NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-09-12
January 30, 2009

A Feeder-Carrier-Based Internet User Accountability Service

Leiwen Deng and Aleksandar Kuzmanovic

Abstract

This paper presents *IDnet mesh*, a general-purpose user identity architecture for the Internet, which provides a scalable common *identity validation* service to the public. This common service can enable diversified new Internet services as well as new features for existing ones. It builds upon a novel *feeder-carrier* identity architecture which increases resilience to rising provider-initiated surveillance attempts. It offers a regular approach to connect a user's online identity with the user's real identity and meanwhile fully preserves the user's privacy on the public Internet. Our system adopts a practical trust model such that it yields high system evolvability towards global deployment; it requires no change to the current Internet infrastructure and protocols, and therefore is completely incrementally deployable.

We use a Linux-based implementation of IDnet mesh algorithm and protocols at a cluster of servers in Emulab to perform benchmarks for the core algorithm and to test the functional integrity of the protocol implementation. We perform extensive evaluation of IDnet mesh's scalability, security, efficiency, and reliability. Finally, we assess the overhead induced by our system in the cases of Email and Web services and demonstrate that IDnet mesh can be scalably integrated with these services.

Keywords: IDnet mesh, feeder, carrier, user accountability, identity validation

A Feeder-Carrier-Based Internet User Accountability Service

Leiwen Deng
Northwestern University
Evanston, IL, USA
karldeng@cs.northwestern.edu

Aleksandar Kuzmanovic
Northwestern University
Evanston, IL, USA
akuzma@cs.northwestern.edu

ABSTRACT

This paper presents *IDnet mesh*, a general-purpose user identity architecture for the Internet, which provides a scalable common *identity validation* service to the public. This common service can enable diversified new Internet services as well as new features for existing ones. It builds upon a novel *feeder-carrier* identity architecture which increases resilience to rising provider-initiated surveillance attempts. It offers a regular approach to connect a user's online identity with the user's real identity and meanwhile fully preserves the user's privacy on the public Internet. Our system adopts a practical trust model such that it yields high system evolvability towards global deployment; it requires no change to the current Internet infrastructure and protocols, and therefore is completely incrementally deployable.

We use a Linux-based implementation of IDnet mesh algorithm and protocols at a cluster of servers in Emulab to perform benchmarks for the core algorithm and to test the functional integrity of the protocol implementation. We perform extensive evaluation of IDnet mesh's scalability, security, efficiency, and reliability. Finally, we assess the overhead induced by our system in the cases of Email and Web services and demonstrate that IDnet mesh can be scalably integrated with these services.

1. INTRODUCTION

Problem. "On the Internet, nobody knows you're a dog," states Peter Steiner's famous New Yorker cartoon [13]. Fifteen years have passed since this cartoon was first published, and things have not changed. Indeed, the Internet architecture hides a user's real identity by design, which causes tremendous problems on a daily basis, simply because there are no effective means to enable *user accountability*. Here, the user accountability means a regular traceability (or auditability) to a user's real identity based on the user's online identity.

The fundamental question we attempt to answer in this paper is the following: *If we want to enable user accountability globally on the Internet today, how can we achieve that?* To motivate this approach, and to demonstrate limitations of the current user identity practices, consider the following examples.

Examples. Email address is a typical example of a user's online identity. Using a security option such as OpenPGP [28], we can well verify the association between a user's Email address and the messages that he sends. However, even with OpenPGP, we have no effective way to counter SPAMs because the Email address carries no meaning about the user who sent us the email if we do not know him in advance. We can hardly tell whether he is a spammer or not. Without the ability to identify who sent it, we accept all Emails sent to us, the majority of which are typically unwanted.

Web account is another example of a user's online identity. However, except for accounts at a small fraction of Web sites that require a user's real name information (*e.g.*, credit card number) for registration, the rest usually carry little meaning about a user's real identity. As a result, they are helpless to counter vandals or spammers. Vandals and spammers are posing significant threats to the rising Web 2.0 applications [40] that are designed to enhance information sharing, collaboration, creativity, and functionality of the Web. More fundamentally, it has become impossible to understand if comments at a site are biased or not. For example, enterprises such as Sony or Wal-Mart have already been caught creating fake blogs [25] to bias customers.

Social-networking sites such as MySpace and Facebook have grown exponentially in recent years, with teenagers making up a large part of their membership. This has created a new venue for sexual predators who lie about their age to lure young victims and for cyber bullies who send threatening and anonymous messages. Under mounting pressure from law enforcement and parents, MySpace agreed in January 2008 to take steps to protect youngsters from online sexual predators and bullies, including to search for ways to better *verify users' age* [2, 12]. MySpace acknowledged in the agreement that it would develop online identity authentication tools. Skeptics are doubtful that MySpace and similar sites can eliminate the problem because such tools could be difficult to implement and predators are good at circumventing restrictions [2]. We argue below that such tools are feasible, and provide the design and implementation of a solution.

Solution. In this paper, we propose a general purpose user identity architecture for the Internet — *IDnet mesh*, which can enable user accountability globally. Comparing with existing unified identity solutions, *e.g.*, *OpenID* [14], *Microsoft Passport* [26], and *VeriSign unified authentication* [22], the IDnet mesh makes a fundamental change in its technical approach by supporting *feeder-carrier decoupling* — it decouples identity providers into *feeders* and *carriers*. Feeders are identity providers where users register and carriers are identity providers where users are authenticated. A feeder can export authentication data to many carriers such that a user registered at the feeder can be authenticated at any

of these carriers independently of the feeder. None of the existing solutions supports the feeder-carrier decoupling. With this new feature, it becomes much easier to enable user accountability, and to achieve high authentication scalability, surveillance resiliency, efficiency, and robustness as we will describe in Section 3.

The IDnet mesh introduces a practical trust model that enables high system evolvability such that it has the potential to evolve towards global deployment within relatively short time. The system ensures user anonymity to the network without relying on additional encryption mechanisms (*e.g.*, SSL) and trust systems (*e.g.*, PKI). It therefore both preserves user privacy and makes the authentication efficient as we will show in Section 5. Moreover, we design an inexpensive biometric user device, *Internet passport*, which enables strong but convenient user authentication, thereby providing both high security and improved user experience. We implement the algorithm and protocols for the IDnet mesh and test their functional integrity on a testbed. We perform extensive evaluation of the architecture and show that it can achieve outstanding performance for scalability, security, efficiency, reliability, and incremental deployability at the same time.

The rest of this paper is organized as follows. In Section 2, we introduce the related work and compare it with our approach. In Section 3, we introduce our basic design of the IDnet mesh. Then, in Section 4, we describe detailed system algorithm and protocols implementation. Next, in Section 5, we evaluate the system performance in terms of scalability, efficiency, reliability, security, and incremental deployability. We also show Email and Web application examples in this section. Finally, we conclude in Section 6.

2. RELATED WORK

Unified identity solutions. Existing identity solutions such as *OpenID* (*e.g.*, *Google's single sign-on* API [19]), *Microsoft passport*, and *VeriSign unified authentication* enable users to establish a single online identity to access information and purchase goods on multiple Web sites. This significantly simplifies a user's online experience by eliminating the need for maintaining multiple user accounts across different sites. The IDnet mesh inherits this property. Meanwhile, it makes a fundamental improvement by additionally supporting the feeder-carrier decoupling. By contrast, all the above solutions inevitably bind the roles of a feeder and a carrier together at each identity provider — where a user registers is where the user can be authenticated. With the feeder-carrier decoupling, the IDnet mesh makes it easier to support user accountability, high authentication scalability, surveillance resiliency, efficiency, and robustness as we will show in Section 3.1.

Host accountability vs. user accountability. *Accountable Internet protocol* (AIP) [29] proposes a network architecture that provides accountability as a first-order property; *host identity protocol* [16] (HIP) provides a network solution that decouples a host's identity from its topological location. Both solutions enable host accountability. However, host accountability is fundamentally different from the user accountability that our architecture can provide. Indeed, the key to solving those problems that we introduced in Section 1 is to enable a regular approach to apply liability. The liability is always applied to users, not hosts. Therefore, host accountability is insufficient. In addition, both HIP and AIP require fundamental changes to the current Internet infrastructure and protocols, and therefore are not incrementally deployable and readily available as our architecture is.

Trust model comparison. The trust model of IDnet mesh shares a flavor of *web of trust* [42] (a.k.a *OpenPGP's PKI* (*public key infrastructure*)) in that both of them exploit a bottom-up trust propagation process and use decentralized trusts, which is realistic in terms of the trust evolution nature. On the contrary, the *X.509 PKI* [10] assumes a strict top-down hierarchy of trust which relies on a single "self-signed" root that is trusted by everyone. The unreality of such a centralized trust structure at a global scale impedes the X.509 PKI from evolving to a global solution. Currently most X.509 PKI systems stay at enterprise scale.

