NeTS: FIND: Collaborative Research: Relationship-Oriented Networking

Mark Allman[†], Michael Rabinovich[‡], Nicholas Weaver[†]

[†]International Computer Science Institute
[‡]Case Western Reserve University

{mallman,nweaver}@icsi.berkeley.edu
misha@eecs.case.edu

1 Introduction

One of the key aspects upon which societies are and have always been built is the notion of *relationships*. In fact, relationships are so ingrained into our lives that we often do not consider the degree to which we depend on this underlying societal fabric. We leverage these relationships for everything from trusting our neighbors to watch our kids for a few minutes to trusting a co-worker's recommendation for a plumber to assuming that a friend will return a borrowed tool after finishing a project in several weeks to believing that a neighbor will accurately relay a message to a mutual friend (who will then believe the message even though it is second-hand). In addition to personal relationships, we also rely on institutional relationships. We do not think twice about walking into a bank we have used for years and depositing money—even if we have never previously encountered the particular teller who conducts the transaction. We trust that the bank would not allow just anyone to stand behind their counter collecting money. Likewise, when we give our credit card to a cashier in a well-known store we are trusting that the credit card number will not be subsequently used fraudulently. While certainly our trust in various relationships is sometimes mis-placed, the general notion of relying on our social network and the relationships within it has proven sound.

Establishing relationships within the electronic realm of networks leaves us without many of our traditional tools for building, sustaining and recognizing relationships. We cannot use visual or auditory cues to verify that we're talking to a friend. Rather, a message that is just words on the screen could just as easily come from someone we know or an imposter. In fact, spam email that comes from spoofed email addresses underscores this point. We cannot use the location of a store or the fact that it is logistically difficult to get "behind the counter" to help us trust that the web site we are dealing with is legitimate. In fact, within the electronic realm it is quite easy to copy a particular site and masquerade as a legitimate service, since doing so only requires copying of files and not more high-stakes actions like taking over real-estate or getting someone "on the inside" of an institution. Another key reason such masquerading is relatively easy within the network domain is that identity is built on insecure *names* rather than cryptographic secure principles.

That said, sometimes we can leverage the *general context* in which we operate to find cues in electronic communication to gain confidence that we are in fact communicating with known and trusted people or institutions. For instance, an email from a neighbor might reference a backyard conversation that would be difficult for an imposter to know about. Or, when communicating with "our bank" we might note that an account balance is correct or that an expected direct deposit from our employer does in fact show up in our account. These bits of out-of-band information are not generally known and therefore lend credibility to the party with whom we are communicating. However, there is little way to tell if some man-in-the-middle is passing information between two actors to conduct a legitimate transaction, meanwhile stealing credentials such that a later fraudulent transaction can be undertaken. Simply because the user is presented with valid information can belie the fact that their communications are being intercepted.

The over-arching goal in the proposed project is to consider how we might be able build and utilize relationships as first-class citizens within networks to provide better overall usability, security and trust in the system, instead of relying on ad-hoc solutions for small pieces of the larger problem. Ultimately, we intend to develop an architectural framework and components that allow users and institutions to develop trusted relationships within digital communications that can be viewed and utilized much like relationships outside of the digital communications realm. Our project will consider leveraging relationships for a variety of tasks, such as access control, service validation, and naming, moving past a simple transaction-oriented network to a system that is able to provide general linking of activity across time and exposing legitimate assistive services.

Our view is that there will not be a single way to define and use relationships within a network, but rather developing the overall concepts will allow for their application at various points and in various ways. For instance, for some tasks a simple user-local tracking of relationships may suffice, while other tasks may require broader exposure of relationships across many users and network components. Another dimension is how relationships are created. We expect some relationships to materialize implicitly and others will require explicit action from users and administrators. Finally, we note that while we may design complex relationships and mechanisms under-the-hood, we understand that the average user is not going to understand the notions of cryptography, key management, signing keys and the like, which will be needed to create identities on which relationships can be established. Therefore, we will strive to abstract these aspects of the system away from the users. That said, users will need an understanding about some of the mechanisms we intend to develop, but this likely doesn't need to go beyond intuitive notions such as "establishing secure communication with my bank" . We, therefore, note that we are keenly aware that how we approach various networking problems in a relationship-oriented fashion will involve navigating the inevitable tussles between security, usability and privacy concerns—none of which should be taken lightly.

