IDnet Mesh: A User Identity Solution for the Internet

Leiwen Deng and Aleksandar Kuzmanovic
Northwestern University

Abstract

This paper presents *IDnet mesh*, a general-purpose user identity solution 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. IDnet mesh uses tamper-resistant biometric-based hardware devices, called Internet passports, to achieve strong user accountability. At the same time, the system exploits cryptographic hash functions and RSA to fully preserve user privacy on the public Internet. Our system adopts a practical trust model such that it yields high system evolvability; it requires no changes 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 case of Email and Web services and demonstrate that IDnet mesh can be scalably integrated with these services, thereby improving their integrity.

1 Introduction

Problem. "On the Internet, nobody knows you're a dog," states Peter Steiner's famous New Yorker cartoon [12]. Fifteen years has passed since this cartoon was first published, and things have not changed in the meantime. 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 *accountability*.

In the context of Steiner's famous cartoon, the fundamental question we attempt to answer in this paper is the following: *Can we build a system that would guarantee that nobody on the public Internet knows that you are a dog, while making you accountable?* Indeed, we argue that in order to achieve this goal, it is essential not only to bring accountability to the end user's host (*e.g.*, [14, 24]), but to the *end user* itself. To motivate this approach, and to demonstrate limitations of the current user identity solutions, consider the following examples.

Examples. Email address is a typical example of user identity. By using a security solution such as OpenPGP [23], we can well verify the association between a user's Email address and the messages that she sends. However, even with OpenPGP, we have no effective way to counter SPAMs because the Email address is usually meaningless for people who do not know the user in advance. We can hardly tell whether an Email is sent from 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 accounts are another example of user identity. However, except for accounts of a small fraction of Web sites that require a user's real name information (*e.g.*, credit card information, bank account) for registration, the rest usually carry little meaning about a user's real identity. As a result, they are helpless to counter vandalizers or spammers. Vandalizers and spammers are posing significant threats to the rising Web 2.0 applications [33], which aim 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 [21].

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, 11]. MySpace acknowledged in the agreement that it would find and 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 solution for the Internet — *IDnet mesh*. This solution introduces a user identity representation that can provide *accountability* for a user's real identity. Such accountability serves as the key to address the aforementioned issues, *e.g.*, to counter Email or Web SPAMs, or to enable an effective adult check.

IDnet mesh supports both user privacy and user accountability. It supports *user privacy* by restricting the exposure of a user's real identity only to *one* third party,

i.e., home IDnet, that she trusts most, (*e.g.*, a bank, a university, a service- or a content-provider such as Google or MySpace). A user's real identity is fully hidden on the public Internet. It supports *user accountability* by making the user accountable to (and only to) its home IDnet provider of its own choice. Indeed, each user has the full control for the choice of this third party at any time, and the business competition among IDnets ensures that the third party must value the privacy demands from users [4].

The IDnet mesh supports the above features by providing a scalable common service to the public — *identity validation*, *i.e.*, to validate whether a user is accountable or not. It exploits large scale replication of authentication data to support high scalability for the service. It takes advantage of cryptographic hash functions to address security challenges for the replication. In addition, we design an inexpensive biometric user device, *Internet passport*, which enables strong user authentication, thereby significantly raising the security level.

Contributions. Our contributions are fourfold: (i) We design an effective user identity solution for the Internet that can support both strong user accountability and privacy. (ii) We propose a practical trust model that enables high system evolvability for the solution. (iii) We implement the algorithm and protocols for the solution and test their functional integrity on a testbed. (iv) We perform extensive evaluation of the solution and show that it can achieve outstanding performance for scalability, security, efficiency, reliability, and incremental deployability at the same time.

Regarding scalability, we show that the identity validation service workload associated with *all* Email and Web requests generated by the *entire* current Internet user population (about 1.5 billion users) could be handled by less than 20,000 IDnet mesh edge servers (*i.e.*, summed over all IDnet providers); the number could be further significantly reduced when we consider the fact that in many cases we only need to use the identity validation service to bootstrap user accountability.

The rest of this paper is organized as follows. In Section 2, we introduce our basic design of the IDnet mesh. In Section 3, we introduce the related work and compare it with our solution. 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 Basic Design

2.1 Local Identity Infrastructure — IDnet

We first introduce *IDnet*, the building block of our solution. Each *IDnet* is a local identity infrastructure administered by a single authority (a bank, a university, a service- or a content provider, *etc*). It provides a common service to the public — *identity validation*, *i.e.*, to validate whether a user is the IDnet's registered user.

Each user can register a unique identity at an IDnet that she trusts most by providing her real identity (real name, national ID number, driver license number, or passport number, *etc*). We call this IDnet the user's *home IDnet*. After registration, the home IDnet issues the (physically present) user an *Internet passport*, with which the user can access services that require identity validation. During the validation, the user being validated generates a temporary electronic identity TID using her Internet passport, and the user (or service) that validates uses the IDnet service to verify whether the TID is valid, *i.e.*, is associated with a registered user of the IDnet.

Each IDnet consists of two basic components: *IDnet authority* and *IDnet agents*. The IDnet authority is the authority that administers the IDnet. It maintains a central database that stores identity information for each registered user (including information about her Internet passport and real identity). IDnet agents are designed to provide *high scalability* for the identity validation service via large scale replication. Each IDnet agent replicates a *hashed copy* of Internet passport data (excluding the real identity information) from the central database. We use the hashed copy instead of the original version of user data to ensure *security*. Each agent stores a different hashed copy to effectively localize security threats, as we explain in more detail below.

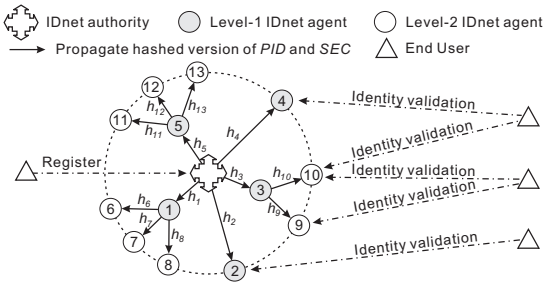


Figure 1: An IDnet with a two-level hierarchy

Internet passport. The IDnet issues each registered user 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 user authentication. It uses a built-in clock to generate a time-changing $passcode$ used for identity validation based on the SEC. In our design, a $passcode$ expires 30 seconds after being generated, as we explain in more detail below. The reading of a $passcode$ is unlocked via the user's biometric properties, *i.e.*, thumbprint. The Internet passport is designed to be tamper-resistant [15, 30] 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.