The trust model of IDnet mesh differs from the web of trust in that it requires each identity provider to explicitly express its trust and prohibits the implicit transitive trust (*i.e.*, if a trusts b and b trusts c, we conclude that a trust c as well). Therefore, it prevents the uncertainty of trust caused by the implicit transitive trust during trust revocations. By contrast, the web of trust fundamentally depends upon the implicit transitive trust for trust propagation, hence it suffers from the uncertainty of trust problem.

Finally, IDnet mesh's trust model is much more practical than social-networking based solutions (*e.g.*, [39]), because it removes the trust "burden" from individual users and delegates this job to identity providers.

Digital certificate. Digital certificate (or public key) is an important technique to address user identity issues. However, as pointed out in [34], it is hard to design an effective user identity solution based on the digital certificate itself. The key difficulty lies in that digital certificate is hard to preserve user privacy on the public Internet because the public key is a fixed value, which enables others to easily track the user and infer his private information. We can think a user identity solution as the answer to the question "Who are you?" There are two different ways to answer it:

- Answer 1: "I'm an accountable user. My name is $\cdots$"
- Answer 2: "I'm an accountable user."

The digital certificate based solutions answer the question in the first way, which exposes a user's privacy. While the IDnet mesh answers it in the second way. Indeed, in many cases when Internet users are asking the question "who are you", what they want to know is just whether you are accountable or not. They do not care much about what your real name (or real identity) is.

So the second way can both well answer this question and preserve user privacy.

In most cases, before a digital certificate can be useful, we must first bind the digital certificate to the owner's identity [34]. But the question is what an effective representation of the owner's identity is. Indeed, the IDnet mesh is answering this question.

Anonymous authentication protocol. An anonymous authentication protocol such as *IBM idemix* [31] can authenticate a user while retaining user anonymity at the same time. Its central technology is based on the group signature cryptography [30]. A user can appear anonymous not only to the network, but to the identity provider as well. This provides a desirable feature to retain user privacy during the authentication. However, we do not adopt this technique in our system for two reasons: (*i*) Efficient membership revocation for large groups still remains an open question for group signatures [41]. The technique therefore is not suitable for a large scale system that we design. (*ii*) When the feeder-carrier decoupling is enabled, making users anonymous to the network is sufficient to retain user privacy. To additionally make them anonymous to the identity provider is unnecessary. We will explain this in detail in Section 3.1 and Section 5.5.2.

Kerberos. *Kerberos* [11] is a user authentication protocol originally designed to be used for a network administered by a single authority. The newest version of Kerberos introduces a cross realm authentication feature to make it feasible to scale to larger sets of networks. Our system design is similar in spirit to the Kerberos cross realm authentication. However, since our system is designed for a much larger network domain — the Internet, we focus on solving far more demanding scalability and security challenges.

In particular, like the digital certificate, Kerberos was never designed to preserve user privacy, *e.g.*, it exposes to the network a username in cleartext. Moreover, Kerberos cross realm authentication approach relies on a PKI to distribute its inter-realm shared keys and to resolve trust relationships, which make it challenging to be deployed at a global scale. This is not only because a global scale PKI does not exist, but also because it inevitably inherits all PKI's problems [34], some of which we discussed above.

3. BASIC DESIGN

3.1 Highlights

Design goal. Our goal is to find a best effort solution that can enable user accountability globally on the current Internet. This includes the following challenges:

- *Supporting user accountability.* It must be able to connect a user's online identity with the user's identity in the real world.
- *Preserving user privacy.* At the same time when it provides user accountability, it should retain user privacy and be resilient against surveillance.
- *Feasibility to deploy globally.* It must support high system scalability, DDoS resiliency, security, and robustness. It should minimize the technical complexity and trust thresholds when it evolve towards global deployment.
- *Incremental deployability.* We need it now. We therefore can not afford a clean-slate solution.

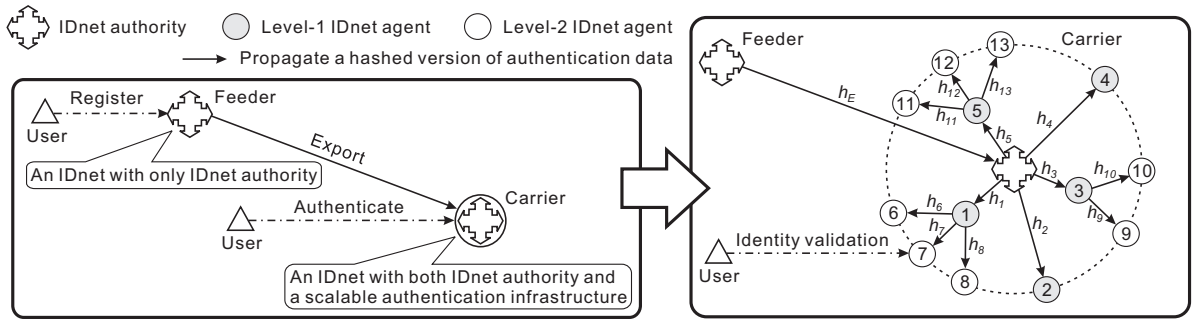
Feeder-carrier decoupling. To meet our goal, a novel technique that we devise and adopt is to decouple identity providers into feeders and carriers as shown in Figure 1(a). Feeders (or identity feeders) are identity providers where users register and carriers (or authentication carriers) are identity providers where users are authenticated. A feeder can export authentication data to many carriers such that a user registers at the feeder can be authenticated at any of these carriers independently of the feeder. To support user accountability, users should register at feeders with their real identities. (The real identities are never exported to carriers.) Therefore, when a dispute arises, there is a regular way to audit a user's real identity through the feeder.

Why feeder-carrier decoupling makes a fundamental change. The feeder-carrier decoupling fundamentally changes the following aspects:

1. *Much easier to enable user accountability using things we already have.* There are already many feeders that have users' real identities registered. For example, a school has the real identities of all its students; a bank has the real identities of all its customers; and a company has the real identities of all its employees. However, most of them are not likely to become carriers as well. With the feeder-carrier decoupling, we can create authentication data at these feeders and export the data to carriers to enable user accountability.

2. *Respecting user privacy.* An identity provider does not spontaneously respect user privacy unless it is forced to do so (*e.g.*, Google was caught disregarding user privacy in its Gmail service [3]). With the feeder and carrier decoupled, users can easily change carriers without taking pains to change the feeders where they registered if they feel some carriers do not respect their privacy. This enables an effective competition among carriers to respect user privacy. On the other hand, feeders can pose no threat on user privacy during authentication services. This is because the authentication processes only involve the carriers, but no feeders. No other solutions (including OpenID, Microsoft passport, and VeriSign unified authentication, *etc*) support such surveillance resiliency.

3. *Scalability and DDoS resiliency.* With the feeder and carrier decoupled, a carrier is relieved from user management burdens and it therefore can focus on providing scalable authentication services. Users and feeders can choose carriers that provide the best services. Moreover, a feeder can export authentication data to many carriers to further improve the scalability and DDoS resiliency for authentication services of its users. By contrast, the current practice is that the scalability relies solely on the single identity provider that users



a) Feeder-carrier decoupling b) IDnet implemention of the feeder and carrier

Figure 1: Feeder-carrier decoupling and its IDnet implementation

register to and users have no effective means to motivate a provider to optimize the scalability aggressively.

4. *Reliability.* A feeder can export its authentication data to many carriers and any of these carriers can provide authentication services for the feeder's users independently of the feeder. Such redundancy significantly raises the reliability of the authentication services. By contrast, nowadays, service outages can be caused by the single point of failure of the identity provider.

5. *Efficiency.* Suppose user b wants to authenticate user a and a's feeder has exported authentication data to many carriers around the world. User b can choose to use a carrier that is most close to it (to reduce delays) or that is most frequently used (to exploit caching) to improve authentication efficiency. By contrast, the current practice does not provide users such choices.

6. *Global level trust issue.* Suppose user b wants to authenticate user a and a's feeder has exported authentication data to many carriers around the world. User b can choose a carrier that it trusts *most* even though it might not trust a's feeder much. The carrier should be *responsible* for b to trust a's feeder (otherwise b would not trust this carrier). By contrast, the current practice provides no guarantee for such a trust.

3.2 IDnet Mesh Framework

IDnet. In our framework, we generalized each identity provider as an *IDnet*, with the feeder and carrier being implemented as two typical cases of the IDnet. IDnets provide a common service to the public — *identity validation*, *i.e.*, to authenticate whether a user is accountable. Each IDnet consists of two basic components: *IDnet authority* and *IDnet agents*. The IDnet authority is the authority that administers an IDnet. It maintains a central database that stores users' authentication data. IDnet agents are designed to provide *high scalability* for the identity validation service via large scale replication. Each agent is propagated with a *hashed copy* of authentication data from the central database. We use the hashed copy instead of the original version of authentication data to ensure *security*. Each agent stores a different hashed copy to effectively localize security threats.

