NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-09-11
April 20, 2009

Towards Tussle-based Rules in Cyberspace

Leiwen Deng and Aleksandar Kuzmanovic

Abstract

How we can regulate users in cyberspace is a mixed question of both the technical domain and the non-technical domain. In this paper, we perform an in-depth study on the way to *decouple* the technical part of this question from its non-technical part such as "rules" (e.g., laws, regulations, and the like) and "tussles" (among people's conflicted interests). In addition, for the technical part, we propose the *feeder-carrier-mapper model* — an architecture that can *accommodate* controversial "rules" and the underlying "tussles".

We also provide an implementation of the feeder-carrier-mapper model to demonstrate its feasibility. We propose candidate algorithms that can meet our central design criteria and explain important design subtleties of this model.

Keywords: IDnet Mesh, feeder, carrier, mapper, publicized rules, tussle

Towards Tussle-based Rules in Cyberspace

Leiwen Deng and Aleksandar Kuzmanovic

ABSTRACT

How we can regulate users in cyberspace is a mixed question of both the technical domain and the non-technical domain. In this paper, we perform an in-depth study on the way to *decouple* the technical part of this question from its non-technical part such as "rules" (*e.g.*, laws, regulations, and the like) and "tussles" (among people's conflicted interests). In addition, for the technical part, we propose the *feeder-carrier-mapper model* — an architecture that can *accommodate* controversial "rules" and the underlying "tussles".

We also provide an implementation of the feeder-carrier-mapper model to demonstrate its feasibility. We propose candidate algorithms that can meet our central design criteria and explain important design subtleties of this model.

1. INTRODUCTION

Difficulty of rules. The Internet is moving towards the mainstream of the societies due to its great success. However, this tendency is also inevitably forcing the Internet to inherit the very complexity of the societies. One typical case is the complexity to manage the order in a society. In the real space, human beings rely on laws, regulations, or the like to do it. We call them "publicized rules" for generality (or simply "rules" when there is no ambiguity). However, in the cyberspace, the publicized rules face significant difficulties to be applied. We explain this as follows.

Rules call for user accountability, while the Internet calls for user anonymity. Publicized rules require the feasibility to react to a liable user, otherwise they are helpless since they are not enforceable. This implies the accountability on a user's identity. However, the Internet hides a user's identity by design. This design principle is indispensable not only due to the security requirement on user privacy, but also because this principle is essential to the Internet's great success, *e.g.*, censorship-free speech and ideas, an extraordinary thriving of creativity, *etc.* Therefore, it inevitable poses a conundrum for applying publicized rules in cyberspace.

For example, there has been mounting pressure from law enforcement on age check at social-networking sites (*e.g.*, MySpace, Facebook). Such sites 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. However, social-networking sites are facing significant difficulty to implement such a law enforcement rule of age check [1, 6].

The Internet milieu asks for fast evolvability for rules. The Internet is such an innovative invention that it not only complicates the enforcement of publicized rules, but challenges their enactment as well. Brand new technologies, brand new services, and brand new concepts of life are coming out every now and then. They call for new rules to manage the order. And they require the enactment of new rules to be fast enough to keep up with their innovation pace. Unfortunately, the current practice is that the enactment of publicized rules is far behind what we need.

For example, the Internet milieu adds many ambiguities to the "fair use" provisions of the copyright-law [11, 23]. Moreover, it even puts the fundamental rationality of copyright law in question as a result of the free software movement, in which the GNU public license and the Linux are typical examples [21, 22]. Regulations that address virtual economy and virtual crimes in online games (*e.g.*, Second Life) tend to be imperative since the "virtual assets" are no longer virtual — they can be traded for real world money [10, 19]. However, such regulations could face brand new disputes like "why didn't you buy a dog that can revive, otherwise it would not have been killed", which sounds weird in the real world but is exactly a valid argument in the "virtual world" [23]. Enforcing user integrity in the rising Web 2.0 applications is another example, which incurs questions such as how to define "spamming" and "vandalization" technically, and how to react to them.

Rules and tussle. Rules could be controversial and need to be evolved. They are usually temporary results of the *tussle* — different stakeholders have interests that may be adverse to each other, and these parties each vie to favor their particular interests. The tussle is an important reality of a society as pointed out in [17]. It highlighted that accommodating the tussle is crucial to the evolution of the Internet's technical architecture. And to do this, it is necessary to separate functions that are within the tussle space from those out of it.

Rules are essential functions within the tussle space. They help to manage the order of tussles (*e.g.*, debates in a court must follow laws, business competitions must conform to laws and regulations). Meanwhile, they affirm the results of tussles, thereby guiding new tussles (*e.g.*, common disputes could lead to enactments of new laws and regulations). Therefore, the question of accommodating the tussles transfers to a sub-question of "how to accommodate rules". In particular, according to [17], it is essential to separate rules from the Internet's engineering design.

In this paper, we propose the design guidelines for a solution to the above sub-question. The core of this solution is a novel user authentication architecture that allows the rules to be well separated from the Internet's engineering design.

A key challenge for this authentication architecture is that it should not *in itself* rely on a controversial rule. This means that we should *neither* assume the validity of supporting user accountability (due to the rule enforcement requirement) *nor* take for granted the principle of preserving user anonymity (due to the Internet's basic demand). Instead, we should:

- Accommodate a tussle between the user accountability and the user anonymity — *Guideline 1.*

Stakeholders involved in this tussle can be users, service providers, victims (including young victims' parents), and the government, *etc.*

Service-provider scope rules. To address the fast evolvability requirement for rules in the Internet milieu, we propose *service-provider scope rules*. This is based on the fact that rules with a smaller application scope can be enacted (or amended) much faster than those with a larger application scope. For example, it usually takes much longer to enact a state-wide law than a consortium-wide regulation; and it usually takes much longer to enact a generalized service regulation than a service-specific policy.

A service-provider scope rule is a rule enacted by a specific service-provider. It defines for a specific service the rights and liabilities of this specific service-provider and its customers. It serves for contexts where no existing rules are applicable or the existing rules are unclear. For example, a content delivery service provider can set its own criteria of "fair use" for cases that are unclear in terms of the copyright law; and different content delivery service providers can set different criteria.

The service-provider rule is enacted and evolved as a result of the tussles (*i*) between the interest of the service provider itself and that of its customers; (*ii*) between itself and the competing service providers; and (*iii*) between itself and other involved stakeholders (*e.g.*, victims, government).

A service-provider scope rule might be much more controversial than existing types of rules such as laws and regulations. Therefore, it is important to give a service-provider both *strong incentives* and *high feasibility* to improve the rule. This leads to the following guidelines in our design. We should:

- Allow an effective competition among competing service providers, such that each of them has strong incentives to improve its rules — *Guideline 2.*
- Minimize the distortion on a service's normal functions when a rule changes, such that the changes can be highly feasible — *Guideline 3.*

Mapping capability. To provide a regular way for service providers to react to liable users in order to enforce the rules (in particular, the service-provider scope rules), we will show (in Section 2.3) that it is *essential* to support the *mapping capability*, *i.e.*, the ability to map different cyberspace aliases of the same user *without* knowing the user's real identity. It is a *week form of user accountability* in contrast with the strong form, *i.e.*, the ability to resolve a user's real identity.