Identity validation. To make the identity validation service both *scalable* and *secure*, the IDnet authority propagates a hashed copy of each user's PID and SEC to IDnet agents. The propagation structure is a tree-like hierarchy as exemplified in Figure 1. The root node of the tree is the IDnet authority. The remaining nodes are IDnet agents. Each edge in the tree indicates a distinct propagation and each propagation uses a different hash function. The IDnet authority first propagates hashed copies to level-1 agents, which in turn propagate hashed copies to level-2 agents, and so on. For example, the IDnet authority first propagates a hashed copy to agent 1 using hash function h_1. Then agent 1 propagates a hashed copy to agent 6 using hash function h_6. As a result, the hashed copy of PID and SEC being stored at agent 6 becomes $h_6h_1(PID)$ and $h_6h_1(SEC)$.

Such design effectively localizes security threats. A compromised agent can at most affect the subtree that roots at it and has no effect on the remaining system. We discuss this issue further in Section 5.5. Note that private information (*e.g.*, name, driver licence or passport number) is never propagated to IDnet agents.

The identity validation service is provided at the IDnet's *edge agents* (*i.e.*, leaf nodes of the tree). For each edge agent, the IDnet authority issues it a pair of public and private keys. Each agent announces to the public an *agent entry* which contains its public key and hash function sequence (*e.g.*, $h_6h_1(\cdot)$ for agent 6). The agent entry is signed by the IDnet authority.

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

$$HPID_i = H_i(PID),\ \ HSEC_i = H_i(SEC) \quad (1)$$

$$passcode = P(HSEC_i, time) \quad (2)$$

$$TID = f(HPID_i, time, context, PubKey_i) \quad (3)$$

$$(HPID_i, time, context) = g(TID, PriKey_i) \quad (4)$$

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

Next, the TID and $passcode$ are sent to agent i (either directly from the end user or indirectly via relays). 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)).

User privacy and accountability. During the identity validation, the user does not reveal her PID and what others can see is just the TID. Equation (3) ensures that others are unable to distinguish whether two TIDs observed at two different times or places are associated with the same user, hence unable to infer the user's identity. In this way, the solution retains a user's *privacy*.

To support *accountability*, the home IDnet authority (*and only the home IDnet authority*) can resolve a user's real identity based on the TID and the agent used. To do this, it first recovers the user's $HPID_i$ from the TID (using Equation (4)). Then it resolves the user's real identity by looking up in a table that maps all users' PID to their $HPID_i$ at the agent.

2.2 Universal Identity Infrastructure — IDnet Mesh

A universal identity infrastructure can be formed by gradually merging IDnets, and we therefore name this universal infrastructure *IDnet mesh*. For example, several IDnets can merge together to form a small IDnet mesh; later on, several small IDnet meshes can merge together to form a more universal IDnet mesh.

High trust merging. The first way of merging is to simply merge the central databases of the two IDnets. This is applicable for cases that the two IDnet providers have strong trust with each other (*e.g.*, one company buys the other or two companies merge together thereby forming a new company under a single administration).

Low trust merging. The second way of merging is a more general case where the two IDnet providers bear little trust with each other but simply have a motivation to reuse each other's infrastructure. For such cases, they can merge by propagating to each other's central database a hashed copy of users' PIDs and SECs (real identities and other private information are never propagated beyond a home IDnet).

From the perspective of each IDnet authority, the other IDnet authority works essentially the same as one of its level-1 IDnet agents. This minimizes risks of the low trust 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.

A big picture of merging. Figure 2 exemplifies a big picture of merging in which seven IDnets belonging to two countries merge together and form a large ID-

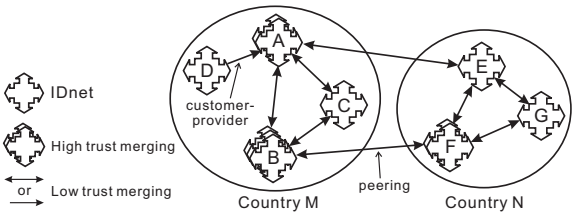


Figure 2: A big picture of merging

net mesh. IDnet A (or IDnet mesh A) results from high trust merging of two separate IDnets, thereby becoming equivalent to a single IDnet now. IDnet B is similar, resulting from high trust merging of three IDnets. IDnet C merges with both IDnet A and B via low trust merging, thereby forming peering relationship with them. There are also two pairs of IDnets across countries (A and E, B and F) forming peering relationships via low trust merging; high trust merging could be rare between IDnets of different countries for security or other reasons.

The merging between IDnet D and IDnet A is a special case of low trust merging, in which D propagates its hashed user data to A while A does not do the same to D. Indeed, this indicates a customer-provider relationship between them. D is a special IDnet that only has IDnet authority but no agents (*e.g.*, a university that maintains user accounts for all its students). D establishes a customer-provider service contract with A and propagates to A the hashed user data. For example, companies such as Google or Akamai, which already have large-scale infrastructures, could provide IDnet services as well. In this way, D can use A's infrastructure to provide wide-area identity validation service for its users.

IDnet forwarding. In the above scenario, D might also ask A to further relay its hashed user data to B, C and E if A's service agreements with B, C, and E allow this. In this way, D can also use B, C, and E's infrastructures such that D's identity validation service becomes more widely available (even across the country). We call such a relay *IDnet forwarding*.

2.3 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.

Mutual initial trust. The initial trust between a user and her home IDnet is established in a mutual way. The user trusts this IDnet most, therefore she selects it as her home IDnet. The home IDnet trusts the user, therefore it issues the user the Internet passport. This mutual initial trust serves as the starting point of the entire trust model.[1]

Trustee area. Figure 3 depicts our entire trust model. First, we define the *trustee area* of an IDnet. The trustee

[1]For example, trust established between Google and its clients who already store their Emails and other documents at Google's servers, is an example of such mutual initial trust.

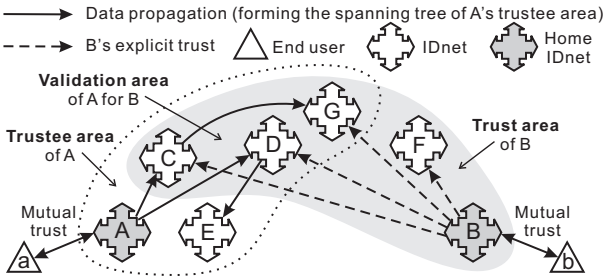


Figure 3: The trust model of IDnet mesh

area of an IDnet is the area that consists of all IDnets that trust this IDnet. For example, in Figure 3, the trustee area of IDnet A consists of IDnets C, D, E, and G. These IDnets trust A by allowing A to propagate its hashed user data to their databases. The propagation is either direct – via high trust or low trust merging, (*e.g.*, $A \rightarrow C$ and $A \rightarrow D$) or indirect – through IDnet forwarding (*e.g.*, $A \rightarrow C \rightarrow G$ and $A \rightarrow D \rightarrow E$). The propagation structure can be represented by a spanning tree rooted at A to all other IDnets in the trustee area, *i.e.*, there is a unique propagation route from A to each IDnet.