As shown in Figure 1(b), a feeder is implemented as an IDnet with only the IDnet authority, but no agents. While a carrier is implemented as a regular IDnet. Users register to a feeder and the feeder creates authentication data for them. The feeder then propagates hashed versions of authentication data to carriers. Each carrier is propagated with a different hashed copy. Such a design not only localizes security threats, but also makes users anonymous across carriers. Two different carriers are unable to infer whether two authentication data entries across their databases are associated with the same user.

Authentication data and Internet passport. In our design, each user should register to a feeder with his real identity. In return, the IDnet issues him a unique 160-bit *permanent identity* (PID) and a 160-bit *secret code* (SEC). Both data are stored in an *Internet passport*, which is a small and cheap device that can be plugged into the user's computer via a USB port. The Internet passport is designed to support strong but convenient user authentication. It uses a built-in clock to generate a time-changing *passcode* used for the authentication based on the SEC. The generation of the *passcode* is unlocked by the user's biometric property, *i.e.*, fingerprint. The Internet passport is designed to be tamper-resistant [17, 37] such that it effectively deters any attempts to steal the SEC. We explain and evaluate the implementation of the Internet passport in Sections 4 and 5, and show that its cost can be made around $10 or less, hence it can be widely used.

The PID and SEC constitutes a user's authentication data. A feeder propagates to carriers the hashed versions of PID and SEC, with which the carriers can perform identity validation for the user independently of the feeder.

Identity validation. A carrier organizes its IDnet agents in a tree structure and provides the identity validation service at the *edge agents* (*i.e.*, leaf nodes of the tree). As shown in Figure 1(b), each edge agent is propagated with a hashed copy of authentication data by following the tree structure and the data at each agent is associated with a different hash function sequence (*e.g.*, $h_7h_1(\cdot)$ for agent 7). In addition, the carrier issues each edge agent a pair of public and private keys. The edge agent announces to the public an *agent entry* which contains its public key and hash function sequence. In

practice, an edge agent can be a datacenter.

The identity validation process can be formulated by Equations (1)-(4). Below we introduce its main idea, and provide details later in Section 4.

$$HPID_i = H_i(PID), \quad HSEC_i = H_i(SEC) \tag{1}$$
$$passcode = P(HSEC_i, time) \tag{2}$$
$$TID = f(HPID_i, time, context, PubKey_i) \tag{3}$$
$$(HPID_i, time, context) = g(TID, PriKey_i) \tag{4}$$

H_i – the full hash function sequence at agent i, equivalent to a composite hash function. P – a cryptographic hash function. $time$ – the time provided by the Internet passport's built-in clock. f – a function to generate TID from $HPID_i$. g – a function to recover $HPID_i$ from TID. $PubKey_i$ and $PriKey_i$ – the public, private key pair of agent i. H_i and P are implemented based on SHA-1. f and g are implemented based on RSA.

First, the user chooses an edge agent (denoted by i) of a specific carrier and computes a *full hash function sequence*. To do this, he appends agent i's hash function sequence to the hash function that his feeder uses to propagate authentication data to the carrier, *e.g.*, in Figure 1(b), the full hash function sequence at agent 7 of the carrier for the feeder's users is $h_7h_1h_E(\cdot)$.

With the full hash function sequence, the user computes his hashed PID and SEC (denoted by $HPID_i$ and $HSEC_i$) stored at agent i (using Equation (1)). Then he generates a *passcode* via the Internet passport (using Equation (2)). After that, he computes a temporary identity TID based on $HPID_i$, the *time* same as the one used to generate *passcode*, a 160-bit service context, and the agent's public key (using Equation (3)).

Next, the TID and *passcode* are sent to agent i. From the TID, the agent recovers the user's $HPID_i$ (using Equation (4)), which in turn helps to retrieve the user's $HSEC_i$ (by querying the database). The agent also recovers the *time* from TID and checks its validity. In our implementation, valid *time* should differ no more than 30 seconds from the agent's system clock, which is *loosely* synchronized with the Internet passport's built-in clock. Then the agent verifies the *passcode* by regenerating it the same way as the user does (Equation (2)).

The identity validation process fully preserves a user's privacy to the public Internet. The user does not reveal his PID to the network and what others can see is just the TID, which makes him anonymous.

Forming the IDnet mesh. An IDnet mesh can be formed by gradually *merging* IDnets. The first type of merging is forming a *customer-provider* relationship as we have already seen between the feeder and carrier. Nevertheless, it can also happen between two carriers. In such cases, a customer IDnet propagates hashed authentication data to a provider IDnet. The second type of merging is forming a *peering* relationship. It only happens between two carriers. Both carriers propagate hashed authenticate data to each other under requests of feeders.

Since only hashed versions of data are propagated, we minimize the risks of merging. A system fault or a compromised agent that occurs in the other IDnet will not cause security threats on an IDnet's own infrastructure. This ensures a low trust threshold of merging, thereby facilitating the IDnet mesh to evolve towards global deployment. In addition, the merging requires no additional hardware deployment or software installation; only changes in software configuration are needed. Therefore, it also minimizes the technical complexity for the IDnet mesh to evolve.

IDnet forwarding. A feeder can ask a carrier C to further relay its hashed authentication data to other carriers that C peers with (if related service agreements allow this), instead of establishing direct customer-provider relationships with those carriers. We call such a relay an *IDnet forwarding*. In such cases, the full hash function sequence used in identity validation should also include a series of hash functions associated with IDnet forwardings. Each hash function associated with an IDnet forwarding is specified by the feeder. Therefore, hash functions used between the same pair of carriers for different feeders can be different.

3.3 IDnet Mesh's Trust Model

In this section, we explain the solution model for an underlying but fundamental question: *How can we trust an IDnet that we previously do not know?*, *i.e.*, the IDnet mesh's trust model.

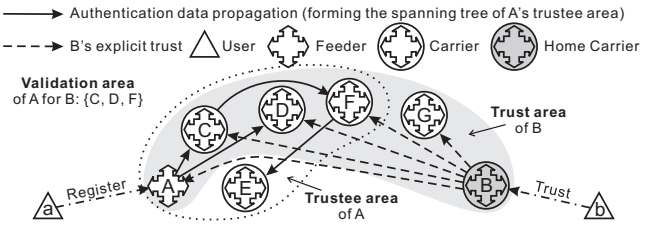


Figure 2: The trust model of IDnet mesh

Trustee area. Figure 2 depicts our entire trust model. First, we define the *trustee area* of a feeder. The trustee area of a feeder consists of all carriers that trust this feeder. For example, in Figure 2, the trustee area of A consists of carrier C, D, E, and F. These carriers trust A by allowing A to propagate its hashed authentication data to their central databases. The propagation structure can be represented by a spanning tree rooted at A to all carriers in the trustee area, *i.e.*, there is a unique propagation route from A to each carrier.

Trust area. Secondly, we define the *trust area* of a carrier. This area consists of all IDnets (both feeders and carriers) that this carrier trusts. The carrier *explicitly* expresses its trust by endorsing the public keys of these IDnets. In Figure 2, carrier B explicitly trusts feeder A and carriers C, D, F, and G thereby defining its trust area. The trust area is defined on a *per service* basis and therefore it specifies not only *who* to trust but *what* to trust as well. For example, a carrier can define very different trust areas for Web, Email, P2P,

and VPN services. In practice, a user will select a *home carrier* — the carrier that he trusts most. And he will trust all IDnets within the home carrier's trust area.

Validation area. Next, we define *validation area*, which is associated with a pair of feeder and carrier. Referring to Figure 2, the validation area *of A for B* is the overlapped area between A's trustee area and B's trust area, but with feeders excluded. This area consists of all carriers through which users (denoted by b) who select B as the home carrier can validate identities of feeder A's users. Users b admit the identity validation results because these carriers are within B's trust area. The identity validation for A's users can be performed because these carriers have imported the hashed copies of A's authentication data.

One exception is that if B's trust area does not includes feeder A, the validation area of A for B will be set to empty to indicate the distrust on the feeder. This reflects our design choice to prohibit implicit transitive trust that we previously mentioned as a key difference from the web of trust. Even if carriers within B's trust area trust A, it does not necessarily lead to that B also trusts A. B must explicitly express its trust on feeders. However, we may use transitive trust as an external mechanism to establish the explicit trust.

Validation agent. Finally, we define *validation agent*, which is associated with two users. Suppose that user a's feeder is A and user b's home carrier is B. A validation agent of a for b is defined as *any* IDnet edge agent of *any* carrier within the validation area of A for B. It is an edge agent through which user b can validate the identity of user a.

3.4 Services

The IDnet mesh provides two basic identity validation services as shown in Figure 3: *online validation* and *offline validation*. In both services, assume a common scenario: User b wants to validate user a's accountability, *i.e.*, to verify whether user a is a registered user (registered with the real identity) of a trustable feeder. To do this, they must first find a validation agent of a for b. We will describe how this validation agent is resolved in Section 5.4.1. Here, we simply assume there exists such a validation agent.