FCM model. The proposed user authentication architecture adopts a *feeder-carrier-mapper (FCM) model* to address the three guidelines described above. First, the model adopts a way similar to the rising authentication approaches such as OpenID [7] and SSO [9]; that is, decoupling identity providers (*i.e.*, those who provide the user authentication service) from service providers. Secondly, it further divides identity providers into (*i*) *feeders* — where users register, (*ii*) *carriers* — where users authenticate, and (*iii*) *mappers*— who can map different aliases of the same user.

The model separates the rules from normal functions of identity providers and service providers through standard interfaces. Under appropriate conditions defined by rules, it can support both the mapping capability and the strong form of user accountability as normal functions. In this way, it can well accommodate rules in cyberspace, including the service provider scope rules.

The rest of this paper is organized as follows. In Section 2, we perform a theoretical analysis on rules in cyberspace and narrow down our research topic into several key points. This provides a background to understand our technical design shown in Section 3 and 4. In Section 3, we introduce the FCM model, *i.e.*, our solution to the question of "how to accommodate rules in cyberspace". In Section 4, we show an implementation of the FCM model to demonstrate its feasibility, to propose core algorithms, and to explain important design subtleties. Finally, we conclude in Section 5.

2. RULES IN CYBERSPACE

2.1 Rules for Internet Users

In this paper, we focus on the publicized rules for *users*, *i.e.*, those that define the rights and liabilities of Internet users. Although the publicized rules in cyberspace necessarily include those for service providers as well, such rules are not the key issue because the service providers are not anonymous. Indeed, it is the anonymity of the Internet users that leads to the very difficulty of the publicized rules. Our central research question therefore is: *How can we accommodate the publicized rules for users in cyberspace?*

Two types of rules. In particular, we focus on the following two types of rules for Internet users:

- *Authorization rule.* It defines "who can do what at

what time under what condition", *i.e.*, a user's *rights*.

• *Reaction rule*. It defines "who is reacted to (*e.g.*, penalized) by what at what time under what condition", *i.e.*, a type of *liability* for a user. A reaction rule is usually in conjunction with an (or multiple) authorization rule(s). A typical instance is "who is penalized with what at what time when he violates which authorization rule(s)." However, the reaction is not necessary a penalty due to the potential controversy of a rule. It might be simply keeping a record in the user's profile as we will introduce in Section 2.3.

Controversy. Rules can be controversial as we explained in Section 1. An authorization rule can be most controversial in its condition, *i.e.*, the "under what condition" clause. For example, the copyright law does not prohibit the "fair use" of a copyrighted material; while the condition of "fair use" can be unclear, in particular in the Internet milieu [11, 21, 22]. Email spamming is annoying and people want to suppress it; while it can be ambiguous on the definition of "spamming" that can well distinguish a spamming act from legitimate advertisement behavior. A reaction rule can be controversial since its condition is usually in conjunction with an authorization rule, hence it inevitably inherits the authorization rule's controversy.

Reaction time. The "at what time" clause in the reaction rule is a critical issue. A reaction rule can often be applied *after* the act of violation rather than when the act is still in progress. For example, a culprit is usually on trial and punished after the act of crime rather than during the act. However, the user anonymity in cyberspace makes such kind of reactions significantly harder. We simply do not know *whom* we should react to. Therefore, the current Internet usually requires that a reaction can be applied extraordinarily fast, often during the act. For example, rules to counter distributed denial of service (DDoS) attacks would require the ability to detect and react to a DDoS attack in its early stage (before it can cause too much impact). In essence, it often requires the reaction to be "nearly proactive" rather than "reactive".

Such a requirement can often be impractical. To identify an act of violation can be complex due to both the controversy of rules (hence a verdict is complex to reach) and the technical escalation (*e.g.*, an attacker's techniques to bypass DDoS attack detection can be escalated, hence the detection algorithms have to be escalated as well). It might be beyond the capability of the computer, hence require human involvement (*e.g.*, to judge whether a content delivery service session infringes the "fair use" of copyrighted materials). Such a process will take much longer time than the computer-automated process. Moreover, even for a process that is possible to be automated by computers, it could still take a lot of time (due to the process's complexity) — much longer than required, *e.g.*, before a content delivery service session is finished, or before an attack can cause too much impact.

Therefore, our research focuses on the cyberspace solution that allows a reaction to be truly "reactive", *i.e.*, the reaction can be applied after the act of violation. The solution is to provide the accountability of a liable user (when it is appropriate). This follows Guideline 1.

2.2 Rules at a Service Provider

Publicized rules at a service provider can include cyberlaws [26] and other Internet regulations. Moreover, we propose a supplementary type of publicized rules at a service provider — *service-provider scope rules*.

Service-provider scope rules. A service-provider scope rule is a publicized rule enacted by a specific service provider. It defines the authorization rule or reaction rule for this service provider's users in context of a specific service. As described in Section 1, the motivation to introduce the service-provider scope rules is to meet the fast evolvability requirement for rules in cyberspace. The main idea is based on the fact that the smaller the application scope of a rule, the faster the rule can be enacted (or amended).

Although our research covers the topic about how to help enforce cyberlaws and other Internet regulations as well, our main focus is on the question of *how to accommodate the service-provider scope rules*, as the service-provider scope rules are the *key* to the fast evolvability for rules in cyberspace.

Openness. All publicized rules at a service provider are open, *i.e.*, "publicized". This is apparent for the cyberlaws and regulations since their contents must be announced to the public before they can become effective. Nevertheless, for service-provider scope rules, this will also become an essential property due to the potential controversy of the rules (they can be even more controversial than the cyberlaws and regulations). The tussle space will force the rules to be exposed to public scrutiny in order to justify them (otherwise, the service provider could fail in competition).

In terms of this property, policies at a service provider that are hidden from the public are not the service-provider scope rules as we defined, since they are not the "publicized rules".

2.3 Service-Provider Scope Rules

Advantages. The service-provider scope rules allow the rule enactment to be much faster than traditional rules such as cyberlaws and regulations. By minimizing the application scope, a service provider obtains the maximum decision-making power on a rule, which significantly simplifies the rule's enactment process.

Meanwhile, the application scope is defined on a per-service-provider basis rather than on the territorial ba-

sis (*e.g.*, as cyberlaws are[1]). This makes the service-provider scope rules particularly suit the cyberspace since the Internet does not make geographical and jurisdictional boundaries clear. A service-provider scope rule takes effect for any users who choose the specific service provider *regardless of their territorial presence*.

In addition, a service provider stands in the best position that can tie a rule with results of the tussle underlying a service. It knows the best about the service that it provides, and bears direct relationships with all major stakeholders involved in the tussle — users, competing service providers, identity providers, the service consortium, the government, *etc*. Therefore, by giving the service provider sufficient decision-making power on a rule, the rule may evolve responsively in pace with the underlying tussle.

Moreover, the openness of service provider scope rules allows better evolvability for *service-provider-side* approaches comparing with those proprietary policies or mechanisms hidden from the public (*e.g.*, Comcast was found to use a "blunt" proprietary mechanism to impose broad restrictions on peer-to-peer file-sharing [4]). The openness facilitates competing service providers to learn from each other such that they can make improvements quickly. It also exposes their approaches to public scrutiny to ensure the approaches' integrity. Such integrity is defined by results of tussles.

Reaction method. A service provider scope rule[2] is *not* necessarily a rule within the legal system as cyberlaws and regulations are (although it may evolve to be when it becomes mature). It therefore may not be authorized to use reaction methods provided by the legal system, *e.g.*, fines, hearings, trials, or custodies.