Trust area. Secondly, we define the *trust area* of an IDnet. The trust area of an IDnet is the area that consists of all IDnets that this IDnet trusts. It is quite different from the trustee area. The trust area is completely defined by each IDnet itself while the trustee area depends on other IDnets' will. The IDnet *explicitly* expresses its trust by endorsing the public keys of other IDnets. In Figure 3, IDnet B explicitly trusts IDnets C, D, F, and G, thereby defining its trust area. A 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, an IDnet can define very different trust areas for Web, Email, P2P, and VPN services.

Validation area. Next, we define *validation area*, which is associated with a pair of IDnets. Referring to Figure 3, the validation area *of A for B* is the overlapped area between A's trustee area and B's trust area. This area consists of all IDnets through which B's users can validate identities of A's users. B's users admit the identity validation results because these IDnets are within B's trust area. The identity validation for A's users can be performed because these IDnets have imported the hashed copy of A's user data.

2.4 Services

The IDnet mesh provides two basic identity validation services as shown in Figure 4: *online validation* and *offline validation*.

Validation agent. Before explaining the two services, we first introduce the concept of *validation agent*, which they will refer to. Suppose that user a's home IDnet is A and user b's home IDnet is B. A *validation agent* of a for b is defined as *any* IDnet agent of *any* IDnet within the validation area of A for B.

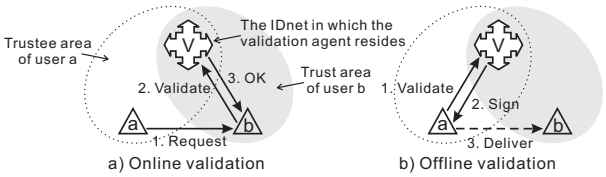


Figure 4: Two basic identity validation services

Online validation. In online validation (for applications such as Web), user a sends her validation data (TID and $passcode$) along with the service request to user b. Then b validates a's accountability via a validation agent 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 160-bit service context (as shown in Equation (3)) to generate the TID. Then a asks a validation agent to validate TID and $passcode$. If the validation is successful, the agent 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 the agent entry (defined in Section 2.1) of the validation agent. 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 she can effectively counter SPAMs). Then an Email user a can use the offline validation to show the accountability.

3 Related Work

Host accountability vs. user accountability. *Accountable Internet protocol* (AIP) [24] proposes a network architecture that provides accountability as a first-order property; *host identity protocol* [14] (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 solution can provide. For example, a host accountability solution is unable to verify if a user is adult or not [2, 11]. Indeed, the key to solving those problem cases 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 solution is.

Trust model comparison. The trust model of IDnet mesh shares a flavor of *web of trust* [34] in that both exploit a bottom-up trust propagation process, which is realistic in terms of the trust evolution nature. On the contrary, the *public key infrastructure* (PKI) [9] 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 "self-signed" root at a global scale impedes the PKI from evolving to a universal solution. Currently most PKI systems stay at enterprise scale.

The trust model of IDnet mesh differs from the web of trust in that it requires each IDnet to explicitly express its trust and prohibits implicit trust, *i.e.*, transitive trust. It therefore ensures *deterministic trust*. However, the transitive trust can be used as an external mechanism to establish an explicit trust. By contrast, the web of trust fundamentally depends upon the use of transitive trust to help trust propagation, which leads to uncertainty of trust. Such uncertainty significantly restricts the usefulness of the web of trust.

Finally, IDnet mesh's trust model is much more practical than social-networking based solutions (*e.g.*, [32]) because it removes the trust "burden" from individual users, and delegates this job to IDnet providers.

Kerberos and Ethane. The newest version of *Kerberos* [10] introduces a cross realm authentication feature to make it more feasible to scale to larger sets of networks. Our identity validation service is similar in spirit to the Kerberos cross realm authentication. However, since our solution is designed for a much larger network environment — the Internet, we focus on solving far more demanding scalability and security challenges.

Ethane [26] proposes an enterprise network architecture that exploits centralized control to facilitate system configuration and maintenance. In our solution, we adopt a similar centralized control design for each IDnet. We have the IDnet authority to propagate user data and system announcements to all agents in a centralized way.

4 Implementation

Here, we provide details about the system implementation. 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

Each IDnet authority or agent maintains a user database that stores both user data of its own IDnet and user data propagated from other IDnets. No private information is ever propagated among IDnets. Data of each user is represented by a user entry. 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. $block\_id$ is an identifer of *user*

block. We divide user data of each IDnet into large user blocks. Each block contains up to 100,000 user entries.[2] The $block\_id$ is 2-byte long. This means that each IDnet can have up to 64K blocks, which correspond to up to 6.5 billion users. This is about the number of the current world population.

The user database is implemented as a set of tables with the same structure in *MySQL* database. Each user entry corresponds to a row in a table. Each table stores up to 16 user blocks of an IDnet. Therefore, each table can hold up to 1.6 million user entries. The name of each table is a 48-character string that encodes both IDnet identifier and block identifier for the user data in the table. The first 5 characters are the prefix "IDnet". The rest 43 characters are the hexadecimal representation of the 20-byte IDnet identifier and the higher 12-bit of the block identifier. Each IDnet identifier is a self-certifying flat name generated using SHA-1.

4.1.2 Core Algorithm Implementation

Here, we introduce our implementation of the core algorithm — the identity validation algorithm. Figures 5(a) and 5(b) describe in detail the algorithm both at the end user and at the edge agent.

Our implementation code is written in C++. We use the *Crypto++* [3] library for cryptographic functions such as SHA-1 and RSA. We use *RSAES-OAEP* for the RSA encryption scheme and *RSASSA-PSS* for the RSA signature scheme, both of which are recommended by RFC-3447 [13] for new applications in the interest of increased robustness.

4.2 IDnet Protocol

The IDnet system has two types of protocols — *IDnet system protocol* and *IDnet user protocol*. The IDnet system protocol operates among an IDnet authority and agents of the same IDnet or between IDnet authorities of two different IDnets. The IDnet user protocol functions between IDnet edge agents and users.

Both protocols share the same general message format as shown in Figure 5(c) and are implemented upon TCP or UDP. Each message consists of a 2-byte message header and a variable length message body. The message header includes two fields: (i) $type$ code, which specifies the message type, and (ii) REQ bit, which indicates whether the message is a request. Figures 5(d) and 5(e) summarize all IDnet system protocol messages and user protocol messages, as we explain in more detail below.