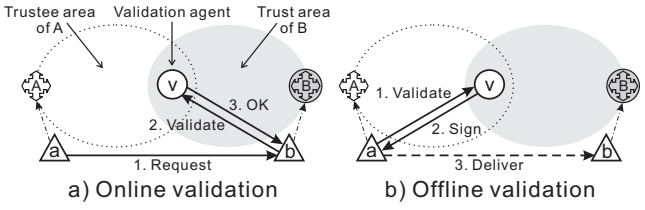


Figure 3: Two basic identity validation services

Online validation. In online validation (for applications such as Web), user a sends his validation data (TID and *passcode*) along with the service request to user b. Then b validates a's accountability via a validation agent v by relaying a's validation data. If the validation is successful, b accepts a's service request, otherwise not. For example, b could be a Web site and a could be one of its users; b can use online validation to protect itself from malicious users.

Offline validation. In offline validation (for applications such as Email), there is no online communication between a and b; a wants to deliver a data object to b, and b wants to validate the accountability of the object sender. To do this, a encodes the object's data fingerprint (using SHA-1) into the service context field (in Equation (3)) to generate the TID. Then a asks a validation agent v to validate TID and *passcode*. If the validation is successful, v returns a a digital signature that certifies the association between TID and the service context (decrypted from TID).

Next, a delivers the data object together with the signature, TID, and v's agent entry (defined in Section 3.2). b can then verify the sender's accountability by checking the consistency among the signature, the object's fingerprint, and the TID.

For example, b could be a user who wants to only read Emails from accountable users (such that he can effectively counter SPAMs). Then an Email user a can use the offline validation to show his accountability.

4. IMPLEMENTATION

Here, we provide details about our system implementation as shown in Figure 4. In particular, we describe the core IDnet identity validation algorithm (including the database implementation), as well as IDnet system and user protocols.

4.1 Core Algorithm

4.1.1 User Database Implementation

We implement the user database at an IDnet authority or agent using *MySQL*. The database includes a number of tables with the same structure. Each table stores up to 16 user blocks for a feeder and each user block stores up to 100,000 user entries. The name of each table is a 48-character string that encodes the feeder's IDnet identifier ($IDnet\_id$, 20 bytes) and the high 12 bits of block identifier ($block\_id$, 2 bytes). The IDnet identifier is a self-certifying flat name generated using SHA-1. Each user entry is a 3-tuple $\{HPID, HSEC, block\_id\}$. $HPID$ and $HSEC$ are the hashed version of a user's PID and SEC at this IDnet authority or agent.

4.1.2 Core Algorithm Implementation

Table 1 depicts the core identity validation algorithm both at the user and at the edge agent. We implement the algorithm in C++. We use the *Crypto++* [4] library for cryptographic functions such as SHA-1 and RSA. We choose *RSAES-OAEP* and *RSASSA-PSS* for RSA encryption and signature schemes respectively, both of which are recommended by RFC-3447 [15] for new applications in the interest of increased robustness.

4.2 IDnet Protocol

Each IDnet uses two types of protocols — *IDnet system protocol* and *IDnet user protocol*. The IDnet system

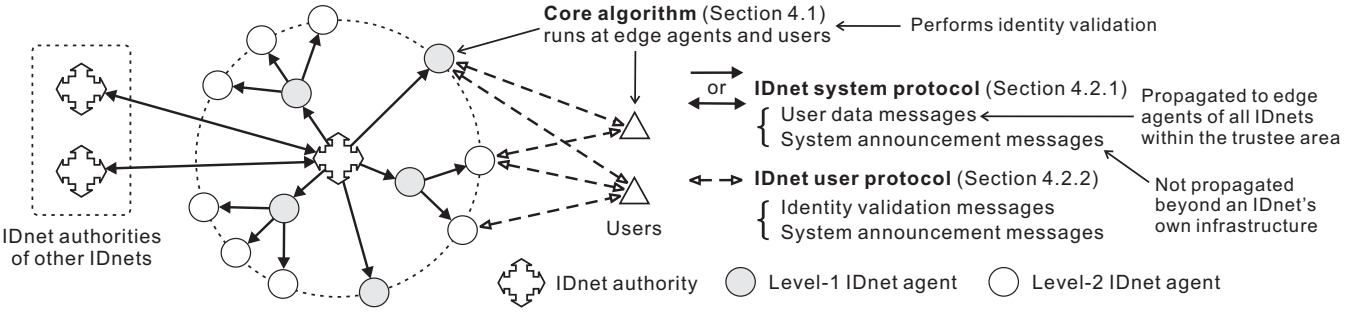


Figure 4: Implementation: Core algorithm, IDnet system protocol, and IDnet user protocol

$HSEC_i = H_i(SEC)$ (5) $nonce = time \mid additional\_nonce$ (6) $passcode = SHA(HSEC_i + nonce)$ (7) $HPID_i = H_i(PID)$ (8) $TID = RSA\_Encrypt(IDnet\_id \mid block\_id \mid HPID_i \mid context \mid time \mid additional\_nonce, PubKey_i)$ (9) $(IDnet\_id \mid block\_id \mid HPID_i \mid context \mid time \mid additional\_nonce) = RSA\_Decrypt(TID, PriKey_i)$ (10) $signature = RSA\_Sign(TID \mid context, PriKey_i)$ (11)	• "$\mid$" – the concatenation mark. • SHA – the SHA-1 hash function. • H_i – the full hash function sequence of agent i. Each hash function $h_k(x)$ in the sequence is defined by a 20-byte hash function id hid_k. It is defined as $h_k(x) = SHA(hid_k \ XOR \ x)$. The maximum length of the full hash function sequence is 14. • *additional_nonce* – an additional nonce used to counter TID replay attacks as we will describe in Section 5.5.4 • *time* – the time at the granularity of microseconds provided by the Internet passport's built-in clock. • *context* (20 bytes) – the service context. For offline validation, this is the SHA-1 fingerprint of the data object to deliver. • $PubKey_i$, $PriKey_i$ – the public and private keys of agent i.
0. The Internet passport stores PID (20 bytes), SEC (20 bytes), *block_id* (2 bytes), and $IDnet\_id$ (20 bytes). 1. The user chooses a suitable validation agent (denoted by i) as we will describe in Section 5.4.1. 2. The user inputs H_i (up to 281 bytes) and *additional_nonce* (12 bytes) into the Internet passport. In return, the Internet passport outputs the *passcode* (20 bytes), *time* (8 bytes), $IDnet\_id$, *block_id*, and PID. The *passcode* is generated using Equations (5)–(7). 3. The user generates TID using Equations (8) and (9). Then he sends TID and *passcode* to the validation agent. 4. Upon receiving the TID and *passcode*, the agent first decodes TID using Equation (10), which restores $IDnet\_id$, *block_id*, $HPID_i$, *context*, *time*, and *additional_nonce*. 5. The agent checks whether *time* differs less than 30 seconds from its own clock. If not, it returns failure. 6. The agent queries its user database to fetch the user's $HSEC_i$ based on $IDnet\_id$, *block_id*, and $HPID_i$. If the corresponding user entry is not found, it returns failure. 7. The agent regenerates the *passcode* the same way as the user does (Equation (7)) and checks whether it is the same as the *passcode* provided by the user. If not, it returns failure. 8. If this is an online validation, the validation is done and the agent returns success. 9. For offline validation, the agent generates a 128-byte digital signature using Equation (11). The signature certifies the association between the TID and *context*. The agent then returns the signature to the user.	

Table 1: Implementation of the core identity validation algorithm

protocol operates among an IDnet authority and agents of the same IDnet or between IDnet authorities of two different IDnets (as shown in Figure 4). The IDnet user protocol functions between IDnet edge agents and users (as shown in Figure 4). Both protocols are implemented upon TCP or UDP.

4.2.1 IDnet System Protocol Messages

IDnet system protocol messages are divided into two categories — *user data messages* and *system announcement messages*. The user data messages are designed to propagate hashed copies of user data from an IDnet authority to all its agents and to other IDnet authorities. The system announcement messages are designed to propagate *system announcements* (including information about agents, trust area, and trustee area) from an IDnet authority to all its own agents. Below we introduce the major IDnet system protocol messages.

A. User data messages

• *User entry update* consists of a list of user entries that need to be updated for a specified feeder. Each user entry contains the hashed version of a user's PID and SEC. The update initiates from the feeder and is later propagated to all IDnet agents within the feeder's trustee area.

We pace at one-hour intervals for the user entry updates initiated at a feeder. Each user entry update is ensured to be propagated to all IDnet agents in the trustee area within the next hour (even in the worst case). We will explain how this can be achieved in Section 5.3.1. This guarantees that any user data changes made at a feeder will take effect in the *entire* trustee area within two hours.

• *User entry sanity check* and *user entry sanity check response* are designed for maintenance purposes. They

help to check the consistency among user databases of different IDnet authorities and agents. They are the only system protocol messages that use UDP. All the others use TCP.