For this reason, we focus on a conservative reaction method — to *keep a record* in the liable user's profile about the violation, which tends to be valid for almost any service provider scope rules. Our key question here is how to exploit this react method to *enforce the rule*, *i.e.*, to deter a user from violating the rule.

Theoretically, keeping the record should be sufficient if user accountability exists, *i.e.*, we are able to identify the physically present user. This is because the record can therefore reserve the feasibility to call for any other reaction methods when appropriate. For example, a service provider can choose to block the user's account based on the record; collaborative service providers who respect the same rule and recognize the same record may jointly limit service access for accounts of the same user. A record or a series of records (either at one service provider or across multiple service providers) can also serve as the evidence that justifies reaction methods provided by the legal system.

Mapping capability — a weak form of user accountability. Unfortunately, the premise of acquiring the user accountability could in itself be subject to the authorization by the legal system. This could be particularly true for the strong form of user accountability, *i.e.*, to resolve a user's real identity. It is not likely that a service provider can be authorized to do so for most of its service-provider scope rules.

We therefore propose a weak form of user accountability — *mapping capability*. It is the ability to map different cyberspace aliases of the same user without knowing the user's real identity. This weak form of user accountability is much more likely to be separated from the legal system than the strong form. Meanwhile, it is still sufficient to help enforce the rules. For example, a service provider can use this ability to resolve accounts associated with the same user. Therefore, if it wants to block the user, it can really block him rather than just block one account of the user (hence the user can circumvent by opening another account). Two collaborative service providers (*e.g.*, two social networking sites) can use this ability to share user profiles to help better manage their users.

3. FEEDER-CARRIER-MAPPER MODEL

3.1 Design Road Map

Figure 1 shows the design road map from our research goal to our solution. The first two boxes (from the left) summarize the goal and issues raised by the goal as introduced in Section 1. As we introduced, these issues lead to the three major design guidelines, which we rewrite below for convenience:

- *Guideline 1*: Accommodate a tussle between the user accountability and the user anonymity.
- *Guideline 2*: Allow an effective competition among competing service providers, such that each of them has strong incentives to improve its rules.
- *Guideline 3*: Minimize the distortion on a service's normal functions when a rule changes, such that the changes can be highly feasible.

The third box lists our partial solutions (Section 3.2) to the raised issues, which implement the three guidelines. Integrating these partial solutions in a coordinated way leads to our holistic solution (the fourth box, Section 3.3) — the feeder-carrier-mapper model.

3.2 Partial Solutions

We first introduce the partial solutions, which are also the design criteria of the holistic solution.

Unified identity, distributed aliases and profiles. To enable the user accountability, our design includes a unified cyberspace identity for each user. To

[1]For example, a transaction between a violator in US and a victim in Britain through a server in Canada could be subject to the cyberlaws of all three countries [26].

[2]For example, a service provider's own provision on "fair use" of copyrighted materials as described in Section 1.

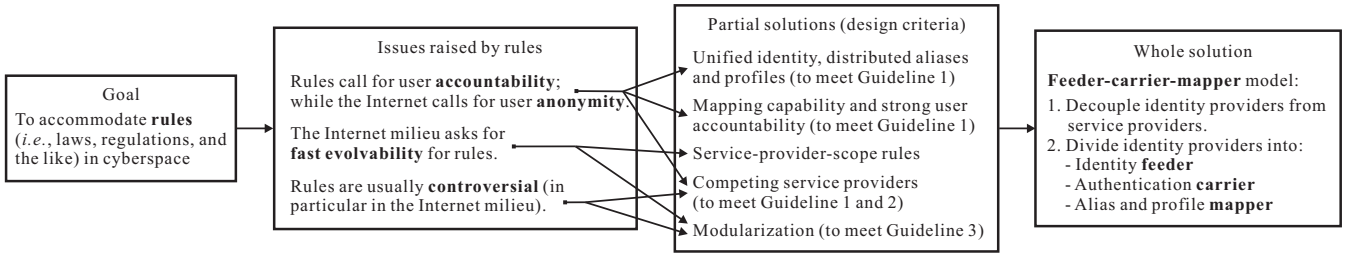

Figure 1: Design road map from the goal to the solution

achieve this, we connect this cyberspace identity to the user's real identity. To support user anonymity, this unified identity is hidden from the public Internet. Different service providers use different aliases of a user and maintain the user's profile distributedly. The aliases can be connected to the unified identity under appropriate conditions. Such conditions are defined by rules resulting from the underlying tussle. In this way, we accommodate a tussle between the user accountability and the user anonymity, which meets Guideline 1.

Mapping capability and strong user accountability. Under the appropriate conditions, our design can support both the mapping capability (*i.e.*, the weak form of user accountability) and the strong form of user accountability, *i.e.*, to resolve a user's real identity. In particular, it provides the mapping capability as a scalable common service served by identity providers to facilitate the service-provider scope rules.

Supporting service-provider scope rules. The design supports the service-provider scope rules, which are the key to the fast evolvability for rules in cyberspace. It allows the service-provider scope rules to be truly "reactive", *i.e.*, a reaction can be applied after the act of violation. It also allows the common reaction method of keeping a record in user profile (as introduced in Section 2.3) to be effective to help enforce the rules by exploiting the mapping capability.

Introducing competing service providers. Our design introduces multiple competing service providers of the same service for the same customer. It provides each customer freedom of choice among the service providers, which can drive an effective competition among them. This meets Guideline 2. Moreover, the competition among identity providers (*i.e.*, service providers of the user authentication service) additionally helps Guideline 1. This is due to the identity providers' important roles in the tussle between the user accountability and the user anonymity.

Modularization. Our design decouples the module of rules from that of the normal functions at each service provider. Moreover, we modularize identity providers into several independent entities. Such design helps to minimize the distortion on the normal functions when a rule changes. This meets Guideline 3.

3.3 Holistic Solution

Now we introduce our holistic solution, the feeder-carrier-mapper (FCM) model, which incorporates all the partial solutions. The FCM model is a user authentication architecture model that can enable the cyberspace to accommodate rules effectively.

Overview. The FCM model adopts a modularization shown in Figure 2. It decouples identity providers from service providers, and divides the identity providers into *feeders*, *carriers*, and *mappers*:

(*i*) Feeders (or identity feeders) are identity providers where users register. (*ii*) Carriers (or authentication carriers) are identity providers where users are authenticated. They provide scalable user authentication services to the public. (*iii*) Mappers (or alias and profile mappers) are *a special type of carriers*. Besides functions of ordinary carriers, mappers additionally have the mapping capability. The mapping capability is authorized by both the user and the feeder. With the mapping capability, mappers can provide to others a *mapping service*, *i.e.*, to map aliases of the same user. The mapping service is managed by rules.

To provide the unified cyberspace identity as described in the partial solution, users should register at a feeder with their real identities. A feeder can export user 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. (The real identities are never exported to carriers.) The authentication data exported to different carriers use different aliases for the same user to retain user anonymity.

Non-identity service provider. Since we decouple identity providers from service providers, we define the term *non-identity service provider* to refer to those service providers other than the identity providers for disambiguation. Thus, service providers are divided into two categories: (*i*) identity providers — who provide the user authentication service, (*ii*) non-identity service providers — who provide other services (*e.g.*, Web, Email, social networking, content delivery, *etc*).

