior requires access to the DNS infrastructure itself and is typically limited to a single compromised sub-domain. Tampering with DNS entries on a global scale requires considerable effort. In addition, DNSsec protects the system against spoofing attacks in which a malicious user tries to forge an answer from the DNS or tries to claim that the queried name does not exist.

However, as discussed before, the DNS system was not designed for large amounts of fast *user-generated* updates. Besides technical challenges, security issues arise when allowing users to modify the contents of the mapping system. The name lookup at the beginning of a communication session is a vulnerable phase. Tampering with it may allow direct as well as indirect denial-of-service (DoS) attacks (e.g., by invalidating the locator mapping or by redirecting traffic addressed to a popular host to a victim). Therefore, the system must be protected regardless of the resolution system.

We illustrate the arising issues using the example of OpenDHT[1] as a system that allows user-generated updates. OpenDHT has been proposed as one choice for a collaboratively managed DHT [5] (see Section VI for further examples). OpenDHT is a publicly available DHT service running in PlanetLab, a world-wide testbed of several hundred servers. In contrast to other DHT systems, users do not have to run a local DHT node to be able to access it. OpenDHT does not require registration to insert and look up data. The open access philosophy of OpenDHT matches the requirements for global name resolution well, because requiring each end host to sign up for a name mapping service hardly matches the principles of the Internet. Available storage and bandwidth in OpenDHT are shared among all users [6].

OpenDHT stores one or several values under each key (e.g., the EID in a name lookup system). This convention is prone to flooding and index poisoning attacks [7]. In these attacks the malicious user stores as much false or random information under the attacked key as possible, thereby effectively drowning the original value. This allows malicious users to present seemingly correct information in the DHT. Index poisoning in an ID-locator mapping service can even be used to mount distributed DoS attacks against victim hosts. Consider a case in which a malicious user uploads the victim's locator under the identifiers of some popular services. This would redirect the traffic destined for the services to the victim's system, thereby overloading its downlink.

There are two possible solutions for this problem: a) The DHT is agnostic with respect to the contents it stores and leaves it to the end host to implement security or b) the DHT enforces the correctness of mappings and updates to mappings before accepting them. Solution a) can be achieved by attaching additional authentication information to the stored mappings (e.g., digital signatures). EIDs based on a cryptographic namespace (e.g., HITs in HIP) simplify this approach because each host can use its EID to sign its locator set. Such signatures would enable a querier to identify the correct value among a set of forged locators. However, in an index poisoning attack, it would also mean that the querier would have to verify the signatures of many returned locators until it identifies a valid entry among the flood of bogus mappings. In contrast, in solution b), DHT nodes would verify the authenticity of the signatures before accepting a new key-value pair. This method requires replay protection to prevent attackers from republishing properly signed but outdated locator mappings.

IV. Resolution System Design

This section summarizes the previous discussion and proposes a secure name resolution architecture in which clients map EIDs to locators using a DHT. We assume that the EIDs are derived from public keys (as HITs in HIP are) and that hosts can prove the possession of an EID by using their private keys. We use HIP as an example of a host-based identifier-locator split protocol because it includes all of the security features that our proposed architecture requires. Moreover, we show how the security features of HIP support the requirements listed in the previous section.

A. General Design

Since the current DNS is sufficient to store the long-lasting mappings from FQDNs to EIDs and these mappings may be independent of the locator-split protocol, we treat the first name resolution step as an orthogonal issue and assume that appropriate measures are taken to ensure secure operation (e.g., by employing DNSsec). However, note that the FQDNs are resolved to EIDs instead of locators.

In the second name resolution step, a DHT is used to map the EIDs to locators. We assume that the EID (or a value that can be securely derived from it) is used as the key in the DHT. The value stored under the key consists of the public key of the host, its locators, a sequence number, and a signature created with the private key of the host. The signature and the sequence number prove to the clients and the DHT nodes that the locator mapping is authentic.

In the previous section, we noted that DoS and replay protection measures are needed to protect the DHT. This requires a challenge-response mechanism for verifying that the host owns the public-keys related to the EID for which it updates the locator mapping. If this verification succeeds, the mapping is stored; otherwise it will be dropped.