This project will meet a number of the over-arching FIND requirements, as follows:

- We will develop fundamental architectural designs that will enable more secure networks not in the traditional way of making the networks themselves more secure, but by building trust into the sources of the data through the use of relationships. Our designs will naturally encompass the tussle-spaces between the benefits of leveraging relationships to increase the robustness of networks and the potential costs in terms of privacy issues. Our work will not seek to impose a particular approach, but expose this tussle-space for users, administrators, developers, and institutions to tackle.
- As part of developing relationships we will have to ensure that a notion of cryptographically strong identity is pervasive in the future Internet architecture. Apart from establishing a network of relationships, the notion that strong identities are available throughout the architecture is of major benefit.
- While we do not seek to change the way networks (current or future) deliver data, we will add a fundamental capability to move *users* away from dealing with *unique* names in favor of *usable* names that are context sensitive within the user's particular context. This ultimately will aid the robustness of information dissemination.
- By allowing users to control their own namespaces we also change the economic equation users often face. Since a user's identity is now often mixed up with their service provider's name there exists a strong disincentive to switch providers. This creates a situation whereby names can hold a user "hostage". Using our proposed naming scheme, users can naturally advertise names that they fully control and can easily move across service providers. Ultimately, this will stimulate competition based on performance and not artificial structures like names.

2 Identity

Before we can develop relationships within a network architecture we must gain a sense of *identity* of the various actors in the system. The traditional Internet view of identity has been through unique names given to various resources. For instance, we name users through such identifiers as email addresses, account names, instant messaging IDs, etc; hosts through MAC addresses, IP addresses and the Domain Name System; and files through Uniform Resource Locators (URLs) and share drive names. These names become de-facto identities. For example, the email address "user@institution.com" is at a basic level simply a rendezvous point whereby a sender can leave a message for some recipient. However, this resource name essentially becomes the recipient's identity within the email system. There are two main problems with this state-of-affairs:

- First, the loose notion of identity sketched above offers little security. For example, we know that legitimate email addresses are often spoofed to source spam email.[1] Likewise, DNS names can be hijacked. Further, many phishing schemes employ subtle tricks that make an illegitimate name appear legitimate at first glance (e.g., playing on the number one appearing similar to a lower-case "l").

- When using names as rough forms of identity, changing jobs or service providers essentially changes a resource's *identity*. However, thinking about network interactions based on relationships suggests that this notion of losing and gaining identities is wrong and that a better notion is a stable identity with a *changing set of relationships*. For instance, people do not change their names when they change their jobs or service providers, however, they do change capabilities—e.g., users will likely lose access to one set of company files and gain access to another. In addition, while they will no longer be able to relay mail through an old service provider's mail server, they will gain access to a similar resource within a new service provider's environment.

  Changing identities (with one simply going away) has the effect of severing all relationships with the previous identity. This means that established relationships with the new credential will have to be built anew (e.g., one's social network will have to update their address books). However, retaining identity across changes in the underlying arrangements (employment, etc.) also has a cost in terms of adjusting relationships. For instance, when moving to a new job, a user may wish to exclude fine-grained access to their calendar from people from their previous employer and may wish to grant access to people from the new employer. Therefore, we are not arguing that a simple switch from a name-based system to relationship-based system is without cost. We argue, however, that the benefits of a relationship-based system outweigh these costs.

  As an additional complication, name-based identities are often bound to a computer, rather than the user (e.g., access control based on IP address). Thus even when a user's identity is stable, using multiple computers (say, at home and at work) can cause problems when the host's identity is used as a proxy for the user's identity or the user's credentials reside on one computer and are not easily moved to a different machine.