Feeder-carrier decoupling. Decoupling identity providers into feeders and carriers makes a fundamental improvement to the *unified identity* solutions comparing with existing ones such as *OpenID* [7], *Single Sign-On (SSO)* [9], *Microsoft passport* [15], and *VeriSign unified*

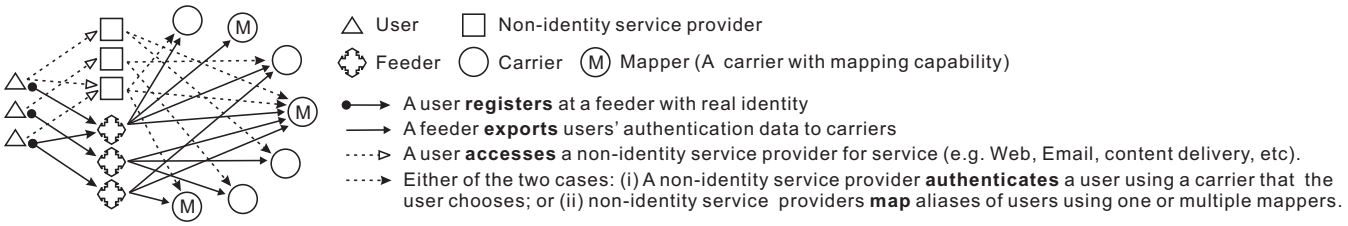

Figure 2: Modularization in feeder-carrier-mapper model

authentication [12].

First, the decoupling makes it much easier to connect a user's cyberspace identity to his real identity using resources we already have. There are already many feeders that have users' real identities registered. For example, a school has the real identities of its students; a bank has the real identities of its customers; and a company has the real identities of 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 connect the two types of identities.

Secondly, the decoupling can drive a more effective competition among identity providers than the existing solutions. It empowers users much more freedom of choice among identity providers. For instance, 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 (*e.g.*, Google was already caught disregarding user privacy [2]). By contrast, the existing solutions do not allow such freedom of choice because a user is restricted to be authenticated at the same identity provider where he registers.

Moreover, the decoupling can yield high scalability and high reliability for the authentication service via service replication. This is due to the feasibility that a feeder can export authentication data to many carriers and any of these carriers can authenticate the users independently of the feeder.

A more complete list of advantages and extensive performance evaluation on a system that adopts the feeder-carrier decoupling can be found in [18]. And we will explain the implementation of such a system in Section 4.1 and the Appendix.

Competing service providers. There are competing non-identity service providers, feeders, carriers, and mappers for the same customer (a user, a non-identity service provider, or a feeder) as shown in Figure 2. This empowers the customer the freedom of choice among the service providers. For example, a non-identity service provider can change the carriers or mappers that it uses (because it trusts or prefers them) without affecting its service; a user can choose a carrier or a mapper on the fly for each authentication; likewise, a feeder can change its choice of carriers or mappers as well.

Mapping service and mapping capability. A mapper can provide a mapping service to non-identity service providers, *i.e.*, help them to map aliases of the same users. The service exploits the mapper's mapping capability. The mapping capability is authorized by both the feeder and the user; and the authorization is done by revealing the user's *mapping identifier* — an identifier unified (for the same user) across all mappers that the feeder chooses.

The feeder does the authorization by exporting to a mapper *verification data* of each user's mapping identifier, while the *mapping identifier itself is hidden*. The mapping identifier will be revealed by the user during an authentication and the user has the right to decide whether he is willing to reveal it. The mapper can verify a revealed mapping identifier using the exported verification data. After that the mapping capability for the specific user is finally enabled at this mapper.

The mapping capability at a specific mapper can be revoked under certain conditions (defined by rules, *e.g.*, if the mapper is caught to use its mapping capability for unreasonable surveillance). The FCM model can provide the revocation both on a *per-feeder* basis (*i.e.*, revoke the mapping capability for all users of a specific feeder) and on a *per-user* basis (*i.e.*, revoke the mapping capability for a specific user).

A mapper may provide the mapping service to ordinary carriers in addition to non-identity service providers directly to make the service scalable. In such cases, the ordinary carriers serve as intermediate agents of the mapping service as we will explain in Section 4.2.4.

Rules and FCM model. The FCM model decouples the module of rules from that of the normal functions at each service provider as depicted in Figure 3. Therefore, rules can change without distorting the normal functions of a service and vice versa.

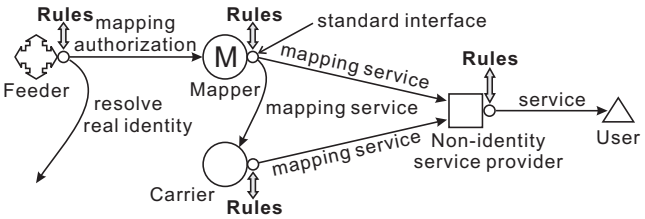

Figure 3: Rules and FCM model

At non-identity service providers, the user accountability serves as an effective interface between the two

modules. The normal functions of a service can simply keep a record in user profile for a concerned act without evaluating it. While the rule module can evaluate the act according to rules and react to a liable user based on the user accountability embedded in the record(s).

At identity providers, the rule module can include provisions about user accountability (which can be controversial). Therefore, decoupling the rule module from the normal functions prevents the technical architecture from relying on a controversial rule. The normal functions here define standard functionalities and the rule module manages the normal functions through standard interfaces. The following are some examples:

(*i*) A feeder has the normal function to export users' mapping identifiers to mappers and the normal function to resolve a user's real identity. While the feeder's rule module can define "which mappers are permitted for the mapping capability authorization", "under what condition a mapper's mapping capability should be revoked", and "to whom and under what condition a feeder is justified to disclose a user's real identity".

(*ii*) A mapper has the normal function to offer the mapping service to non-identity service providers. While the mapper's rule module can define "to which non-identity service providers and for what purposes, the mapping service is permitted". For example, the World Wide Web Consortium (W3C) may enact rules that provision the mapping service to social networking sites in light of the mounting demands for interoperability among such sites [13].

For all service providers, rules incorporated in the rule module can include cyberlaws, Internet regulations, and the service-provider scope rules.

4. AN IMPLEMENTATION

In this section, we show an implementation of the FCM model. Our main purpose here is to demonstrate the feasibility to implement this model, to propose candidate algorithms that meet its central design criteria, and to explain important design subtleties.

4.1 Feeder-Carrier Decoupling

We first introduce our core algorithm to implement the feeder-carrier decoupling, in particular, how to *export* authentication data from a feeder to a number of carriers *without compromising the data confidentiality*. Our solution here is to exploit cryptographic hash functions. Instead of replicating the original authentication data, a feeder exports to carriers hashed copies of the data. With the hashed data, each carrier can authenticate the feeder's users independently of the feeder, while the data confidentiality is well retained.

The original authentication data at the feeder for each user are in form of a 2-tuple $\{PID, SEC\}$. PID is the user's *permanent identity*; SEC is the user's *secret code*. Both fields are assigned by the feeder on a per-user basis and only known by the user and the feeder. Denote by C a specific carrier. Denote by H_C an *exporting hash function* for carrier C. Denote by the 2-tuple $\{HPID_C, HSEC_C\}$ the hashed authentication data for each user at carrier C. We have:

$$HPID_C = H_C(PID),\ HSEC_C = H_C(SEC) \quad (1)$$

In our implementation, we choose the following *SHA-256* based cryptographic hash function for H_C:

$$H_C(x) = SHA(x\ XOR\ hid_C) \quad (2)$$

SHA – the SHA-256 hash function. hid_C – the *hash function identifier* for C assigned by the feeder, $hid_C \neq 0$. The feeder assigns different carriers with different hid_C, hence different H_C.