[2]The key design reason for defining the user block is to bound the precomputation time cost for reverse mapping operations. When we want to reverse map the $HPID$ to its PID at the home IDnet, we need to precompute a table that maps each PID to its $HPID$. With user blocks, this precomputation only needs to be performed on a block's boundary, which takes less than 3 seconds based on benchmark results on our test machine. The precomputation time cost is amortized across all reverse mapping operations for $HPID$s in the same block and the mapping tables for frequently accessed blocks can be cached.

4.2.1 IDnet System Protocol

We define eight types of IDnet system protocol messages as shown in Figure 5(d), four of which are illustrated in detail in Figure 7. They are divided into two categories — *user data messages* and *system announcement messages*. The user data messages are designed to propagate hashed copy 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 (*e.g.*, information about agents, trust area, and trustee area) from an IDnet authority to all its agents.

A. User data messages

• *User entry update* consists of a list of user entries that need to be updated for an IDnet whose identifier is indicated by the field $IDnet\_id$. Each user entry contains the hashed version of a user's PID and SEC. The update initiates from the home IDnet authority and later propagates to all IDnet agents within the trustee area. The propagation paths are: (i) from an IDnet authority to other IDnet authorities, (ii) from an IDnet authority to all its level-1 agents, and (iii) from a level-1 agent to all its level-2 agents.

We pace the user entry updates initiated by an IDnet at one-hour intervals. Each user entry update is guaranteed to be propagated to all IDnet agents in the trustee area within the next hour. We will explain how this can be achieved in Section 5.3.1. This guarantees that any user data updates made at a home IDnet authority will take effect in the *whole* trustee area within two hours.

• *User entry sanity check* and *user entry sanity check response* are designed for maintenance purpose. They help to verify the consistency among user databases of different IDnet authorities and agents. They are the only system protocol messages that use UDP.

B. System announcement messages

• *Agent entry update* is designed to announce agent information. 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 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 updates agent entries every day. If no changes happen to an agent's information (which is the common case), only the signature block needs to be updated.

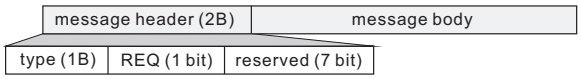
• *Trust area update* is designed to announce an IDnet's trust area definition. It includes a *trust area summary* and a list of *trust area entries*. The former is a short digest for the trust area definition. The latter lists all IDnets in the trust area. Each trust area entry corresponds to one IDnet. It consists of an IDnet identifier and a 256-bit service type bitmap. The service type bitmap defines types

User side algorithm
0. The Internet passport stores $IDnet\_id$ (20 bytes), PID (20 bytes), SEC (20 bytes), and $block\_id$ (2 bytes). • $IDnet\_id$ – the identifier of the user's home IDnet.
1. The user chooses a suitable agent (denoted by i) (Section 5.4.1).
2. The user inputs H_i (up to 300 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 computed using Equations (5)–(7). • "|" – the concatenation mark. • SHA – the SHA-1 hash function. • H_i – 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. $h_k(x)$ is defined as $SHA(hid_k\ XOR\ x)$. The maximum length of a hash function sequence is 15. • $additional\_nonce$ – 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.
3. The user's computer generates TID using Equations (8) and (9). • $context$ (20 bytes) – the service context. For offline validation, this is the SHA-1 fingerprint of the data object to deliver. • $PubKey_i$ (128 bytes) – the public key of agent i.
Agent side algorithm
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 $additional\_nonce$. • $PriKey_i$ – the private key of agent i.
5. The agent checks whether $time$ differs less than 30 seconds from its own clock. If not, it returns failure for the validation.
6. The agent queries its user database to fetch the user's $HSEC_i$ based on $IDnet\_id$, $block\_id$, and $HPID_i$. If user entry is not found, it returns failure for the validation.
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 agent returns success.
9. For offline validation, the agent generates a 20-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.

(a) Core algorithm description

$$HSEC_i = H_i(SEC) \tag{5}$$
$$nonce = time \mid additional\_nonce \tag{6}$$
$$passcode = SHA(HSEC_i + nonce) \tag{7}$$
$$HPID_i = H_i(PID) \tag{8}$$
$$TID = RSA\_Encrypt(IDnet\_id \mid block\_id \mid HPID_i \mid context \mid time \mid additional\_nonce, PubKey_i) \tag{9}$$
$$(IDnet\_id \mid block\_id \mid HPID_i \mid context \mid time \mid additional\_nonce) = RSA\_Decrypt(TID, PriKey_i) \tag{10}$$
$$signature = RSA\_Sign(TID \mid context, PriKey_i) \tag{11}$$

(b) Core algorithm equations



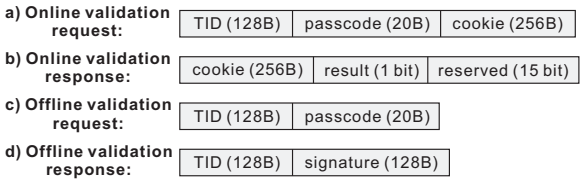
(c) General format of IDnet protocol messages

Protocol message name	type	REQ	Body size (bytes)
1. User data messages:			
User entry update	0h	0	$28 + 44N_{user}$
User entry sanity check	1h	1	42
User entry sanity check response	1h	0	28
2. System announcement messages:			
Agent entry update	10h	0	730
Trust area update	11h	0	$324 + 72N_{trust}$
Endorsement update	12h	0	$2 + 864N_{endorse}$
Endorsement signature update	13h	0	$2 + 320N_{endorse}$
Trustee area update	14h	0	$324 + 40N_{trustee}$

(d) IDnet system protocol messages

Protocol message name	type	REQ	Body size (bytes)
1. Identity validation messages:			
Online validation request	30h	1	404
Online validation response	30h	0	258
Offline validation request	31h	1	148
Offline validation response	31h	0	256
2. System announcement messages:			
Agent entry request	32h	1	4
Agent entry response	32h	0	734
Endorsement entry request	33h	1	24
Endorsement entry response	33h	0	870
Trust area summary request	34h	1	4
Trust area summary response	34h	0	326
Trustee area summary request	35h	1	4
Trustee area summary response	35h	0	326
Trust area list request (TCP)	36h	1	4
Trust area list response (TCP)	36h	0	$324 + 72N_{trust}$
Trustee area list request (TCP)	37h	1	4
Trustee area list response (TCP)	37h	0	$324 + 40N_{trustee}$