The problems with name-based identity drive us to our first major contribution, which will be to incorporate cryptographic identity throughout the network architecture. Cryptographic identities are more secure than name-based schemes because to use the identity one must possess some form of key or certificate. While cryptographic identities provide a solid basis for instantiating relationships, simply having such identities pervasively available within the myriad of protocols, services, layers and applications that make up a network is of major benefit. For instance, currently users do not often encrypt email to each other or use mechanisms like PKI derived certificates [19], IPsec [22, 20] and PGP [5] because of the headache in doing so (which requires both understanding the process and collecting keys from their friends—who also have to understand the process). However, since we envision establishing cryptographic-based identities for building relationships, these identities can also be leveraged to encrypt communication, thus adding stronger privacy, authentication and integrity to data transfer when compared with traditional networks.

[1]In fact, one of the PIs of this proposal apparently routinely "sends" spam to his colleagues!

In many ways, we envision that the ability to create and use cryptographic identifiers should be akin to how TCP provides for reliable communication or TLS provides confidentiality and integrity to a wide variety of applications. By embedding the notions of identity into the network architecture, multiple applications can benefit.

While we strive to move past name-based identifiers, we do not discount the use of names for usability. Therefore, our system allows names to be associated with identities, but names effectively act as a user-convenience, and may often be local in scope.

Building relationships on top of cryptographic identity has many benefits in terms of establishing trust in the system, but such a foundation also some drawbacks. First, we recognize that relying on deep understanding of cryptography from users would doom our work to failure and therefore our intent is to abstract the cryptography from the users. Second, we must consider how to handle compromised identity in the form of stolen keys—as, in the large, this will happen with regularity. With these two aspects in mind, we next offer several ways to manage identity.

Key Fob: [2] As a user-focused identity management scheme, we intend to prototype a small USB device to hold user credentials and aid in the authentication process. (Co-PI Weaver has extensive experience with prototyping hardware.) Such a device would be similar—in form and function—to standard car or house keys that people already rely on to access their physical property. The device would go beyond simply holding the user's credentials such that they could be handed over to the attached host. Instead, the USB device would never relinquish its keys, but rather would sign and/or encrypt data itself. The user can thus employ the USB device on untrusted machines without fear of their keys being stolen in the process. This is akin to people's general expectations with (say) a house key—which is not compromised by accidently inserting it into the wrong lock. In addition, such a device allows users to easily move between hosts without losing their credentials and therefore their relationships.

Such a device also reduce the chances of a key being compromised by an attacker because (i) the device is simple enough to be hardened against attack, (ii) the user's credentials will be offline when the user is offline—which reduces any window of vulnerability and (iii) users largely understand how to secure their physical keys. Finally, we envision this device would simply serve up its public key(s) upon request. Therefore, users could establish trust with brick-and-mortar entities (e.g., a bank) by plugging their device in when setting up an arrangement (e.g., a new checking account). This provides a concrete bridge from real-world relationships to digital world relationships.

We envision the key fob being used across a range of applications from simple "user presence" applications such as decrypting email to "user consent" application such as validating a financial transaction. For this latter class, we note that phishing attacks have already leveraged compromised systems to modify financial transactions even in the presence of one-time password systems (e.g., RSA SecureIDs [38]) [37]. Therefore, we will add a small speaker and button to the key fob to allow services to have a direct path to the user for verification that does not rely on any particular intermediate host. In this case, the service creates an audio file encompassing the request (e.g., "transfer $1,000 to checking account 432-41-23145 belonging to John Q. Smith") and a nonce. The audio file is sent to the recipient's key fob (via their current host). The fob plays the message via its built-in speaker. When the user pushes the button to consent the fob cryptographically signs the audio file and returns it to the service. Now user consent for the unique transaction can be assured only if the user is in the physical possession of the fob, even if the user's identity is compromised by a man-in-the-middle.