B. System announcement messages

• *Agent entry update* is designed to announce information about an IDnet's own agents. It contains an *agent entry*, which consists of the identifier, hash function sequence, and public key of an agent. In addition, it includes a signature block which certifies the entry. The signature block includes: (*i*) an SHA-1 fingerprint for the entry data, (*ii*) the inception date and expiration date of signature, (*iii*) the signer, which is the IDnet's identifier, and (*iv*) a 2048-bit RSA signature by the IDnet authority. The signature block is updated every day and expires after two days. An IDnet authority refreshes all agent entries every day. If no change happens to an agent's information (which is the common case), the only field in the update that needs to change is the signature block.

• *Trust area update* is designed to announce an IDnet's trust area definition. It includes a *trust area summary* and a list of *trust area entries*. The former is a short digest for the trust area definition and includes a signature block that certifies the entry. The latter lists all IDnets in the trust area. Each trust area entry corresponds to one IDnet. It consists of an IDnet identifier and a service type bitmap. Bit 0 of the service type bitmap indicates whether the specified IDnet is a feeder or a carrier. The rest bits define the types of services that the specified IDnet is trusted for. If all bits of the bitmap are set to zero, the specified IDnet will be revoked from the trust area. An IDnet authority propagates a trust area update to all its agents every day. The update is usually *incremental* — it only includes those IDnets whose information has been changed.

• *Trustee area update* is designed to announce an IDnet's trustee area definition. It includes a *trustee area summary* and a list of *trustee area entries*. Each trustee area entry corresponds to one IDnet. It consists of an IDnet identifier and a hash function sequence prefix. The hash function sequence prefix specifies the hash functions associated with IDnet forwardings. We can concatenate it with the hash function sequence of an agent (of the IDnet specified in the entry) to form the full hash function sequence. An IDnet authority propagates a trustee area update to all its agents every day. The update is usually incremental.

• *Endorsement update* and *endorsement signature update* are designed to announce and certify information about each IDnet in the trust and trustee areas. The latter is a compact version of the former. In the general case, an IDnet authority propagates daily an endorsement update, which includes IDnets whose information has been changed, and an endorsement signature update, which includes the remaining IDnets. The endorsement update consists of a list of *endorsement entries*, each of which includes the identifier, domain name, and public key of an IDnet. It also contains a signature block that certifies the entry.

All system announcements will be further propagated from IDnet edge agents to users via the IDnet user protocol. However, one subtlety is that if the IDnet is a feeder, it can not propagate system announcements to users by itself since it has no agents. To solve this, a feeder assigns a *delegated carrier*. It first propagates its system announcements to the delegated carrier and then have this carrier to propagate them to users. Meanwhile, a feeder's system announcements messages only include the trustee area update, endorsement update and endorsement signature update.

4.2.2 IDnet User Protocol

IDnet user protocol messages are divided into two categories — *identity validation messages* and *system announcement messages*.

The identity validation messages define the request and response format for online and offline validations. There is a *cookie* field in the *online validation request / response* messages. It can be used to encode the identifier and states associated with a service session. With the *cookie*, a service provider (*e.g.*, a Web site) does not have to maintain any state for the service session until the online validation completes.

The system announcement messages enable users to fetch and refresh system announcements from IDnet edge agents: (*i*) *Agent entry request / response* are designed for users to fetch and refresh the agent entry for the edge agent that the request is sent to. (*ii*) *Trust area summary request / response*, *trust area update request / response*, *trustee area summary request / response*, and *trustee area update request / response* are designed for users to obtain an IDnet's trust and trustee areas definitions. (*iii*) *Endorsement entry request / response* are designed for users to fetch and refresh the endorsement entry for a specified IDnet.

5. EVALUATION

We deployed our Linux-based implementation of core algorithm and protocols on a server-class test machine and the Emulab testbed [7]. We ran benchmarks for the core algorithm implementation on the test machine. We tested the functional integrity of the protocol implementation on the Emulab. For systematical performance evaluation, since it refers to a large-scale system that is hard to deploy (or emulate) on existing testbeds, we develop analytical models for the evaluation.

5.1 Internet Passport

The Internet passport helps to provide strong but convenient authentication. It exploits *two-factor authentication* [24] to support strong authentication. It unlocks itself using a user's biometric property (*e.g.*, to touch with a finger) instead of having a user type a PIN or a password such that it both supports security and is easy to use. To make our system scalable, the device should be available at a reasonably low cost. We can evaluate this by comparing its hardware com-

plexity with two types of related products — security tokens and biometric devices. Based on the comparison, we estimate that the cost of an Internet passport can be made around $10 or less.

Our Internet passport design shares similar features with security tokens such as *RSA SecurID* [17, 18] and *VASCO Digipass* [21], which are widely used for VPN, e-commerce and e-government applications. Their common design is to generate a time-changing passcode based on a built-in clock, a secret seed that is protected by the device's tamper-resistant feature [17, 37], and a cryptographic algorithm. The only major difference is that when generating the passcode, our algorithm also takes external parameters input from an Internet passport's USB port. However, this only slightly adds complexity to the hardware. The main complexity of this hardware lies in its cryptographic algorithm implementation and the tamper-resistant feature. Our cryptographic algorithm is based on SHA-1, which is even slightly simpler in its hardware implementation than the AES algorithm as used in the RSA SecurID. The price of an RSA SecurID is about $10 and that of a VASCO Digipass is about $7 as of 2005 [27].

As for the cost to support the biometric feature, we can refer to the cost of biometric devices nowadays: (*i*) a biometric USB flash drive can be purchased as cheaply as $7; (*ii*) a biometric optical mouse is at a price comparable to a regular optical mouse.

5.2 Core Algorithm

In this section, we evaluate the processing speed and scalability of our core algorithm at edge agent servers.

5.2.1 Processing Speed Benchmark

We first show benchmark results for processing speed of the core algorithm at edge agents. We perform the benchmark on a test machine with two dual-core 64-bit Intel Xeon 2.8 GHz processors. We set up a user database on this machine that consists of 4.8 million user entries. These entries are distributed across three full size tables, *i.e.*, each table has 16 user blocks and each block has 100,000 entries. We randomly select 10,000 entries from the database and precompute their TIDs and *passcode*s as the input for the benchmark.

Figure 5(b) shows the average processing time of online and offline validations for the 10,000 entries. It also itemizes the processing time of major steps that constitute the validation algorithms. For reference, we list micro-benchmark results on the same machine for basic cryptographic algorithms in Figure 5(a). As we can see, the processing speed of the core algorithm is mainly bounded by RSA operations — an RSA decryption operation in the online validation and an additional RSA signature operation in the offline validation.

Since RSA operations are CPU-bound, we can significantly improve the processing speed via multi-threading on a multi-processor machine. Figure 5(c) shows the processing time benchmark when multi-thread is used. As we can see, the processing time on this two-due-core-CPU machine converges quickly to $0.84\,ms$ for online validation and $1.56\,ms$ for offline validation as the number of threads increases, which is more than doubling the processing speed of a single thread.

Our benchmark result also reveals that if we can improve the RSA operation speed at edge agents by an order of magnitude (*e.g.*, using dedicated hardware [43]), the processing speed will no longer be bounded by RSA, but by the database query operations.

5.2.2 Service Scalability

Based on the benchmark results, we can gain understanding on the system's service scalability by assuming the following (aggressive) workload for all Internet users:

1 Assume each user on average accesses 100 Web pages that incur online validations every day.

2 Assume each user on average sends 20 Emails that incur offline validations every day.

3 Assume the workload at the peak time of a day is 10 times the average workload.

According to [9], there are 1,464 million Internet users in the world as of June 30, 2008. Therefore, to meet the peak time workload for all these users in the Web and Email service contexts, the system should be able to process 16.9 million online validations and 3.4 million offline validations every second. Using the benchmark results of the last section — $0.84\,ms$ for online validation and $1.56\,ms$ for offline validation, we get that each edge agent server can serve 75,000 users on average and we need only 19,520 servers totally to serve all the 1,464 million users of the current Internet.

For server load balancing, we can adopt the same approach as Google platform [32] does: (*i*) DNS servers resolve a common domain name (*e.g.*, *agent.IDnet-domain-name*) to a set of IP addresses, which acts as a first level of load balancing by directing users to different datacenters, *i.e.*, edge agents. The order of IP addresses provided by the DNS servers follows round-robin policy. (*ii*) A load-balancer (a proxy server) at each datacenter takes the user request and forwards it to one of the servers. This acts as a second level of load balancing.

5.3 IDnet System Protocol

The job of the IDnet system protocol is to *reliably* propagate user data and system announcements within the time constraint enforced by predefined *responsiveness upper bound*. The responsiveness upper bound quantifies the system's *guaranteed* responsiveness to data changes. It is defined as the time upper bound that outdated data could remain in the system *in the worst case*. The shorter the value, the better.