(e) IDnet user protocol messages



(f) Message bodies of identity validation messages

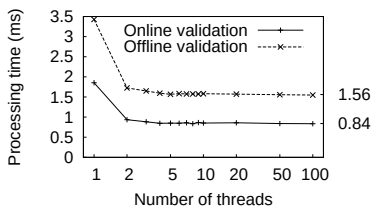
Figure 5: IDnet system implementation

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 6: Processing time benchmark for core algorithm

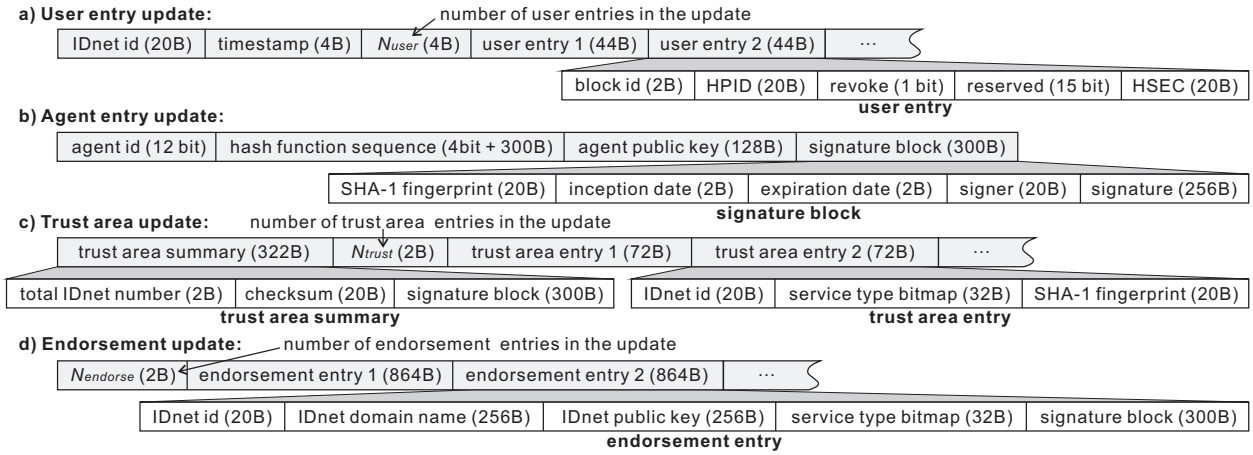


Figure 7: Message bodies of IDnet system protocol messages

of services that the specified IDnet is trusted for. If all bits of this 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.

• *Endorsement update* and *endorsement signature update* are designed to announce and certify information about each IDnet in the trust area. 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, public key, and service type bitmap of an IDnet. In addition, it contains a signature block that certifies the rest four fields. The signature block is updated every day and expires after two days.

• *Trustee area update* is designed to announce an IDnet's trustee area definition. Its format is very similar to that of the trust area update.

4.2.2 IDnet User Protocol

The IDnet user protocol messages are divided into two categories — *identity validation messages* and *system announcement messages* as shown in Figure 5(e). All messages except *trust area list request / response* and *trustee area list request / response* use UDP.

The identity validation messages define the request and response format for online and offline validations. Their formats are illustrated in Figure 5(f). The $cookie$ field in the *online validation request / response* can be used to encode identifier and states associated with the service session. With the cookie, a service provider (*e.g.*, a Web site) does not have to maintain any state for a service session until the validation completes.

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

5 Evaluation

We deployed our Linux-based implementation of core algorithm and protocols on a server-class test machine and the Emulab testbed [6]. We performed 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 plays an important role for the user side security. To make our solution scalable, this device should be available at a reasonably low cost. In this section, we evaluate the cost of the Internet passport by comparing its hardware complexity 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.

Security token products such as *RSA SecurID* [15, 16] and *VASCO Digipass* [18] exploit *two-factor authentication* [20] technology to support strong authentication. They are widely used for VPN, e-commerce, e-banking, and e-government applications. The token hardware can generate a one-time password based on a built-in clock, a token-specific secret seed, and a cryptographic algorithm such as AES, 3DES, or a proprietary hash algorithm. The hardware is designed to be tamper-resistant [15, 30] such that it effectively deters any attempts to steal the secret

seed. In addition to the one-time password, a second factor of authentication is provided by a PIN number or a paraphrase that the user knows. The price of an RSA SecurID is about $10 and that of a VASCO Digipass is about $7 as of 2005 [22].

Our Internet passport design shares very similar features with security tokens. It generates a time-changing passcode based on a built-in clock, the secret code SEC, and the cryptographic algorithm denoted by Equations (5)–(7) in Figure 5(b). The only major difference is that in order to generate the passcode, our algorithm also takes external parameters inputted from Internet passport's USB port. However, this difference 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 second factor of authentication in our Internet passport solution is provided by a user's biometric property, *i.e.*, thumbprint. We can use the user's biometric property to unlock the passcode reading. Such biometric authentication is both simpler and more secure than asking a user to type a PIN or paraphrase. The following are some references for 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 the benchmark results for processing speed of the core algorithm. We perform the benchmark on a test machine with two dual-core 64-bit Intel Xeon 2.8 GHz processors. We set up a database on this machine that consists of 4.8 million user entries. These entries are distributed in three full size tables, *i.e.*, each table contains 16 user blocks and each block has 100,000 entries. We randomly select 10,000 entries from the user database and precompute their TIDs and $passcode$s as the input for the benchmark.

Figure 6(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 online and offline validation algorithms. For reference, we list micro-benchmark results on the same machine for basic cryptographic algorithms in Figure 6(a). As we can see, the processing time of the identity validation at edge agents is mainly bounded by the RSA operations — an RSA decryption operation in the online validation and an additional RSA signature operation in the offline validation.

Since the RSA operations are CPU-bound, we can significantly improve the processing time of identity validation via multi-threading on a multi-processor machine. Figure 6(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 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 [35]), the identity validation algorithm 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 make a rough estimation for the number of edge agent servers needed in order to provide scalable identity validation services. According to [8], there are 1,464 million Internet users in the world as of June 30, 2008. Assume the following (aggressive) workload for these Internet users:

1 Each user on average accesses 100 Web pages that incur online validations every day.
2 Each user on average sends 20 Emails that incur offline validations every day.
3 The workload at peak time of the day is 10 times the average workload.

To meet the peak time workloads, the system should be able to process 16.9 million online validations and 3.4 million offline validations each second. Using the benchmark results in the last section — 0.84 ms for online validation and 1.56 ms for offline validation, we need 19,520 edge agent servers.

The above number can be reduced with proper application design, *i.e.*, for the way to use identity validations. As we will show in Section 5.4.2, we can dramatically reduce the workload of online and offline validations, thereby significantly reducing the number of servers needed for scalable services.