[2] Due to a change in scope of this project we are not developing the key fob discussed in this proposal as part of this project. We intend to design and prototype such a fob in a related and synergistic project.

A key fob as we have sketched is susceptible to physical theft. However, without additional information about the users, which services they use, etc., it would be very difficult to use fraudulently.

We believe that designing and building such a device can have security benefits on its own, as well as providing strong identities that can be a key *enabler* for the relationship style of networking we sketch in the next two sections, which requires a sound and usable notion of cryptographic user identity.

Opportunistic Identity: We don't *require* the presence of a physical key fob to establish identity and therefore form relationships. We could use a software approximation of the fob—with additional security issues. In addition, particular applications could manage their own forms of identity and relationships (as discussed further in § 3). A key way to abstract identity management from the user will be through the use of *opportunistic* identity [4]. Using such a methodology means that identity creation and use happens on an as-needed basis during the course of normal transactions.

For instance, a user's mail tool could generate a key-pair on a user's behalf the first time the tool is executed. Subsequently, the tool would always sign outgoing email and append the public half of the opportunistically created key. This could in turn be used by the recipient to establish a rough notion of identity that grows stronger as the sender's *track record* shows them to be legitimate. This process abstracts the cryptography from the user, yet provides a more sound version of identity than current naming-based identifiers in common use.

Identity Migrations: Our notion of identity and relationships within networks is that they will likely be long-term in nature—as in the physical world. Therefore, one of the key aspects of identity in our overall architecture is that "identity" should not equate to a single certificate or key-pair. Rather, the notion of identity for a given actor should be able to *span particular credentials*—which are simply instantiations of their identity. This is akin to people getting new passports or driver's licenses every few years, such that the credential includes a current picture and ancillary information (e.g., address, that a person now requires eyeglasses to drive, etc.). A person's identity remains consistent across a series of credentials. Likewise, cryptographic identity can also require updating for a number of reasons, such as key sizes and algorithms becoming out-dated (and thus insecure), keys being compromised or simply lost, organizations wishing to change keys as their staff turns over, etc. Therefore, our intention is for the identities in our system to include the notion of credential roll-over as a first-class component. This is a natural part of many cryptographic systems whereby an older key can sign a newer key as its replacement. For example, this is commonly considered good practice in PGP to both self-sign your current key and sign any successor keys [40].

One particularly powerful form of roll-over is establishing a "backup" identity that is advertised at all times. That is, identity would encompass a primary identity, I_p, and a backup identity, I_b. The public keys for both I_p and I_b can be advertised as the identity of the service, but only I_p is ever used. The secret half of I_b in fact would be kept safely offline such that its chances of compromise are very low. If I_p is found to be compromised, I_b could be put into place without sacrificing the relationships built with I_p. The key to this system is that I_b is established and advertised as a matter of course before the primary key has been compromised. In addition, the *use* of I_b acts as a signal that I_p is no longer valid. This sort of advanced planning allows for identity and relationships to persist, even in the face of an adversary.

Leveraging Relationships: Finally, we note that relationships can aid the problem of establishing identity. For instance, a new employee may be recognized by the company as an employee in some fashion—which will allow the company's users and systems a concrete notion that the person is in fact an employee. Such attestations can be used to bootstrap both a user's trust that a new person is legitimate and/or that some new cryptographic identity is in fact part of a long-standing relationship.

3 Architectural Approaches

Establishing identities, per the last section, is a building block to establishing relationships between actors within the network. As discussed in § 1, our touchstone is to enable network communication to enjoy the same sorts of relationships as are pervasive in the physical world. Our goal is to enable users to gain the same sort of confidence in services as they have when they walk into a physical store or talk with a friend on the phone. While we aim to vastly increase the security of future networks when compared with current networks, we do not aim for airtight security, as this would likely impose too much hassle on users. In this section, we discuss several architectural considerations that will be key in our overall design of a network architecture that leverages relationships as first-class citizens.