B. An Identifier Resolution System For HIP

The Host Identity Protocol [3][8][9] introduces a new cryptographic namespace between the transport and IP layers. The namespace is based on public-key cryptography and consists of so-called Host Identities, which are RSA and DSA public keys. Using full-length public keys in packet headers would result in too much overhead and would be incompatible with unmodified (legacy) applications. For this reason, public keys in HIP are also represented in a shorter 128-bit (IPv6-compatible) format, called the Host Identity Tag. HITs can be used directly with IPv6-enabled applications because of their size and format. Since HIP uses cryptographic keys as identifiers, host authentication and the establishment of a secure channel between HIP hosts is very simple. Moreover, HIP is designed to be extensible. A modular packet and parameter concept allows adding new functionality to HIP easily. HIP parameters are carried in HIP control packets.

In essence, the HIP base exchange (BEX) is a four-way handshake and key negotiation phase to create an IPsec security association between hosts. The BEX verifies that the peers own the private keys that were used to create their identities. The BEX also includes puzzle protection against DoS attacks and other flooding attacks. In our approach, we use the BEX as the challenge-response mechanism for verifying the ownership and freshness of the locators to be stored in the DHT.

[1] In May 2009, the maintainer of OpenDHT informed the community that the service would be discontinued. However, since it was a widely used service for years, we still use OpenDHT as a practical example of a DHT.

In order to initiate the BEX, the *initiator* (the host that initiates the communication) needs to know the HIT of the *responder* (e.g., a server) and a way to map the HIT to an IP address. Currently HIP offers two different ways to perform the resolution. In our solution, both of these approaches are used. Firstly, DNS can be used to store HIP-related identifiers using HIP Resource Records (HIP RRs) [10] protected by signatures. This allows for translation of FQDNs to Host Identities (HIs). The resolver then issues an *A* query to map the HIT into the host's IP addresses. Alternatively, HIP can utilize the HIP DHT interface [5]. The HIP DHT Resource Record (HDRR) is a HIP control-packet-like structure used to store HIP mappings in OpenDHT. It can contain multiple IPv4 and/or IPv6 addresses, and it also contains the Host Identity (the public key from which the EID was created). The HDRR is protected by a signature calculated over the HIP packet header and the included parameters. For our purposes, this format lacks only a sequence number to prevent replay attacks. However, due to the modular parameter concept, the sequence number is easy to add to the HDRR. We can even reuse the sequence number used in the basic HIP control packets, because the parameter format for the HDRR is the same as in HIP control messages.

We use the client authentication in the HIP BEX to prevent attackers from inserting forged locators for EIDs of other hosts into the DHT. By requiring a HIP connection between the client and the DHT node, the client has to prove that it is uploading mappings for its legitimate host identity – the key in the DHT. Implementing this authentication check is simple and can be done through the standard Berkeley Sockets API. The HIT, as an IPv6-compatible identifier, can be used directly by any IPv6-capable DHT. The authentication, as well as the basic DoS protection, are handled on the HIP layer. The only modification required to the DHT is a test for equality of the HIT and the key in the *put* message. The self-certifying property of the HIT obsoletes further authentication measures like client certificates or user registration.

Depending on the security architecture of the DHT, a client should either perform the BEX with a DHT gateway node, or with the node storing the EID and locator information if the DHT system cannot be regarded as secure. Such protection is only possible if the DHT is used exclusively for HIT-to-locator resolution, because a general-purpose DHT is not able to guarantee the cryptographic binding between a host and the updated keys

V. Evaluation

In this section we study the feasibility of the system by analyzing our proof-of-concept implementation. Our prototype uses the Host Identity Protocol as the identifier-locator split protocol. HIP was chosen because it readily includes cryptographic identifiers that can be used for authentication, a built-in challenge-response mechanism, and a cryptographic puzzle mechanism for DoS prevention. We implemented a DHT Interface [5] for the *HIP for Linux* (HIPL) implementation[2] to store the identifier-locator mappings.