Carrier C can authenticate the user using $\{HPID_C, HSEC_C\}$, which proceeds as follows:

$$TID = f(HPID_C \mid nonce \mid ... \mid PubKey_C) \quad (3)$$
$$passcode = SHA(HSEC_C\ XOR\ nonce) \quad (4)$$
$$(HPID_C \mid nonce \mid ...) = g(TID, PriKey_C) \quad (5)$$

"|" – concatenation mark. f – a function to generate TID from $HPID_C$. g – a function to recover $HPID_C$ from TID. f and g correspond to the encryption and decryption functions of a public key cryptographic algorithm such as *RSA* or *ECC* [3]. In our implementation, we choose the RSA algorithm with 1024-bit keys for f and g. *nonce* – a nonce generated based on time. "..." – additional information encrypted in TID. $PubKey_C$ and $PriKey_C$ – C's public and private keys.

First, the user generates a one-time *temporary identity* (denoted by TID) and a one-time *passcode* based on the $\{PID, SEC\}$ using Equations (1), (3), and (4). Then, he sends the TID and *passcode* to C. Next, C decrypts $HPID_C$ from TID (using Equation (5)), which in turn helps C to resolve $HSEC_C$ (via database search). Finally, C verifies the *passcode* provided by the user by regenerating it the same way as the user does (Equation (5)).

The authentication fully retains the user's anonymity on the public Internet by exploiting the public key cryptography. In addition, it makes the user anonymous across carriers since the hashed authentication data for the same user at different carriers are different.

We provide other implementation details about the feeder-carrier decoupling in the Appendix, including: (*i*) the trust model and the architecture based on which information such as H_C and $PubKey_C$ can be securely announced to the public without relying on a *public key infrastructure (PKI)*[3]; (*ii*) common authentication services that the carriers can provide; (*iii*) the network protocols of the system; (*iv*) the approach to achieve impersonation resiliency.

[3]A global-scale PKI does not exist on the Internet and whether there will be one is yet questionable. We therefore should not assume a global-scale PKI as the design premise.

Moreover, we provide extensive explanation and evaluation on the implementation of a feeder-carrier decoupling system in [18]. It shows that the system can achieve scalability[4], security, efficiency, reliability, and incrementally deployability *at the same time*.

4.2 Mapping

In this section, we explain the core algorithms for alias mapping. We first describe how a mapper can acquire the capabilities of (*i*) *intra-feeder mapping* (Section 4.2.1), *i.e.*, to map aliases of users registered at the same feeder, and (*ii*) *cross-feeder mapping* (Section 4.2.2), *i.e.*, to map aliases of users registered across different feeders. Then we illustrate how mappers can provide mapping services to non-identity service providers based on such capabilities (Section 4.2.3). Finally, we show how to exploit ordinary carriers to make the mapping services scalable (Section 4.2.4).

4.2.1 Intra-Feeder Mapping

Mapping capability authorization. A mapper acquires the intra-feeder mapping capability by obtaining a user's *mapping identifier* (Section 3.3). Since this identifier is unified (for the same user) across all mappers of a feeder, any mapper can easily identify the unique users registered at the given feeder, thereby providing the intra-feeder mapping.

Here, we describe the algorithm of how a mapper can acquire a user's mapping identifier through the authorization of both the feeder and the user as we highlighted in Section 3.3. Denote by M the specific mapper, then each user's authentication data exported to this mapper are $\{HPID_M, HSEC_M\}$, and the corresponding exporting hash function is H_M. The algorithm also includes the following user data:

- MID: a user's mapping identifier.
- *mid*: the user's *mapping identifier seed*, only known by the user and the feeder.
- K_M: the user's *mapping key* at M.
- MVC_M: the user's *mapping verification code* at M.

They bear the following relationships:

$$MID = SHA(mid) \quad (6)$$
$$K_M = H_M(mid) \quad (7)$$
$$MVC_M = SHA(MID\ XOR\ HSEC_M)\ XOR\ SHA(K_M) \quad (8)$$

When a feeder authorizes M to be its mapper, it exports to M each user's MVC_M. Therefore, each user's authentication data at M become the 3-tuple $\{HPID_M, HSEC_M, MVC_M\}$. During an authentication, a user that is willing to authorize M the mapping capability will provide MID and K_M, which are hashed from his *mid* (using Equations (6) and (7)). M can then verify them using Equation (8).

[4]It includes the system's resiliency against DDoS attacks.

This algorithm has the following key properties: (*i*) With $HSEC_M$ and MVC_M, M can easily verify MID and K_M, but can not infer MID and K_M. (*ii*) Even if M knows the list of MIDs (while the pairwise associations between MIDs and users are hidden), M still can not infer a user's MID, *i.e.*, the association between the user and his MID. (*iii*) The user can not forge fake MID and K_M that satisfy Equation (8).

Properties (*i*) and (*ii*) ensure that M's mapping capability for each user must be authorized by the user before it can be enabled. Property (*ii*) is necessary since mappers that have been revoked the mapping capability may know the list of MIDs as we will introduce next. In addition, it makes the algorithm immune to leaking MIDs (considering that the same *MID* is disseminated to all mappers, hence has the chance to leak at any of them). Property (*iii*) ensures that a user must provide the real MID in order to pass the verification.

Mapping capability revocation. As introduced in Section 3.3, the FCM model supports the revocation of a mapper's mapping capability both on a *per-user* basis and on a *per-feeder* basis.

A. Per-user revocation. The per-user revocation for a specific mapper (denoted by M_R) is done by changing a user's MID. To do this, a feeder changes the user's mapping identifier seed *mid*. Accordingly, the MVC_M changes and should be updated to mappers. However, since the MID is unified, the change should be applied to all mappers of the feeder rather than just to M_R. Therefore, it may also revoke the mapping capability of other mappers (as a side effect). To address this, the feeder exports to all mappers except M_R a *mapping hint* (denoted by $MHINT_M$) in addition to the new MVC_M to help other mappers retain the mapping capability.

Denote by K_M^{old} the user's *old* mapping key. $MHINT_M$ bears the following relationship with MID and K_M^{old}:

$$MHINT_M = MID\ XOR\ H_M(K_M^{old}) \quad (9)$$
or
$$MID = MHINT_M\ XOR\ H_M(K_M^{old}) \quad (10)$$

Since a mapper that has been authorized the mapping capability for the user knows K_M^{old}, it can get the user's new MID using Equation (10).

Meanwhile, the feeder also exports to all mappers except M_R the difference (denoted by KD_M) between the user's old and new mapping keys to help authorized mappers retain their knowledge on the mapping key K_M. KD_M is computed as follows:

$$KD_M = K_M\ XOR\ K_M^{old} \quad (11)$$

Therefore, the authentication data that a feeder exports to a mapper can be the 5-tuple $\{HPID_M, HSEC_M, MVC_M, MHINT_M, KD_M\}$. All the five fields are different (not unified as MID is) across mappers, thereby *disabling* the mapping capability *by default*.

B. Per-feeder revocation. The per-feeder revocation (*i.e.*, to revoke a mapper's mapping capability for all users of a feeder) is done by changing the exporting hash function H_M associated with the specific mapper $M = M_R$. As a result, all users' authentication data $\{HPID_M, HSEC_M, MVC_M\}$ at M_R will change accordingly. The change only affects M_R, and therefore is much more efficient than the alternative approach of doing per-user revocation for all users. Although M_R may know some MIDs (since MIDs remain unchanged), the pairwise associations between MIDs and users are lost, hence the mapping capability is revoked.