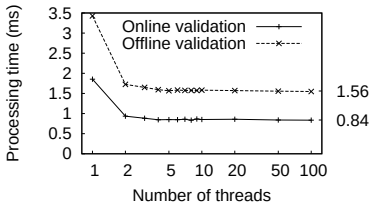
Short responsiveness upper bound and high reliability is critical for us to achieve *timely* updates and revocations for *trusted* information (such as the user credential, an agent's public key, an IDnet's trust and trustee area definitions). However, both the responsiveness upper bound and the reliability depend on a system's scale. The larger the system, the longer it takes to propagate data changes to the entire system

Algorithm	Time per operation	
	1024-bit	2048-bit
RSA encryption	0.10 ms	0.28 ms
RSA decryption	1.55 ms	8.13 ms
RSA signature	1.55 ms	8.13 ms
RSA verification	0.12 ms	0.32 ms
SHA-1	0.59 μs	

(a) Micro-benchmarks of cryptographic algorithms

Online validation	**1.85** ms
- Decrypt TID (RSA decryption)	1.55 ms
- Fetch $HSEC$ (database query)	0.26 ms
- Other program overhead	0.04 ms
Offline validation	**3.43** ms
- Online validation	1.85 ms
- Generate signature	1.58 ms

(b) Benchmark result



(c) Benchmark result (multi-thread)

Figure 5: Processing time benchmark for core algorithm

and the more complex the system is; therefore, it becomes more challenging to achieve short responsiveness upper bound and high reliability.

In this section, we evaluate the responsiveness upper bound and reliability of the IDnet system protocol using the topological model of a very large scale IDnet mesh system as described in Table 2.

5.3.1 Responsiveness Upper Bound

A. Responsiveness upper bound values

We first introduce our predefined responsiveness upper bound values for system data.

The responsiveness upper bound for user entries is two hours. As described in Section 4.2.1, we pace the user entry updates initiated by an IDnet at one-hour intervals. Each update is ensured to be propagated to all IDnet agents in the trustee area within the next hour. This implies that any changes in the user entries are guaranteed to take effect in the entire system within two hours, hence the two-hour responsiveness upper bound. Comparing with other Internet user credential approaches such as OpenPGP, our responsiveness upper bound is significantly shorter. OpenPGP's user credentials rely on the expiration time of digital certificates to invalidate themselves [28] in the worst case. The expiration time is typically set to one year, which implies a one-year responsiveness upper bound.

The responsiveness upper bound for system announcements (including agent entries, trust and trustee area summary, trust and trustee area entries, endorsement entries) is two days. As described in Section 4.2.1, we perform daily refreshment for signature blocks in all system announcements and set the signature blocks to expire after two days, which implies this two-day responsiveness upper bound. Comparing with similar secure global announcement approaches such as DNSSEC [6], our responsiveness upper bound is much shorter. In DNSSEC, the refreshment period and lifetime of signatures (for DNS data) are typically on the order of weeks or a month, thereby leading to a much longer responsiveness upper bound.

B. Bandwidth requirements

Here we evaluate the bandwidth requirements on related Internet paths in order to achieve the above responsiveness upper bounds. Denote by B the goodput to transmit the system protocol messages over an Internet path. We evaluate requirements on B and show that the requirements can be easily satisfied.

Denote by T_1 the time that it takes to propagate a message from an IDnet authority to all edge agent servers within the same IDnet; denote by S the message size. Using the topological model described in Table 2, we can get:

$$T_1 = 2 \times \frac{10S}{B} + 2D + d. \tag{12}$$

T_1 does not include the TCP connection establishment time. We assume that the TCP communication channels between an IDnet authority and a level-1 agent, and between a level-1 agent and a level-2 agent, are preestablished and kept alive all the time. Moreover, when propagating user entry updates, we need to perform an SHA-1 hash for each user entry. However, the hashing can be performed at line speed, hence is ignored here.[1]

Denote by T_2 the time that it takes to propagate a message from an IDnet authority to all edge agent servers of all IDnets within the trustee area. We can get:

$$T_2 = \frac{6S}{B} + 6D + T_1 = \frac{26S}{B} + 8D + d. \tag{13}$$

To achieve the two-hour responsiveness upper bound for user entries, we need to ensure that a user entry update can be propagated to all IDnet agents in the trustee area within one hour. Here we evaluate the minimum goodput B required to ensure this.

Assume the following scenario for a feeder with one million users: (i) The Internet passport for each user expires after three years (similar to a credit card); therefore each user needs to renew the Internet passport every three years. (ii) On average, each user loses track of his Internet passport once during the three years such that the user has to reclaim the Internet passport once. (iii) To be conservative, we assume that on average each user has his user entry updated 8 times for other possible reasons during the three years.

Based on the workload associated with the above scenario and the user entry update format that we implemented, we can get the message size S of each user entry update paced at one-hour intervals to be 16.4 KB. Letting $T_2 = 1\,hour$ and using Equation (13), we can get the minimum goodput $B = 0.12\,KBps$. This means that for a feeder with one million users and with the above workload for user entry updates, to guarantee the

[1] Suppose $B = 10\,MBps$, then the transmission time for each user entry is 4.4 μs. While the time to hash a user entry is only 0.3 μs.

Description of the model
1. The structure of each IDnet is a two-level complete 10-ary tree, that is, each IDnet has 10 level-1 agents and each level-1 agent has 10 level-2 agents. Therefore, each IDnet has 100 level-2 agents, which are its edge agents.
2. Each edge agent is a datacenter that consists of 10,000 servers. Therefore, each IDnet has 1 million edge agent servers.
3. Total number of IDnets in the IDnet mesh is 40,000.
4. An IDnet can propagate hashed versions of authentication data to other IDnets using IDnet forwardings through up to L intermediate (IDnet-level) hops. We set L to 6.
5. Denote by D the one-way propagation delay on Internet paths between two IDnet authorities, between an IDnet authority and each of its level-1 agent, or between a level-1 agent and each of its downstream level-2 agent. We set D to $500\,ms$.
6. Denote by d the total queuing delay at each edge agent to forward data to all its 10,000 servers. We set d to $1.5\,sec$.
Rationale for the model parameter settings
• We set item 1 and 2 by referring to the largest replica server system on the current Internet — Google platform [32]. The Google platform is estimated to have over 450,000 servers. These servers are distributed across tens of datacenters across the world. We set each IDnet in our model to have a comparable scale of the Google platform.
• We set item 3 by referring to the total number of *autonomous systems* (AS) on the current Internet because an IDnet and an AS share the similar administrative domain nature. There are about 40,000 AS numbers currently allocated by IANA. We therefore set the total number of IDnets in this model to 40,000.
• We set item 4 by referring to the *small world phenomenon* [36], which suggests a six degrees of separation between any two persons in the world. Since the separation between two identity providers should be less than that between two persons, we therefore set the separation upper bound L to 6.
• We set item 5 based on typical propagation delays on Internet paths. The typical propagation delay between two endpoints within the same continent ranges from several ms to several tens of ms; the typical propagation delay between two endpoints across different continents can span up to several hundreds of ms. We conservatively set D to $500\,ms$.
• For item 6, we assume a linear logical topology for the forwarding and conservatively assume the bandwidth between any two servers within a datacenter is $10\,MBps$. Suppose the size of each packet is 1,500 bytes, then the queuing delay of one packet is about $0.15\,ms$. Therefore, we get $d = 10,000 \times 0.15\,ms = 1.5\,sec$.

Table 2: Topological model of a very large scale IDnet mesh used to evaluate IDnet system protocol

two-hour responsiveness upper bound, we only need to ensure a goodput share of $0.12\,KBps$ on related Internet paths for user entry updates initiated by this feeder.

To achieve the two-day responsiveness upper bound for system announcements, we must ensure to propagate all system announcement updates within one day. To evaluate the minimum goodput B required to ensure this, assume an extreme case that the IDnet's trust and trustee areas include all the 40,000 IDnets. And consider the extreme case (everything is not incremental) for the daily system announcement updates volume: (i) 100 agent entries, (ii) a trust area update consisting of 40,000 trust area entries, (iii) a trustee area update consisting of 40,000 trustee area entries, and (iv) an endorsement update consisting of 40,000 endorsement entries. Based on formats of system announcement messages that we implemented, we can get $S = 42.7\,MB$. Letting $T_1 = 1\,day$ and using Equation (12), we can get the minimum goodput $B = 10.1\,KBps$.[2]

C. Signature generation time cost

To achieve the two-day responsiveness upper bound for system announcements, we must also ensure to refresh signature blocks in all system announcements daily. We can evaluate the signature generation time for this task using the above extreme case. Then, the number of signature blocks the IDnet needs to refresh daily is 40,102, including: (i) 100 for the agent entries of the 100 level-2 agents, (ii) 1 for the trust area summary and 1 for the trustee area summary, and (iii) 40,000 for endorsement entries corresponding to the 40,000 IDnets.