TABLE I
Computational complexity of cryptographic operations in HIP.

	operations per sec.	ms/operation
HMAC(MD5)	42267	0.02
SHA-1	30809	0.03
DSA signature	1887	0.53
DSA verify	1645	0.61
RSA signature	890	1.12
RSA verify	18502	0.05
DH key generation	51	19.64

Our resolution system does not pose a need for extensive infrastructure changes. The support for HIP RR and the DHT nodes can be provided gradually by the willing Internet operators. Moreover, there is no need for complete end-system penetration, e.g. clients that do not support HIP do not need to support our resolution system and their use of the DNS is not disrupted.

We begin our performance evaluation by analyzing the number of required cryptographic operations and messages for inserting a new entry into the DHT based on our HIP-centric security solution. We focus on the performance of the resolution system and not on the performance of mobile nodes. We also provide measurements of the processing times for the Bamboo DHT[3] and for OpenLookup v2,[4] to show the resolution performance of a real system.

A. Feasibility

Performance is a primary concern for a resolution system that employs online public-key operations. The HIP BEX is dominated by the processing times for creating and verifying the public key signatures. Hence, we measured the performance of these operations to estimate the number of clients that a server is capable of serving per second. The cryptographic operations are only relevant for write operations because reads do not require authentication. Hence, reads will be drastically faster. Moreover, we do not consider routing overhead in the DHT in this measurement either. The goal of the analysis is not to give an accurate estimation of the expected performance of a world-wide system, but rather to provide a general impression of the feasibility of using HIP for securing a DHT system for HIT-to-IP mapping.

We used a quad core Intel Xeon 5130 running at 2 GHz with 2 GB of main memory to perform the cryptographic calculations. However, the cryptographic measurements used only one core. We conducted the cryptography tests with the OpenSSL 0.9.8g speed test. The results of the tests are shown in Table I. We used 1024-bit keys because that is the default size for the RSA and DSA keys in the HIP for Linux implementation, which we used for latency measurements. For the hash functions, we used the maximum block size of 8192 bytes.

[2] HIPL, http://www.infrahip.net
[3] http://sites.google.com/site/lxpworkroom/bambooipv6version
[4] http://openlookup.net/

Based on the information presented in Tables I and II, we can calculate that it will take circa 20.8 ms to complete the cryptographic calculations needed in the BEX on the server side (1 RSA signature, 1 RSA verification, and 1 Diffie-Hellman key generation). This amounts to 192 key updates processed by each DHT node per second. Considering a system comparable to OpenDHT, which consisted of about 150 nodes on average, the system could process about 28,800 updates per second. These estimations show that even a moderately-sized system can support a substantial number of mobile devices.

B. Resolution and Update Delay

The resolution and update delay of the system is a crucial factor for clients because the resolution step precedes every communication to each host for which the HIT-to-IP mapping is not known. The update delay determines how long the locator information in a DHT stays outdated upon a change of the locator. In our system, read accesses are protected by a signature in the DHT resource record while write accesses are protected using HIP between the client and the DHT. Using HIP prolongs the update process by the time required for cryptographic processing plus two RTTs for establishing the HIP association to the gateway or DHT node.

We measured the mean latency of an IPv6-enabled Bamboo DHT and OpenLookup v2 to determine the resolution performance of these systems. OpenLookup v2 implements the same XML-RPC client interface as the Bamboo DHT, but it is not strictly speaking a DHT, and it does not share data with OpenDHT. OpenLookup v2 is an administratively decentralized system based on full data replication.

Bamboo DHT and OpenLookup v2 were running on the same hardware as described in Section V-A. We used a laptop with Intel Core 2, 2 GHz CPU processor with 2 GB of main memory as the client. All machines involved in our measurements were located in our local Gigabit network with a mean round-trip latency of 0.88 ms (std.dev. 0.03 ms). Since we only modified the lookup and update API, we focus on the communication between the end host and the resolution system. For this reason, we made our measurements using a Bamboo DHT configuration containing just one node. Thus the results obtained do not reflect the expected total lookup time of a world-wide deployment requiring routing within the DHT. A second reason to exclude the DHT lookup times is that these strongly differ with the number of DHT nodes and their and technical specifications. Hence, including lookup times would blur the statements about the interface performance.