A related topic here is how to change the H_M *without revoking* the mapping capability, which may be used for maintenance purposes. In such a case, the new authentication data exported to the mapper will be $\{HPID_M, HSEC_M, MVC_M, H_M(K_M^{old}), KD_M\}$. $H_M(K_M^{old})$ helps to establish the pairwise association between the MID and the new authentication data for each user whose MID and K_M^{old} were known. KD_M helps the mapper to retrieve K_M from K_M^{old}. In this way, the acquired mapping capability can be retained.

4.2.2 Cross-Feeder Mapping

Cross-feeder mapper. To support alias mapping for users registered across two feeders (denoted by $F1$ and $F2$), the two feeders authorize one or multiple carriers as their *cross-feeder mappers*. A cross-feeder mapper maintains a *mapping table* that maps users' MIDs at the two feeders (denoted by MID_{F1} and MID_{F2}). Each row of the table is in form of $\{SHA(MID_{F1}), SHA(MID_{F2})\}$. Here we use the SHA version instead of the original MID to avoid the disclosure of MID. A cross-feeder mapper provides to other mappers a service to map different feeders' MIDs. Based on this service and each mapper's own intra-mapping capability, the cross-feeder mapping can be achieved.

Mapping table generation. The mapping table can be generated using an external algorithm that exploits users' real identities registered at feeders. Each pair of feeders can decide an external algorithm respectively that best suits them. The following is an example algorithm for a common case that could happen.

Suppose feeders $F1$ and $F2$ are exporting a mapping table to mapper M, but each of them does not want the other feeder to know what users it has. Meanwhile, they must prevent M from knowing users' real identities. To achieve this, they can adopt the algorithm below:

1. $F1$ and $F2$ choose certain unique user identifier(s) (*e.g.*, citizen ID number, social security number, or driver license number) that both have.
2. $F1$ and $F2$ agree on a cryptographic hash function and keep it secret from M.
3. $F1$ exports to M the $SHA(MID_{F1})$ along with the hashed copy of the chosen unique user identifier of each user using the above secret hash function. $F2$ does in a similar way.
4. Based on the hashed copy of unique user identifiers provided by $F1$ and $F2$, M can establish the mapping between $SHA(MID_{F1})$ and $SHA(MID_{F2})$, thereby generating the table.

Since $F1$ and $F2$ export the hashed copy of the chosen unique user identifier only to M but not to each other, they prevent each other from knowing what users they have. Meanwhile, since M does not know the secret hash function, it is unable to infer users' real identities even if it knows the candidate list of users' unique identifiers (*e.g.*, the list of all possible citizen ID numbers).

Each feeder is responsible to keep its cross-feeder mapper updated when any user's MID is changed. For example, it should send to M an update of the $SHA(MID)$ along with the user's hashed unique identifier.

4.2.3 Mapping Service

Figures 4(a)–(f) illustrate how mappers can provide mapping services to non-identity service providers based on their mapping capability. Throughout this section and Section 4.2.4, all "service providers" mean the non-identity service providers.

User identifier. As shown in Figure 4(a), before a mapper can provide a mapping service, it should first issue a user identifier (denoted by UID) to a service provider for a specific user. A mapper issues the user identifier after the user authorizes it the mapping capability (by providing the MID and K_M) during an authentication. Later, a service provider can quote this UID when using the mapping service.

Denote by $UID_{(S,M,F)}$ the UID issued to service provider S by mapper M for a user registered at feeder F. Denote by $HPID_{(M,F)}$ the user's $HPID_M$ stored at M and hashed from its PID registered at F. Denote by ID_S an identifier of S (*e.g.*, the MD5 digest of S's name). Denote by G_M the encryption function of a symmetric-key algorithm (*e.g.*, AES, 3DES) that M chooses, with a key only known by M. $UID_{(S,M,F)}$ is computed as follows:

$$UID_{(S,M,F)} = G_M(HPID_{(M,F)} \mid ID_S) \quad (12)$$

Since the UID is a function of ID_S, the UIDs issued by the same mapper to different service providers are different for the same user, thereby preserving the user's anonymity across service providers by default.

Mapping service. Now we explain the mapping services through typical cases. In all cases, assume the mappers' rule modules permit the specific mapping service to the specific service providers.

Figure 4(a) shows a trivial case of intra-feeder mapping in which a service provider S always uses the same mapper M of a feeder F. In this case, the UID can help S to uniquely identify a user of F due to the *uniqueness*

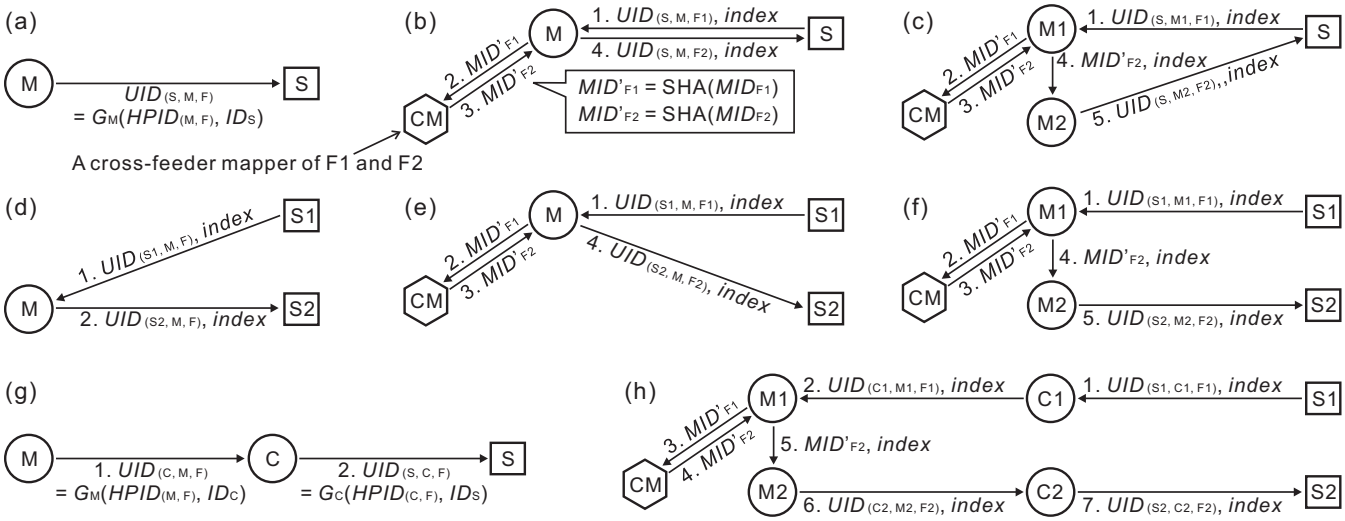

Figure 4: The mapping services

of users registered at F (which is guaranteed by the real identity registration at F as described in Section 3.3).

Figure 4(b) shows a scenario of cross-feeder mapping. Service provider S wants to resolve the same user's UIDs derived from his accounts registered at two different feeders $F1$ and $F2$, such that it can identify the unique user. To do this, S uses the mapping service provided by M, which is a mapper of both $F1$ and $F2$.