For server load balancing, we can adopt the same approach as the Google platform [25] does: (i) DNS servers resolve a domain name to multiple IP addresses, which acts as a first level of load balancing by directing users to different data centers (*i.e.*, the edge agents in our solution). The order of IP addresses provided by the DNS servers is done using round-robin policy. (ii) A load-balancer (a proxy server) at each data center 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 design goal 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 is the time upper bound that outdated data could remain in the system *in the worst case*. It quantifies the system's

guaranteed responsiveness to data changes. The shorter the responsiveness upper bound, the better.

However, both the responsiveness upper bound and the reliability depend on the system scale. The larger the system, the longer it takes to propagate the data and the more complex a system becomes; 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 using the topological model of a very large scale IDnet mesh as described in Table 1.

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.
2. Each level-2 agent is a data center that consists of 10,000 agent servers. Therefore, each IDnet has 1 million edge agent servers.
3. Total number of IDnets is 40,000.
4. An IDnet may propagate its hashed user data to another IDnet using *IDnet forwarding* (Section 2.2) via several intermediate IDnets hop by hop. Denote by L the maximum number of hops for such IDnet forwarding. We set L to 6.
5. Denote by D the one-way propagation delay of the Internet paths between two IDnet authority, 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.
Rationale for the model parameter settings
• We set item 1 and 2 by referring to the largest replica server system on the current Internet — the Google platform [25]. The Google platform is estimated to have over 450,000 servers. Such servers are distributed across tens of data centers in cities around 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 Internet since 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 our model to 40,000.
• We set item 4 based on the *small world phenomenon* [29], which suggests a six degrees of separation between any two persons.
• 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 on different continents may span up to several hundreds of ms. We conservatively set the one-way propagation delay D to 500 ms.

Table 1: Topological model of a very large scale IDnet mesh used to evaluate IDnet system protocol

5.3.1 Responsiveness Upper Bound

A. Message propagation time

Denote by T_1 the time that it takes to propagate a message from an IDnet authority to all agents within the same IDnet; denote by S the message size; denote by B the goodput to transmit the message (*i.e.*, the throughput for the TCP payload) over an Internet path; denote by d the total queuing delay at each level-2 agent, *i.e.*, a data center, to forward the message to all the 10,000 servers. Using the topological model described in Table 1, we can compute T_1 as:

$$T_1 = \frac{20S}{B} + 2D + d. \tag{12}$$

Here, $\frac{20S}{B}$ corresponds to the total transmission time, which is the time to sequentially send the message from the IDnet authority to the 10 level-1 agents plus the time to sequentially send the message from each level-1 agents to 10 downstream level-2 agents. $2D$ corresponds to the total propagation delay for the two levels of communication channels.

For the value of d, suppose we use a linear logical topology for the forwarding. Assume the size of each packet is 1,500 bytes, and the transmission bandwidth between two servers in the data center is 10 $MBps$. Then the queuing delay of one packet is about 0.15 ms. Therefore, d becomes $10,000 \times 0.15\, ms = 1.5\, sec$.

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 pre-established and kept alive all the time. Therefore, T_1 does not include the TCP connection establishment time. Moreover, when propagating user data messages, we need to perform an SHA-1 hash for each user entry during forwarding. However, the hashing can be performed at line speeds[3] and therefore Equation (12) does not need to include the time spent for hashing.

Denote by T_2 the time that it takes to propagate a message from an IDnet authority to all agents of all IDnets within the trustee area. Then, T_2 becomes:

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

Here $\frac{6S}{B}$ is the total transmission time for IDnet forwarding for up to 6 hops. $6D$ is the total propagation delay for IDnet forwarding channels.

B. Responsiveness upper bound for user entries

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 the responsiveness upper bound for user entries is two hours. Comparing with other Internet user credential solutions such as OpenPGP, our responsiveness upper bound is significantly shorter. OpenPGP's certificate for each user relies on the expiration time to invalidate itself [23]. The expiration time is typically set to one year, which implies a one-year responsiveness upper bound.

Here, we evaluate how we can guarantee the two-hour responsiveness upper bound in our solution. More specifically, what is the minimum goodput B required to ensure that a user entry update can be propagated to all IDnet agents in the trustee area within one hour?

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

Assume the following scenario for an IDnet 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 or her 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 or her user entry updated 8 times for other possible reasons during the three years.

The above scenario corresponds to a workload of 10 user entry changes per three years for each user on average, *i.e.*, 10 million user entry changes per three years in total. This is equivalent to 381 user entry changes per hour, which means that each user entry update paced at one-hour intervals will carry 381 user entries on average. According to Figures 5(c) and 5(d), we can compute the size of each user entry update as $S = 30+44N_{user}$ bytes with N_{user} set to 381.

Based on the representation of T_2 (Equation (13)), we can compute the minimum value of B as follows:

$$B = \frac{26S}{T_2 - 8D - d} = \frac{26 \times (30 + 44N_{user})}{T_2 - 8D - d}. \quad (14)$$

Letting $N_{user} = 381$, $T_2 = 3600\,sec$ (1 hour), $D = 0.5\,sec$, and $d = 1.5\,sec$, we get $B = 0.12\,KBps$. This means that for an IDnet with one million users and with the above workload for user entry updates, to guarantee the 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 IDnet.

C. Responsiveness upper bound for system announcements

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. This implies that the responsiveness upper bound for system announcements is two days. Comparing with similar secure global announcement solutions such as DNSSEC [5], 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 much longer responsiveness upper bound.

To guarantee the two-day responsiveness upper bound, we must ensure that we can generate new signatures daily for all system announcements and to propagate system announcement updates within one day. Below, we evaluate these two aspects in terms of the time dedicated to generate the daily signatures and the minimum goodput B required to ensure the timely propagation of system announcement updates.

Assume an extreme case that the trust area of a specified IDnet contains all the 40,000 IDnets. Then, the number of signature blocks the IDnet needs to update 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 6(a), each RSA signature operation for 2048-bit keys takes $8.13\,ms$ on the 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 speed can be more than doubled on the same machine.

To estimate the minimum goodput B, we consider an extreme case for the daily system announcement updates volume: (i) 100 agent entries, (ii) a trust area update consisting of 40,000 trust area entries, (iii) an endorsement update consisting of 40,000 endorsement entries, and (iv) a trustee area update consisting of 40,000 trustee area entries. According to Figures 5(c) and 5(d), we can compute the total size of these messages as: $S = (100 \times 732) + (326 + 72N_{trust}) + (4 + 864N_{endorse}) + (326 + 40N_{trustee})$ bytes. Letting $N_{trust} = N_{endorse} = N_{trustee} = 40,000$, we get $S = 37.3\,MB$.