TABLE II
CRYPTOGRAPHIC AND COMMUNICATION OVERHEAD OF THE HIP BEX.

	Verify		Sign		DH key	
	PK	HMAC	PK	HMAC	Generation	# of msgs
Initiator	2	1	1	1	1	2
Responder	1	1	1	1	1	2

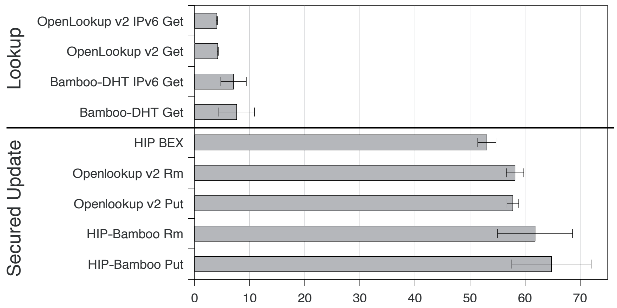

Fig. 1. Latencies of update and get operations (in ms).

Figure 1 shows that OpenLookup v2 performs slightly better in the lookup operations in comparison to the Bamboo DHT. The larger mean values and standard deviations for the Bamboo DHT are due to an unnecessary periodic delay caused by the queue management in the iterative lookup procedure, resulting in an extra delay for a small fraction of the requests (approx. 10%). Since the shortest latencies of the Bamboo DHT (IPv4 3.8 ms, IPv6 4.0 ms) match the shortest lookup times of OpenLookup v2 (IPv4 4.2 ms, IPv6 4.0 ms) we assume that both systems achieve a similar level of performance under realistic conditions. The measurements show that the additional delay for the systems is low. However, these numbers only consider the Interface delay and not routing within the lookup structure.

The tested systems do not support updates of keys. However, by deleting a key and re-inserting it with new locator information, updates can be achieved. The latency of updating a record in the system is 116.4 ms (±0.8 ms) for OpenLookup v2 and 127.6 ms (±5 ms) for the Bamboo DHT. In Figure 1 we also show the latencies of the HIP BEX in the test environment and the latencies of a Bamboo DHT utilizing HIP. Utilizing HIP in the Bamboo DHT and in OpenLookup v2 did not add latency, other than the 53 ms (±3.3 ms) caused by the BEX. Otherwise, the Bamboo DHT and OpenLookup v2 utilizing HIP performed as expected from the results in the *get* case. As pointed out in the previous section, the main cause of the delay introduced by HIP is the cryptographic operations during the BEX. In contrast to our estimation of the additional cryptographic load on the DHT gateways, the additional delay of 53 ms in Figure 1 also includes the processing time of the client, the packet processing, and the network latency. Our measurement focused on the performance of the gateway and do not take into account the higher RTTs between the client and the gateway under realistic conditions. As expected, HIP introduces a notable delay for updates; however, at the same time it eliminates the possibility of index poisoning and forged locator updates, without requiring additional administrative measures like user registration.

VI. RELATED WORK

Mathy et al. [11] describe how LISP-DHT serves as an efficient and secure mapping service for the LISP 3 variant. LISP-DHT requires every autonomous system (AS) to have its own DHT node to serve identities in the AS. In LISP-DHT, security is based mainly on the assumption that the architecture is administered and joining the DHT requires a valid X.509.v3 certificate. To improve efficiency, LISP-DHT proposes the use of a Stealth DHT, where client nodes may acquire DHT routing information (but do not take responsibility for any data segment on the ring). In this way, the stealth nodes can inject lookups into the system in a more efficient way than by always directing queries via a gateway node. LISP reduces latencies by caching and so hinders mobility. Mobility in LISP and its influence on name resolution are currently under design [12].

In the Node Identity architecture [13], the node identities are the public keys of public-private key pairs. Name resolution in the Node Identity architecture uses the DNS to map FQDNs to EIDs, while EIDs are mapped to locators using a global DHT shared by all node identity routers. The Node Identity architecture is similar to LISP in the sense that both are network-based and need customized routers to work. The security of the mappings in the architecture is only briefly addressed by stating that the security is inherited from registration security. However, the registration security is not discussed in detail.