First, S sends to M the $UID_{(S,M,F1)}$, *i.e.*, the UID derived from the user's account at $F1$. Then M decrypts $UID_{(S,M,F1)}$ to recover $HPID_{(M,F1)}$, which in turn helps M to fetch MID_{F1} of the user through database search. Next, M sends $SHA(MID_{F1})$ to a cross feeder mapper CM (of $F1$ and $F2$) which can resolve $SHA(MID_{F2})$ of the user. If M has been authorized by the user the mapping capability for his account at $F2$, M should have known MID_{F2} (hence $SHA(MID_{F2})$) and can resolve $HPID_{(M,F2)}$ through database search. Finally, M generates $UID_{(S,M,F2)}$ based on $HPID_{(M,F2)}$ and returns $UID_{(S,M,F2)}$ to S.

M can provide the mapping service in bulk, *i.e.*, perform alias mapping for a bulk of S's users. In such a case, S sends a bulk of $UID_{(S,M,F1)}$ along with an *index* associated with each $UID_{(S,M,F1)}$. M returns a bulk of $UID_{(S,M,F2)}$ along with the corresponding indices.

In general cases, M does not necessary happen to be a mapper of both $F1$ and $F2$. Figure 4(c) shows such a case. The scenario is similar to Figure 4(b), but here $F1$'s mapper is $M1$ and $F2$'s mapper is $M2$. In this case, the mapping service is jointly provided by $M1$ and $M2$. After $M1$ resolves $SHA(MID_{F2})$ via CM, it relays $SHA(MID_{F2})$ to $M2$. And $M2$ will return the final result $UID_{(S,M2,F2)}$ to S.

Figure 4(c) also covers a general scenario of intra-feeder mapping as its special case, *i.e.*, $F1 = F2$. This is the intra-feeder mapping scenario when S does not always use the same mapper. For this case, the steps to resolve $SHA(MID_{F2})$ from $SHA(MID_{F1})$ are skipped since the two values are the same.

Figures 4(d)–(f) depict the cases of mapping services to a pair of service providers $S1$ and $S2$ instead of to a single one. Comparing with Figures 4(a)–(c), the difference lies in that the service input is provided by $S1$ while the service output is given to anther service provider, $S2$. In such cases, the field *index* plays an additional role other than supporting the bulk service. It maps the same user at $S1$ and $S2$ without exposing to $S2$ the user's UID at $S1$ (and vice versa). This can be desirable in light of the service providers' privacy if they have some sensitive use of the UIDs.

4.2.4 Mapping via Carriers

In this section, we explain how to make ordinary carriers serve as intermediate agents of the mapping services. We first explain the motivation of doing this.

Management hierarchy. The controversy of the mapping capability (as introduced in Section 1) implies the potential complexity to manage the mapping services. Although the FCM model has put most of such complexity into the rule space through the decoupling between the rule and normal function modules, it is still an essential engineering space question to address *how to facilitate scalable management* of the mapping services. Our solution to this question is to distribute the management burden of identity providers across a *management hierarchy*:

(*i*) Each feeder can export authentication data to many carriers, but it only authorizes a small number of carriers among them to be its mappers. The feeder is responsible to manage the mapping capability authorized to these mappers. (*ii*) Each mapper provides the mapping service to a number of carriers (and service providers) and is responsible to manage them. (*iii*) Each carrier provides the mapping service to a number

of service providers and is responsible to manage them. For example, a carrier can sign a service agreement with the service providers to provision the purposes that the mapping services can be used for. And it should deny the mapping services if a service provider violates the agreement. A mapper can deny the mapping services to a carrier if the carrier fails to well manage its mapping services to the service providers. A feeder can revoke the mapping capability authorized to a mapper if the mapper fails to manage its carriers well. Overall, each identity provider only manages a relatively small number of directly affiliated entities.

Technical approach. Figures 4(g) and (h) illustrate our technical approach to make the carriers serve as the intermediate agents of the mapping services.

As shown in Figure 4(g), instead of issuing a UID to the service provider, a mapper can issue a UID to a carrier, which in turn issues another UID to the service provider. In such a case, the user needs to do two authentications, one at the carrier C, the other at the mapper M. However, regardless of the service provider S, the user only needs to be authenticated at M *once* as long as he uses the same carrier C. Once C obtains $UID_{(C,M,F)}$ from M, C stores it and binds it with the user's authentication data, therefore the user no longer needs to be authenticated again at M.

Figure 4(h) depicts a general case of cross-mapping service to a pair of service providers $S1$ and $S2$. In this case, $S1$ and $S2$ each choose a different carrier ($C1$ and $C2$ respectively) as the direct provider of the mapping service. $C1$ and $C2$ in turn use $M1$ and $M2$ for the mapping service. Comparing with Figure 4(f), which shows the corresponding case without the intermediate carriers, the procedure only adds two extra steps: (i) $S1$ inputs to $C1$ the UID that $C1$ issues to it, and $C1$ maps this UID to the UID that $M1$ issues to $C1$. (ii) $C2$ maps the UID that $M2$ issues to it to the UID that $C2$ issues to $S2$, and outputs the latter UID to $S2$.

5. CONCLUSION

In this work, we perform an in-depth study on the question *how to accommodate rules in cyberspace*: (i) We propose the design guidelines for an Internet architecture that can separate functions within the rule space from those out of it, and can allow the underlying tussles to drive the evolution of rules effectively. (ii) We propose the service-provider scope rules, which are the key to meet the fast evolvability requirement for rules in cyberspace. (iii) We propose the FCM model which adopts the design guidelines and supports the service-provider scope rules. Moreover, we provide an implementation of the FCM model to demonstrate its feasibility and to explain important design subtleties.

6. REFERENCES

[1] Conn. bill would force MySpace age check. http://www.msnbc.msn.com/id/17502005/.

[2] Consumer watchdog overreacts about gmail. http://blogs.zdnet.com/Google/?p=1181.

[3] Elliptic curve cryptography. http://www.isg.rhul.ac.uk/ sdg/ecc.html.

[4] FCC chief again critiques Comcast net tactics. http://www.reuters.com/article/internetNews/idUSN2232444820080422?virtualBrandChannel=0.

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

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

[7] OpenID. http://openid.net/.

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

[9] SAML single sign-on (SSO) service for Google apps. http://code.google.com/apis/apps/sso/saml_reference_implementation.html.

[10] Trade Second Life currencies via the VirWox API. http://blog.programmableweb.com/2009/01/02/trade-second-life-currencies-via-the-virwox-api/.

[11] U.S. copyright office - Fair use. http://www.copyright.gov/fls/fl102.html.

[12] VeriSign unified authentication (white paper). http://www.verisign.com/static/016549.pdf.

[13] W3C: Interoperability key to social networking. http://www.eweek.com/c/a/Web-Services-Web-20-and-SOA/W3C-Interoperability-Key-to-Social-Networking/.

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

[15] Windows Live ID (Microsoft Passport). https://accountservices.passport.net/PPPrivacyStatement.srf.

[16] Carr, D. How Google works. *Baseline Magazine* (July 2006).

[17] Clark, D. D., Wroclawski, J., Sollins, K. R., and Braden, R. Tussle in cyberspace: defining tomorrow's Internet. In *ACM SIGCOMM '02* (Pittsburgh, PA, Aug. 2002).

[18] Deng, L., and Kuzmanovic, A. A feeder-carrier-based internet user accountability service. *Northwestern University Technical Report NWU-EECS-09-12* (Jan. 2009).