We first note that we expect that the *notion* of relationships will manifest itself in a number of places and ways depending on the particulars of individual uses. In this section, we sketch several aspects of the design space that will be explored within our project.

Building Relationships. The process for establishing relationships between entities is a key area for our research. Our overall architectural will permit relationships to be established implicitly or explicitly. For applications where tight access control to some sensitive service is required, explicitly building a relationship between the service and the user is likely the best path. For instance, the relationships between a bank and a patron and the patron and their account would likely need to be explicitly configured and set up. Per the discussion in § 2, such a relationship could be setup via a physical meeting and a user's USB key fob inserted into a reader at the bank. On the other hand, less sensitive services may be able to rely on less formal opportunistic relationships. For instance, an enterprise printer may be willing to accept print jobs from anywhere in the network as long as the actor requesting service has been passively observed to have been physically at the enterprise's location in the recent past. Likewise, connections between people can also be built implicitly or explicitly. In [4] we sketch a system whereby a relationship is established when a recipient finds a particular sender's signed email to be legitimate. Alternatively, users may swap identifiers in order to use each other's namespaces (as discussed in the next section).

One important operation is "user introduction". It is quite common for a third party to introduce two parties. We could introduce explicit introduction messages to enable this form of bootstrapping. Or, we could use a loose and opportunistic form of introductions. A data transmission with multiple recipients could also include signed copies of the known public keys of all parties. For instance, when Alice's email tool knows both Bob and Carol's public keys and Alice sends an email to both Bob and Carol, Bob and Carol's public keys can be included in the message. In turn, Bob and Carol's email clients can know each other's keys.

Another way to bootstrap identity is for users to make a "leap of faith" [29] on the first communication with another user or service. In this case, the user assumes their peer is who they say they are and instantiate a relationship—which can then be verified as time progresses using the peer's track record.

Sharing Relationship Information. Another natural question to tackle is how much relationship information to share with others. In some cases, relationships are used for the express purpose of sharing information (e.g., in the naming case sketched in the next section). However, in other cases the value of sharing relationship information can be viewed as more of a tradeoff. There is likely value in being able to expose a relationship with some service provider to a peer group, as everyone in that peer group can leverage these relationships as recommendations. Further, relationships can also be leveraged as a vote of confidence when trying to determine if an unknown service provider is legitimate or malicious. In addition, relationships can offer some amount of transitivity. That is, for some tasks a user might not want to explicitly authorize every user, but might simply want to scope access (say) based on the social graph. E.g., a personal information service (for instance, sharing pictures with a tool like iPhoto) might be available to all "friends" and "friends of friends". Users who are more than two hops away in terms of the social graph would then be prohibited (unless allowed by exception).

Exposing relationships gives users benefit by leveraging the broader social network. However, users may not wish to expose their relationships too widely as there are clearly downsides and privacy implications. For instance, it is now common practice for employers to read Facebook pages before making hiring decisions. Therefore, we re-iterate a key touchstone that we intend to design for this tussle. Our goal is to leave the users in ultimate control of which information is exposed within each of their relationships. Any system we develop to share relationship information will have rich access controls that users can employ (many of which will rely on the social network itself). That said, exercising fine-grained control may be beyond what many users want to deal with and therefore we will need sensible defaults that respect user's privacy first and foremost.

How to Share Relationship Information. Given that at least some relationship information will be shared, a natural next question relates to how the information will be shared. Will actors that use relationship information be required to manage the information? For instance, would a router be expected to account for its relationships with other routers with whom routing information is swapped? Would an email system be required to retain information about valid senders who should not be subjected to content filtering (e.g., as sketched in [4])? Or, would some institutional or global database capture the myriad relationships that develop?