DHT-MAP [14] proposes a mapping system, useful for LISP and similar protocols. The difference relative to other solutions is that EIDs are mapped in the DHT to the address of a server that handles the resolution to a host's real RLOC. Mobility is supported by allowing the mobile host to register with the resolution server of the access network to which it attaches. In this way, DHT-MAP avoids triangular routing and the concept of a *home network*. Luo et al. [14] state that their approach may allow EID spoofing attacks, and they suggest a challenge-response mechanism similar to the mechanisms provided by HIP.

Baumgart [15] proposes a distributed two-stage name resolution service (P2PNS) built on top of a DHT. That paper presents requirements and solutions similar to ours but does not discuss mobility. In P2PNS, flooding attacks are hampered by introducing computational puzzles that have to be solved before the mappings can be inserted. As an additional feature, the number of values under a key is restricted. When a key is queried from P2PNS, it is queried in parallel from all replicas that have the key and its value. Based on the received values, the issuer of the query makes a majority decision.

VII. Conclusion

In this paper, we presented a discussion about identifier resolution for identifier-locator split protocols and pointed out the shortcomings of the current Domain Name System (DNS). Based on our observations, we described an architecture for secure identifier-locator mappings based on a distributed hash table. In particular, we discussed three core problems of name resolution for host-based identity locator split protocols: a) support for flat namespaces, b) rapid user-generated updates, and c) the security of the mappings.

We address these problems by implementing secure key updates based on the cryptographic properties of the identifiers in the Host Identity Protocol (HIP). Our system works with user-generated identities and does not require any user management or the deployment of a global PKI system because it makes use of the self-certifying identities in HIP. With its identity concept and IPv6 compatibility, HIP integrates nicely into existing lookup systems and enhances their security features with DoS resilience and authenticated locator updates. Our performance analysis of the HIP-enabled DHT API demonstrates the feasibility of the architecture and indicates that employing HIP as a security solution provides acceptable performance with considerably increased security.

References

[1] R. Atkinson, "ILNP Concept of Operations: draft-rja-ilnp-intro-03," Feb. 2010, work in progress.

[2] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP): draft-ietf-lisp-06.txt," Jan. 2010, work in progress.

[3] R. Moskowitz and P. Nikander, "RFC 4423: Host Identity Protocol (HIP) Architecture," May 2006.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol Modifications for the DNS Security Extensions," RFC 4035 (Proposed Standard), Internet Engineering Task Force, March 2005, updated by RFC 4470. [Online]. Available: http://www.ietf.org/rfc/rfc4035.txt

[5] J. Ahrenholz, "HIP DHT Interface: draft-ahrenholz-hiprg-dht-06," Nov. 2009, work in progress.

[6] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Uses," in *Proceedings of ACM SIGCOMM 2005*, Aug. 2005.

[7] J. Liang, N. Naoumov, and K. Ross, "The index poisoning attack in p2p file sharing systems," in *INFOCOM*. IEEE, 2006.

[8] R. Moskowitz, P. Nikander, P. Jokela, and T. R. Henderson, "RFC 5201: Host Identity Protocol," Apr. 2008.

[9] A. Gurtov, *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley and Sons, 2008.

[10] P. Nikander and J. Laganier, "RFC 5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension," Apr. 2008.

[11] L. Mathy and L. Lannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," in *Proc. of ACM ReArch 2008*, Dec. 2008.

[12] D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, "LISP Mobility Architecture: draft-meyer-lisp-mn-01," Feb. 2010, work in progress.

[13] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, "A node identity internetworking architecture," in *Proc. of INFOCOM 2006. 25th IEEE International Conference on Computer Communications.*, Apr. 2006.

[14] H. Luo, Y. Qin, and H. Zhang, "A DHT-Based Identifier-to-Locator Mapping Approach for a Scalable Internet," in *IEEE Transactions on Parallel and Distributed Systems*. IEEE Computer Society, Feb. 2009.

[15] I. Baumgart, "P2PNS: A Secure Distributed Name Service for P2PSIP," in *Proc. of 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, 2008.