As shown in our micro-benchmark results in Figure 5(a), each RSA signature operation for 2048-bit keys takes $8.13\,ms$ on our test machine using a single thread. Therefore, we only need to dedicate $40,000 \times 8.13\,ms = 325.2\,sec$ CPU time daily to signature generation. If we use multi-threading, the signature generation time can be more than halved on the same machine.

5.3.2 Reliability

In addition to the bandwidth requirements and signature generation time as evaluated above, we also consider the following two reliability factors for the system protocol design: (i) possible connectivity failures on Internet paths, and (ii) possible IDnet system faults.

The current Internet only provides a best effort service which does not guarantee the connectivity. To ensure the timely propagation of protocol messages, we have to consider this factor in addition to the bandwidth requirements. According to [38], Internet path connectivity problems can usually be recovered within 20 minutes. We therefore expanded the guaranteed maximum propagation time to one hour to address this.

The IDnet system devices may experience software or hardware faults that impede the timely propagation of system announcements, which could affect the service availability. Therefore, it is particularly important to ensure a high reliability for the timely propagation of

[2]For the case when the IDnet is a feeder, we can get a similar result of $B = 9.9\,KBps$ by slightly changing the equation (to include an additional propagation hop from the feeder to the delegated carrier and to reflect that its system announcement messages can at most include a trustee area update and an endorsement update.)

system announcements. For this reason, we expanded the guaranteed maximum propagation time for system announcements to one day. This should be sufficient to recover system faults via automated failovers or manual technical support in most cases.

5.4 IDnet User Protocol

The evaluation of the IDnet user protocol depends on the service context where our system is applied (to provide user accountability). In this section, we use the Web and Email services as typical examples to evaluate the performance of the IDnet user protocol.

5.4.1 Application Examples and Time Overhead

The Web service is a typical example where online validation can be applied, while the Email service is a typical example where offline validation can be applied. Table 3 describes our prototype implementation for these two applications. Meanwhile, we analyze the corresponding time overhead when integrating the IDnet user protocol with these two services.

We evaluate the time overhead in terms of the measures RTT and D which are defined as follows:

RTT is the average round trip time on an Internet path between (i) a user and a local IDnet edge agent, (ii) a user and a local DNS, (iii) a user and a Web site, or (iv) a user and a server of an Email service provider. RTT is typically several ms to several hundred ms. D is the transmission delay for a trust area update response message (Section 4.2.2). It varies between several ms to several *sec* depending on the message size. The transmission delays for other IDnet user protocol messages are negligible compared to RTT.

Denote by C_d the delegated carrier (Section 4.2.1) of a user's feeder. The time overhead does not include the following operations from the user's perspective: (i) selecting an edge agent of C_d (this includes to resolve the agent via a local DNS, to download and to verify the agent entry), and (ii) downloading the feeder's trustee area update from the above agent of C_d and verifying it. Both operations are preprocessed automatically once a user's computer connects to the Internet.

The time overhead for transmitting any system announcement messages in the IDnet user protocol is *amortized across the whole day*. This is because the system announcements of an IDnet are updated at most once a day. Moreover, all the system announcements are very likely to remain static over longer time scales, which makes them good candidates for *caching*. Therefore, in the best case, what the daily updates (at the user's computer) actually do is simply refreshing the signature blocks and verifying that the cached system announcement data are still valid.

Below we summarize the time overhead incurred by identity validation in both the worst case and the best case according to Table 3. The best case results from the effective use of caching (*e.g.*, system announcement data are already cached and still valid).

Web: Online validation: The time overhead incurred by identity validation for the Web application is

Select an edge agent of an IDnet other than C_d	
• Worst case: 3 RTT (amortized across the whole day)	
1. Fetch the endorsement entry for the IDnet from C_d to get the IDnet's domain name and public key.	1 RTT
2. Resolve an edge agent of the IDnet via local DNS.	1 RTT
3. Fetch the agent entry from the edge agent.	1 RTT
4. Verify the integrity of the agent entry.	-
• Best case: 1 RTT (amortized across the whole day)	
1. Do the following in parallel:	1 RTT
1) Update the endorsement entry from C_d.	(1 RTT)
2) Update the agent entry from the edge agent.	(1 RTT)
Web: Online validation	
• Worst case: 5 RTT (2 RTT is amortized) - 2 RTT (the service request and response time when online validation is not used)	
1. Send a pre-service request to the Web site b. b responds with a list of (up to 20) preferred edge agents distributed across a number of preferred IDnets. Each entry in the list contains the agent's IP and the corresponding IDnet identifier.	1 RTT
2. Select a validation agent v based on the feeder's trustee area and the preferred agents of b.	-
3 Do the following in parallel:	1 RTT
1) Suppose v belongs to IDnet V. Fetch V's endorsement entry from C_d.	(1 RTT)
2) Fetch v's agent entry from v.	(1 RTT)
4. Verify the integrity of v's agent entry based on V's public key provided in the endorsement entry.	-
5. Generate TID and $passcode$. Then send a (TCP) service request to b together with the TID, $passcode$, and v's IP.	1.5 RTT
6. b relays the TID and $passcode$ to v in form of the online validation request and performs the online validation using v.	1 RTT
7. b responses to the service request based on the online validation result provided by v.	0.5 RTT
• Best case: 4 RTT (1 RTT is amortized) - 2 RTT	
1. Do the following in parallel:	1 RTT
1) Update V's endorsement entry from C_d.	(1 RTT)
2) Update v's agent entry from v.	(1 RTT)
2. Steps 5–7 of the worst case.	3 RTT
Email: Offline validation (sender side)	
• Worst case: 9 RTT+D (8 RTT+D is amortized)	
1. Resolve the trust area of the receiver.	5 RTT+D
1) Resolve the home carrier (B) of the receiver's Email provider (*e.g.*, Gmail, hotmail) via the Email provider's server or via DNS.	(1 RTT)
2) Select an edge agent of B based on B's domain name. (Need not fetch B's endorsement entry.)	(2 RTT)
3) Fetch the (TCP) trust area update of B from the above agent.	(2 RTT+D)
2. Compute the validation area as described in Section 3.3 and choose an IDnet (V) within the validation area.	-
3. Select an edge agent (v) of V.	3 RTT
4. Do offline validation via v and get v's signature.	1 RTT
5. Send the Email message together with the TID, v's signature, and v's agent entry. (Note: time cost for this step is not overhead.)	-
• Best case: 2 RTT (1 RTT is amortized)	
1. Do the following in parallel:	1 RTT
1) Verify that B is still the home carrier of the receiver's Email provider.	(1 RTT)
2) Update B's trust area summary from B.	(1 RTT)
3) Update V's endorsement entry from C_d.	(1 RTT)
4) Update v's agent entry from v.	(1 RTT)
2. Steps 4,5 of the worst case.	1 RTT
Email: Offline validation (receiver side)	
• Both worst case and best case: 1 RTT (amortized)	
1. Fetch V's endorsement entry from C_d.	1 RTT
2. Verify the integrity of the Email using the signature and the agent entry attached to the Email.	-

Table 3: Time overhead of IDnet user protocol

3 RTT in the worst case and 2 RTT in the best case. *In both cases, only 1 RTT of the overhead is incurred for every validation, the rest is amortized across the day.*

Email: Offline validation: The time overhead incurred by identity validation at the sender side is 9 RTT + D in the worst case and 2 RTT in the best case. *In both cases, only 1 RTT of the overhead is incurred for every validation, the rest is amortized across the day.* For the receiver side, the time overhead is 1 RTT for both cases and is amortized across the day.

5.4.2 Space Overhead

When using the offline validation for Email, a sender needs to attach the following data to an Email: TID (128 bytes), SHA-1 fingerprint (20 bytes) of the Email message, the signature (128 bytes) provided by the validation agent v, and the agent entry (570 bytes) of v. With Base64 encoding, these data result in $1.14\,KB$ space overhead per Email. To the best of our knowledge, the Email traffic composes 1∼1.5% [1] of total Internet traffic today and the average Email message size is of the order of tens of KB [8]. Therefore, the above space overhead for Email is reasonable.

5.5 Security

In this section, we evaluate several security concerns related to our work.

5.5.1 Impersonation Resiliency

Our system provides strong resiliency to user impersonation in the following way: (i) The tamper-resistant feature of a user's Internet passport ensures that others can not steal the SEC without being detected. The only way to get the SEC to impersonate the user is to get the Internet passport itself. (ii) Using a user's biometric property to unlock the generation of *passcode* ensures that even if others could get the Internet passport or hijack a user's computer, they won't be able to generate the *passcode* to impersonate the user. (iii) A user can easily revoke a missing Internet passport via the feeder by changing the SEC.