Based on the representation of T_1 (Equation (12)), we can compute the minimum goodput B required to propagate these messages from an IDnet authority to all agents in the same IDnet within one day as follows:

$$B = \frac{20S}{T_1 - 2D - d}. \quad (15)$$

Letting $S = 37.3\,MB$, $T_1 = 86400\,sec$ (1 day), $D = 0.5\,sec$, and $d = 1.5\,sec$, we get $B = 8.84\,KBps$.

5.3.2 Reliability

As a basic infrastructure that other Internet services rely on, the IDnet mesh must ensure high system reliability. In addition to the bandwidth requirements and signature generation speed 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. Therefore, to ensure the timely propagation of protocol messages, we have to consider this factor in addition to the bandwidth requirements. According to [31], Internet path connectivity problems can usually be recovered within 20 minutes. We therefore set the guaranteed maximum propagation time to no less than 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 impact IDnet service availability. Therefore, it is particularly important to ensure a high reliability for the timely propagation of system announcements. For this reason, we set 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 IDnet user protocol depends on the concrete service context. 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 Overhead

A. 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 2 illustrates our prototype implementations for these two applications. In addition, it also analyzes the time overhead of IDnet user protocol for the two services.

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

The time overhead does not include the following operations from the user's perspective: (i) selecting an edge agent of the home IDnet (including resolving the agent via a local DNS, downloading and verifying the agent entry), and (ii) downloading the trust area list and trustee area list of the home IDnet from the selected edge agent. Both operations are preprocessed automatically after a user's computer connects to the Internet.

The time overhead for all system announcement messages in IDnet user protocol is *amortized across a whole day* because the system announcements of each IDnet are updated at most once per day. In addition, 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 in both the worst case and the best case (system announcement data are already cached and still valid) according to Table 2.

Web: Online validation: The time it takes to finish a Web service request and response is 5 RTT in the worst case and 4 RTT in the best case. Deducting 2 RTT as the service request and response time when online validation is not used, the time overhead is therefore 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 whole day.*

Email: Offline validation: The time overhead at the sender is 10 RTT+D in the worst case and 2 RTT in the best case. *In both cases, only 1 RTT of the overhead is*

Select an edge agent of an IDnet other than the home IDnet	
• Worst case: 3 RTT (amortized across the whole day)	
1. Fetch the endorsement entry for the IDnet from the home IDnet agent to get the IDnet's domain name and public key.	1 RTT
2. Access the local DNS to resolve an edge agent of the IDnet.	1 RTT
3. Fetch the agent entry from the edge agent.	1 RTT
4. Verify the agent entry based on the IDnet's public key.	-
• Best case: 1 RTT (amortized across the whole day)	
1. Do the following two in parallel:	1 RTT
1) Update the endorsement entry.	(1 RTT)
2) Update the agent entry from the edge agent.	(1 RTT)
Select an edge agent of the home IDnet (not included in time overhead)	
• Worst case: 2 RTT (amortized across the whole day)	
1. Steps 2–4 of the worst case in "select an edge agent of an IDnet other than the home IDnet." Step 1 is excluded because the user knows the home IDnet's domain name and public key in advance.	2 RTT
• Best case: 1 RTT (amortized across the whole day)	
1. Update the agent entry from the edge agent.	1 RTT
Web: Online validation	
• Worst case: 5 RTT (2 RTT is amortized across the whole day) - 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 within b's trust area. Each entry in the list contains the IP and the IDnet identifier of an agent.	1 RTT
2. Select a validation agent v based on the trustee area of the user and the preferred agents of b.	-
3 Do the following two in parallel:	1 RTT
1) Suppose v belongs to IDnet V. Fetch V's endorsement entry from the home IDnet agent.	(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 TID, $passcode$, and v's IP.	1.5 RTT
6. b relays 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 across the whole day) - 2 RTT	
1. Do the following two in parallel:	1 RTT
1) Update V's endorsement entry.	(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: 10 RTT+D (9 RTT+D is amortized across the whole day)	
1. Resolve the trust area of receiver.	6 RTT+D
1) Resolve the home IDnet (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.	(3 RTT)
3) Fetch the (TCP) trust area list of B from the above agent.	(2 RTT+D)
2. Compute the validation area as described in Section 2.3 and choose an IDnet (V) within the validation area.	-
3. Select an edge agent (v) of V.	3 RTT
4. Do offline validation using v and get the signature.	1 RTT
5. Send the Email together with TID, the signature, and the agent entry of v. (P.S. Time cost for this step is not overhead.)	-
• Best case: 2 RTT (1 RTT is amortized across the whole day)	
1. Do the following three in parallel:	1 RTT
1) Update B's trust area summary.	(1 RTT)
2) Update V's endorsement entry.	(1 RTT)
3) 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 across the whole day)	
1. Fetch V's endorsement entry from the home IDnet agent.	1 RTT
2. Verify the signature embedded in the Email.	-

Table 2: Time overhead analysis of IDnet user protocol

incurred for every validation, the rest is amortized across the whole day. The time overhead at the receiver is 1 RTT for both cases and is amortized across the whole day.

B. Space overhead

When using offline validation for Email, the 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 edge agent, and the agent entry (730 bytes) of the edge agent. Using Base64 encoding, it results in $1.35\,KB$ space overhead per Email. To the best of our knowledge, the Email traffic composes 1$\sim$1.5% [1] of total Internet traffic today and the average Email message size is of the order of tens of kilobytes [7]. Therefore, the above space overhead for Email is quite small.

5.4.2 Application Design Guideline

We can optimize application performance using the following design guideline — "use identity validation only to bootstrap user accountability." (This is quite similar to the practice that people "use RSA cryptography only to bootstrap a security association.") When user accountability is established via identity validation, we can use faster approaches to retain the user accountability instead of performing identity validations repeatedly. With this guideline, we can (i) significantly improve the application efficiency by reducing the overhead incurred by identity validations, and (ii) dramatically reduce the identity validations' workload, thereby greatly enhancing the IDnet mesh's service scalability.

For example, for Web sites that require a user to register an account, we may use the online validation *only* for the registration, and then refresh the validation over longer time scales, *e.g.*, once a week. We bind TID and the agent entry to the user's registration information such that we retain the user accountability for the registered account. Later, the user simply logins with her account to access the Web site the same way as she does today and the user accountability is automatically enabled.

Necessarily, when a user's machine can not guarantee to be spyware free, there is a tradeoff between efficiency and user impersonation resiliency (which we discuss in more detail in Section 5.5.1) when we apply this guideline. Indeed, it depends on applications to decide the balance between the two. For instance, for insensitive applications such as Internet forums, favoring efficiency could be a good choice; at the same time, for sensitive applications such as VPN or e-commerce, we are better off favoring impersonation resiliency by requiring IDnet-based verification at all times. In addition, benefiting from the user accountability enabled by our solution, we can easily inform the owner when detecting user impersonation, such that the owner can quickly react to it.