[19] Kane, S. F. Virtual wealth management. *New Jersey Law Journal* (Sept. 2006). http://www.virtualjudgment.com/images/stories/NJ_Law_Journal_November_2006.pdf.

[20] Kommerling, O., and Kuhn, M. G. Design principles for tamper-resistant smartcard processors. In *USENIX Workshop on Smartcard Technology* (Chicago, IL, May 1999).

[21] Lessig, L. *The Future of Ideas: The Fate of the Commons in a Connected World*. Random House Inc., Oct. 2001.

[22] Lessig, L. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. The Penguin Press, Mar. 2004.

[23] Lessig, L., and Lessing, L. *Code and Other Laws of Cyberspace*. Basic Books, Inc., July 2000.

[24] 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).
[25] Stallings, W. The PGP web of trust. *BYTE 20*, 2 (Feb. 1995), 161–162.
[26] Trout, B. J. *Cyber Law: A Legal Arsenal For Online Business*. World Audience, Inc., Sept. 2007.

7. APPENDIX: IDNET MESH ABSTRACT

This appendix summarizes main features of a feeder-carrier-based authentication architecture, *IDnet mesh*, proposed in [18]. It helps to understand implementation subtleties of the feeder-carrier decoupling described in Section 4.1.

7.1 IDnet Mesh Framework

IDnet mesh is a general-purpose user identity architecture for the Internet that adopts the feeder-carrier decoupling. It provides to the public a common service — *identity validation*, *i.e.*, to authenticate whether a user is *accountable*, more specifically, whether he or she is a registered user of a trustable feeder. This service manages to validate a user's identity while retain the user anonymity at the same time.

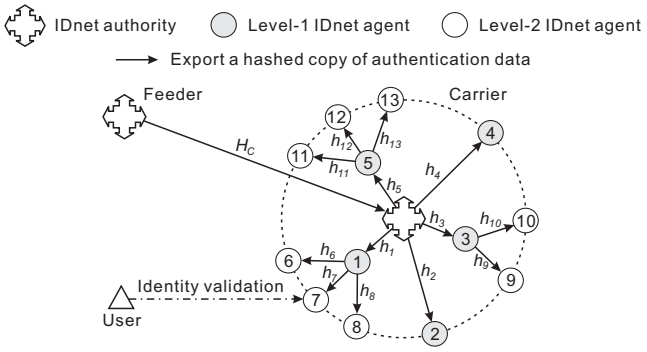

Figure 5: IDnet mesh framework

IDnet. In IDnet mesh, each identity provider is generalized as an *IDnet*, with the feeder and carrier being implemented as two typical cases of the IDnet. An 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 users' authentication data. IDnet agents are designed to provide *high scalability* for the identity validation service through service replication. Each agent is exported from the central database a *hashed copy* of the authentication data and each can authenticate the users independently.

As shown in Figure 5, 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 with their real identities and the feeder creates authentication data for them. The feeder then exports hashed copies of authentication data to carriers (each with a different hashed copy). This not only preserves data confidentiality for the exporting, but also makes users anonymous across carriers.

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 5, each edge agent is exported with a hashed copy of authentication data by following the tree structure and the hashed 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. Each edge agent will announce its public key and hash function sequence to the public. In practice, each edge agent can be a datacenter. And the carrier can adopt a service load balancing approach similar to the Google platform [16].

The identity validation proceeds similarly to the authentication described in Section 4.1. The only difference is the parameters: (*i*) The exporting hash function H_C in Equation (1) is now replaced with a composite hash function that appends H_C with the hash function sequence of the edge agent where the user authenticates (*e.g.*, $h_7h_1H_C(\cdot)$ if the user authenticates at agent 7). (*ii*) The public and private keys used in Equations (3) and (5) become the keys assigned to the edge agent instead of the keys of the carrier.

7.2 IDnet Mesh's Trust Model

The IDnet mesh's trust model addresses the following question: *How can someone or some entity trust an IDnet that it previously does not know?* Here, we outline the model by comparing it with the trust models adopted by *web of trust* [25] (a.k.a. *OpenPGP's PKI*), *X.509 PKI* [5], and social-networking based solutions (*e.g.*, [24]).

The trust model of IDnet mesh shares a flavor of the web of trust 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. By contrast, the X.509 PKI 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 (by endorsing the public key of each IDnet to 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.

The trust model of IDnet mesh adopts a more practical approach than the social-networking based solutions. It removes the trust "burden" from individual users and delegates this job to identity providers (either the feeder that a user registers at or a carrier that the user trusts most).

7.3 IDnet Protocols

The IDnet mesh has two types of network 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 are implemented upon TCP and UDP.

The IDnet system protocol defines two types of protocol messages — *user data messages* and *system announcement messages*. The user data messages are designed to export hashed copies of users' authentication 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.*, each edge agent's public key and hash function sequence, the list of trustable IDnets, public keys of trustable IDnets, *etc*) from an IDnet authority

to all its own agents.

The IDnet user protocol also defines two types of protocol messages — *identity validation messages* and *system announcement messages*. The identity validation messages define the request and response format for the identity validation service. The system announcement messages are designed to propagate an IDnet's system announcements from the IDnet edge agents to the users. A subtlety here is that each feeder will assign a *delegated carrier* to help propagate its system announcements since the feeder does not have its own IDnet agents.

7.4 Services

The IDnet mesh provides two basic identity validation services as shown in Figure 6: *online validation* and *offline validation*. In both services, assume a common scenario: User b wants to validate whether user a is accountable.

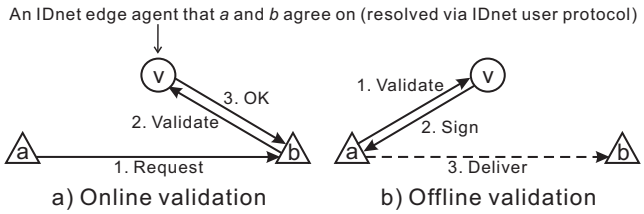

Figure 6: Two basic identity validation services

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 identity via a specific IDnet edge 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 and content delivery), 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 embeds the object's digital fingerprint (using SHA-256) into the additional information encrypted in TID (in Equations (3) and (5) in Section 4.1). Then a uses a specific IDnet edge 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 object's digital fingerprint (decrypted from TID).

Next, a delivers the data object together with the signature, TID, and information about v (including v's public key and hash function sequence, H_C, all of which are signed by the carrier C that v belongs to). 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.

7.5 Impersonation Resiliency

Since the identity validation service can be subject to sensitive use, it is important to provide strong resiliency against user impersonation. In [18], we proposed a small and inexpensive user device, *Internet passport*, to address this. This device can be plugged into the user's computer via a USB port. It can support strong (via two-factor authentication [14]) but convenient user authentication.

A feeder issues each user an Internet passport upon registration. The Internet passport stores the user's secret code SEC and other information, and embeds a built-in clock, based on which it can generate the time-changing *passcode* for authentication. The generation of the *passcode* is unlocked by a password (specified by the user) or the user's biometric property, *i.e.*, fingerprint. The device is designed to be tamper-resistant [8, 20] such that it can effectively deter any attempts to steal the SEC.

This provides strong resiliency against user impersonation in the following way: (i) The tamper-resistant feature 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 password or the user's biometric property to unlock the generation of *passcode* ensures that even if others could get the Internet passport or hijack the user's computer, they won't be able to generate the *passcode* to impersonate the user.