5.5.2 Surveillance Resiliency

A misbehaving provider may choose to spy on their clients, *e.g.*, collect user browsing patterns and sell to third parties for Internet advertising [23]. There are two issues with respect to this problem. First, user privacy is becoming a first-order issue nowadays (*e.g.*, [5]). With our feeder-carrier architecture, users can easily change carriers if they feel some carriers do not respect their privacy. This enables an effective business competition that enforces carriers to refrain from such surveillance activities. Meanwhile, feeders and collaborative carriers can easily react to a misbehaving carrier by opting it out from their trustee and trust areas.

Secondly, with the feeder and carrier decoupled, the IDnet mesh is *inherently* resilient to surveillance attempts. Neither a feeder or a carrier can effectively reverse engineer a client's identity: the feeder because it is not involved in the identity validation process, and the carrier because it is not the feeder.

5.5.3 Key Size

As of 2002, a key size of 1024 bits was generally considered the minimum necessary for the RSA algorithm. RSA claims that 1024-bit keys are secure (not likely to be "crackable") by 2010, while 2048-bit keys are sufficient until 2030 [20]. We use 2048-bit keys for the IDnet authority such that we provide high security for system announcement messages which are signed by such keys. We use 1024-bit keys for edge agents since our system can easily update edge agents' keys periodically through system announcement messages.

5.5.4 TID Replay Attacks

The *passcode* associated with each TID remains valid for up to 30 seconds. To prevent replay attacks using the same TID within this period, we can exploit the *additional_nonce* field used to generate the *passcode* (Equation (6) in Section 4.1.2). For example, an application can encode a server's IP and the service's TCP or UDP port to this field such that the *passcode* is valid only for the specified service on the specified server. The service process on this server caches all those TIDs that have passed identity validations in the recent 30 seconds such that it can block the replays. For online validation, this server could be a Web site server. For offline validation, this server could be the load balancer (the proxy server) at an edge agent.

5.5.5 Agent Spoofing Attacks

In online validation, a misbehaving user may spoof an edge agent's IP to send a fake online validation response to a server that he attempts to cheat. However, we can effectively counter such attacks by exploiting the online validation request / response's two-way communication property. The server can encode certain data only known by itself into the *cookie* field of the online validation request, such that only the edge agent who receives the request can provide a response with the same cookie. The server can therefore easily filter fake responses based on the cookie's validity.

5.5.6 DDoS Attacks

Malicious users could launch distributed denial-of-service (DDoS) attacks by sending a large number of identity validation requests to IDnet edge agents to deplete their CPU resources.

Our countermeasures to such attacks include the following: (i) The large scale identity validation service replication and load balancing mechanism used in the IDnet mesh provide the first level of resiliency to DDoS attacks. (ii) We can use CAPTCHA [35] (*e.g.*, challenge the user with a distorted image) or proof-of-work approaches [33] (*e.g.*, challenge the user's computer with a computational puzzle) to mitigate DDoS attacks when the attack level is high. (iii) We may deploy dedicated hardware for RSA operations [43] at edge agents to raise the agents' DDoS resiliency.

5.6 Incremental Deployability

The IDnet mesh requires no changes to the existing Internet infrastructure and protocols, and is there-

fore completely incrementally deployable. Each IDnet provider can gradually add servers to their system and use the Internet for wide-area system communication. All system secure communication channels are static and therefore can be easily implemented using cryptographic techniques such as IPsec tunnels. Such channels include those between the IDnet authority and a level-1 agent, between a level-1 agent and a level-2 agent, and between two IDnet authorities.

6. CONCLUSION

In this paper, we proposed IDnet mesh, a general purpose user identity architecture for the Internet. It can enable diversified new Internet services as well as new features for existing ones. It supports both user accountability and user privacy. It adopts a novel feeder-carrier decoupling approach which provides both high surveillance resiliency and high service scalability. It requires no change to the current Internet infrastructure and protocols, hence is completely incrementally deployable. It adopts a practical trust model such that it yields high system evolvability towards global deployment. Our evaluation shows that the proposed system can achieve outstanding performance for scalability, security, efficiency, and reliability at the same time.

7. REFERENCES

[1] http://blog.wired.com/27bstroke6/2008/04/ddos-packets-ar.html.
[2] Conn. bill would force MySpace age check. http://www.msnbc.msn.com/id/17502005/.
[3] Consumer watchdog overreacts about gmail. http://blogs.zdnet.com/Google/?p=1181.
[4] Crypto++ library. http://www.cryptopp.com/.
[5] Cuil. http://www.cuil.com/info/privacy/.
[6] DNSSEC: DNS security extensions. http://www.dnssec.net/.
[7] Emulab. http://www.emulab.net/.
[8] Google answers: What is the average size of an email message? http://answers.google.com/answers/threadview?id=312463.
[9] Internet world stats. http://www.internetworldstats.com/stats.htm.
[10] ITU-T Recommendation X.509: Information technology - Open systems interconnection - The directory: Public-key and attribute certificate frameworks.
[11] Kerberos. http://web.mit.edu/Kerberos/.
[12] MySpace to tighten security. http://www.knoxnews.com/news/2008/Jan/15/myspace-to-tighten-security/.
[13] New Yorker. http://www.unc.edu/depts/jomc/academics/dri/idog.html.
[14] OpenID. http://openid.net/.
[15] RFC 3447: Public-key cryptography standards (PKCS) #1: RSA cryptography specifications version 2.1. http://www.ietf.org/rfc/rfc3447.txt.
[16] RFC 4423: Host identity protocol (HIP) architecture. http://www.ietf.org/rfc/rfc4423.txt.
[17] RSA SecurID 6100 USB token. http://www.indevis.de/dokumente/rsa_securid_usb_token.pdf.
[18] RSA SecurID token. http://www.rsa.com/node.aspx?id=1156.
[19] SAML single sign-on (SSO) service for Google apps. http://code.google.com/apis/apps/sso/saml_reference_implementation.html.
[20] TWIRL and RSA key size. http://www.rsa.com/rsalabs/node.asp?id=2004.
[21] VASCO Digipass. http://www.vasco.com/products/Digipass.html.
[22] VeriSign unified authentication (white paper). http://www.verisign.com/static/016549.pdf.
[23] Watching while you surf. http://www.economist.com/science/tq/displaystory.cfm?story_id=11482452.
[24] What is two factor authentication? http://www.tech-faq.com/two-factor-authentication.shtml.
[25] What we should learn from Sony's fake blog fiasco. http://adage.com/smallagency/post?article_id=113945.
[26] Windows Live ID (Microsoft Passport). https://accountservices.passport.net/PPPrivacyStatement.srf.
[27] Data lockdown, Dec. 2005. http://money.cnn.com/magazines/fsb/fsb_archive/2005/12/01/8365397/index.htm.
[28] RFC 4880: OpenPGP message format. http://www.ietf.org/rfc/rfc2440.txt.
[29] Andersen, D. G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. Accountable Internet protocol (AIP). In *ACM SIGCOMM '08* (Seattle, WA, Aug. 2008).
[30] Bellare, M., Micciancio, D., and Warinschi, B. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT '03* (Warsaw, Poland, May 2003).
[31] Camenisch, J., and Herreweghen, E. V. Design and implementation of the idemix anonymous credential system. In *ACM CCS '02* (Washington, DC, USA, Nov. 2002).
[32] Carr, D. How Google works. *Baseline Magazine* (July 2006).
[33] chang Feng, W., chi Feng, W., and Luu, A. The design and implementation of network puzzles. In *IEEE INFOCOM '05* (Miami, FL, Mar. 2005).
[34] Ellison, C., and Schneier, B. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal 16*, 1 (2000), 1–7.
[35] Kandula, S., Katabi, D., Jacob, M., and Berger, A. Botz-4-sale: surviving organized DDoS attacks that mimic flash crowds. In *NSDI '05* (Berkeley, CA, May 2005).
[36] Kautz, H., Selman, B., and Shah, M. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM 40* (1997), 63–65.
[37] Kommerling, O., and Kuhn, M. G. Design principles for tamper-resistant smartcard processors. In *USENIX Workshop on Smartcard Technology* (Chicago, IL, May 1999).
[38] Labovitz, C., Ahuja, A., Bose, A., and Jahanian, F. Delayed Internet routing convergence. In *ACM SIGCOMM '00* (Stockholm, Sweden, Aug. 2000).
[39] Mislove, A., Post, A., Gummadi, K. P., and Druschel, P. Ostra: Leverging trust to thwart unwanted communication. In *NSDI '08* (San Francisco, CA, Apr. 2008).
[40] O'Reilly, T. What is Web 2.0. *O'Reilly Network* (Sept. 2005).
[41] Song, D., and Tsudik, G. Quasi-efficient revocation of group signatures. In *Financial Cryptography '02* (Southampton, Bermuda, Mar. 2002).
[42] Stallings, W. The PGP web of trust. *BYTE 20*, 2 (Feb. 1995), 161–162.
[43] Yesil, S., Ismailoglu, A. N., Tekmen, Y. C., and Askar, M. Two fast RSA implementations using high-radix montgomery algorithm. In *ISCAS (2)* (2004), pp. 557–560.