5.5 Security

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

5.5.1 Impersonation Resiliency

Our solution provides strong resiliency to user impersonation in the following way: (i) The tamper-resistant feature of 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 reading of $passcode$ ensures that even if others get the Internet passport or hijack a user's computer, they won't be able to generate the $passcode$ to impersonate a user. (iii) A user can easily revoke a missing Internet passport via the home IDnet by changing the SEC.

5.5.2 Surveillance Resiliency

A misbehaving IDnet may choose to spy on their clients, *e.g.*, collect user browsing patterns and sell to third parties for Internet advertising [19]. There are two issues with respect to this problem. First, user privacy is becoming a first-order issue nowadays (*e.g.*, [4]). The business competition enforces that each IDnet provider must refrain from such activities. Otherwise, it faces high risks to undermine its business reputation and in turn lose both customers and collaborative IDnets (*e.g.*, other IDnets would remove this IDnet from their trust areas and opt themselves out from this IDnet's trustee area).

Second, IDnet mesh is *inherently* resilient to surveillance attempts. This is because there is *no* single third-party in our system, and hence no single IDnet can effectively surveil clients. In particular, when a client from home IDnet A validates itself at an edge server belonging to IDnet provider B (common case), neither of the two IDnets can reverse engineer a client's identity: IDnet A because it is not involved in the validation process, and IDnet B because it is not the home IDnet.

5.5.3 Key Size

As of 2002, a key size of 1024 bits was generally considered the minimum necessary for the RSA encryption algorithm. RSA claims that 1024-bit keys are secure (not likely to become 'crackable') by 2010, while 2048-bit keys are sufficient until 2030 [17]. We use 2048-bit keys for the IDnet authority such that they can remain unchanged for a long time period. We use 1024-bit keys for the edge agents since our system can easily change agents' keys over relatively short time period (*e.g.*, once every three months).

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)). For example, an application can encode a server's IP and the service TCP/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 the TIDs that have passed validations in the recent 30 seconds such that it can block the re-

plays. 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 the 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 she 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 (Figure 5(f)), such that only the agent who receives the request can provide a response with the same $cookie$. The server can therefore easily filter fake responses.

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 user data replica and load balancing mechanism used in IDnet mesh provide the first level of resilience to DDoS attacks. (ii) We can use CAPTCHA [28] (*e.g.,* challenge the user with a distorted image) or proof-of-work approaches [27] (*e.g.,* challenge the user's computer with a computational puzzle) to mitigate DDoS attacks when the attack level is high. (iii) We may use dedicated hardware for RSA operations [35] at edge agents to raise agents' DDoS resilience.

5.6 Incremental Deployability

The IDnet mesh requires no changes to the existing Internet infrastructure and protocols, and is therefore 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 solution for the Internet. The solution can enable diversified new Internet services as well as new features for existing ones. It can support both strong user accountability and privacy. It requires no changes to the current Internet infrastructure and protocols, and thereby is completely incrementally deployable. It adopts a practical trust model such that it yields high system evolvability. Our evaluation shows that the proposed system can achieve outstanding performance for scalability, security, efficiency, and reliability at the same time.

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] Crypto++ library. http://www.cryptopp.com/.

[4] Cuil. http://www.cuil.com/info/privacy/.

[5] DNSSEC: DNS security extensions. http://www.dnssec.net/.

[6] Emulab. http://www.emulab.net/.

[7] Google answers: What is the average size of an email message? http://answers.google.com/answers/threadview?id=312463.

[8] Internet world stats. http://www.internetworldstats.com/stats.htm.

[9] ITU-T Recommendation X.509: Information technology - Open systems interconnection - The directory: Public-key and attribute certificate frameworks.

[10] Kerberos. http://web.mit.edu/Kerberos/.

[11] MySpace to tighten security. http://www.knoxnews.com/news/2008/Jan/15/myspace-to-tighten-security/.

[12] New Yorker. http://www.unc.edu/depts/jomc/academics/dri/idog.html.

[13] RFC 3447: PKCS #1: RSA cryptography specifications version 2.1. http://www.ietf.org/rfc/rfc3447.txt.

[14] RFC 4423: Host identity protocol (HIP) architecture. http://www.ietf.org/rfc/rfc4423.txt.

[15] RSA SecurID 6100 USB token. http://www.indevis.de/dokumente/rsa_securid_usb_token.pdf.

[16] RSA SecurID token. http://www.rsa.com/node.aspx?id=1156.

[17] TWIRL and RSA key size. http://www.rsa.com/rsalabs/node.asp?id=2004.

[18] VASCO Digipass. http://www.vasco.com/products/Digipass.html.

[19] Watching while you surf. http://www.economist.com/science/tq/displaystory.cfm?story_id=11482452.

[20] What is two factor authentication? http://www.tech-faq.com/two-factor-authentication.shtml.

[21] What we should learn from Sony's fake blog fiasco. http://adage.com/smallagency/post?article_id=113945.

[22] Data lockdown, Dec. 2005. http://money.cnn.com/magazines/fsb/fsb_archive/2005/12/01/8365397/index.htm.

[23] RFC 4880: OpenPGP message format.

[24] 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).

[25] CARR, D. How Google works. *Baseline Magazine* (July 2006).

[26] CASADO, M., FREEDMAN, M. J., PETTIT, J., LUO, J., MCKEOWN, N., AND SHENKER, S. Ethane: taking control of the enterprise. *ACM SIGCOMM Compute Communication Review 37*, 4 (2007), 1–12.

[27] CHANG FENG, W., CHI FENG, W., AND LUU, A. The design and implementation of network puzzles. In *IEEE INFOCOM '05* (Miami, FL, Mar. 2005).

[28] 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).

[29] KAUTZ, H., SELMAN, B., AND SHAH, M. ReferralWeb: Combining social networks and collaborative filtering. *Communications of the ACM 40* (1997), 63–65.

[30] KOMMERLING, O., AND KUHN, M. G. Design principles for tamper-resistant smartcard processors. In *USENIX Workshop on Smartcard Technology* (Chicago, IL, May 1999).

[31] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed Internet routing convergence. In *ACM SIGCOMM '00* (Stockholm, Sweden, Aug. 2000).

[32] 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).

[33] O'REILLY, T. What is Web 2.0. *O'Reilly Network* (Sept. 2005).

[34] STALLINGS, W. The PGP web of trust. *BYTE 20*, 2 (Feb. 1995), 161–162.

[35] 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.