Application-by-application and protocol-by-protocol handling of relationship information is in some ways tidy. There is no additional infrastructure required and no complicated mechanisms for importing and exporting relationship data. However, such a system suffers from a lack of power in broadly leveraging relationships. For instance, users tend to communicate in a variety of ways (e.g., using IM, email, A/V chatting, sharing disks, exposing calendars, etc.) and therefore leveraging a single relationship across these activities might be worthwhile since it would be much less burdensome than requiring users (or their tools) to build relationships within the confines of each use.

Part of our research will assume sharing is desired and design systems to accommodate this sharing at varying granularities. Here we offer two possible approaches to address these issues, although these are no doubt simple examples and our work will inevitably uncover additional avenues of exploration.

- As discussed in § 4.4, one of the areas we intend to apply relationships is in terms of naming network services and resources in a user- and relationship-centric fashion. The system we have sketched relies on a distributed hash table (DHT) [41, 32, 36, 42, 34] to represent a database of user-configured names. One of the key naming types is a pointer to another namespace. Once such a database exists we can consider using these pointers—likely with some additional information—as a way to construct social graphs. Users can scope the information to only those they deem desirable.

- An additional possibly useful construct for using and sharing relationship information could be a relationship layer. For instance, one could envision this as part of the Silo architectural vision whereby layers can easily be added and composed [11]. Within this layer an actor could expose its identity in quite simple terms (e.g., "I am Steve"), but also express its identity in terms of its social graph (e.g., "I am Steve, a friend of Bob"). A service that does not know "Steve" might decide that since Steve is a friend of some known actor ("Bob") that service will be given.[3]

4 Relationship Applications

While throughout the previous sections we have anecdotally touched on applications of the general theme of relationships within networks, in this section we delve into several specific classes of applications for relationships. We expect that as our project progresses and in particular as we consider relationships in the context of the range of architectures being developed within FIND that additional uses will emerge.

4.1 Access Control

Access control is a central component of today's distributed systems and networks. It permeates all levels and components of the network infrastructure. Access control spans the gamut from low-level access controls on the physical network infrastructure to high-level application specific control. We expect relationships to have an impact on access control at all levels of the network infrastructure. One of the key aspects of relationships is the notion that we move identity from name-based systems to cryptographic-based schemes. Therefore, the first level of enhanced access control simply comes from having stronger notions of identity. Therefore, we can control access to resources based on actors instead of rough approximations such as MAC addresses, IP addresses or DNS names. However, an even richer set of access controls is possible when we consider the relationships built from the cryptographic-identities. Examples of such access controls would be:

- "Allow friends and friends-of-friends to use my wireless network."
- "Allow employees in human resources access to the disk share that holds personnel files."
- "Allow Bob, Jane and Alice access to the shared music on my laptop."
- "Do not apply spam filtering to mail from my friends."

As noted above, these statements are natural language versions of access controls that would be tightly specified in terms of cryptographic identities and the relationships that have been established based on those identities.

Expressing access control policies in these terms offers a number of benefits. First, the policies are expressed in terms of the way people generally work—i.e., with natural groups of people like "human resources employees". Second, these policies are more secure than traditional policies that are expressed in terms of approximations (e.g., an IP address). Finally, since these policies are articulated in a way that people can readily understand they are less prone to errors and more amenable to debugging than a system involving esoteric name-based rules that require meta-information to navigate (e.g., that some IP address actually represents some person's desktop computer).

[3]Recall that all identities and statements about relationships are cryptographically secure and therefore the above version is colloquial and the actual encoding would be verifiable.

4.2 Service Validation

While access control is an important aspect, it is only half the story. As discussed above, service providers certainly need to vet incoming requests to ensure that they are from legitimate users. Certainly a bank wants to gain a solid understanding that a user is the legitimate owner of some account. However, also crucial in this process is that the user would like to understand that they are communicating with the expected service provider and not an imposter. Today's e-commerce web sites use SSL certificates that could be used for this purpose. However, verifying that the right certificate is being used is beyond most users level of understanding. Our approach is to leverage relationships to address the general problem of service validation in several ways:

- We intend to provide a way for users to easily instruct their computers to access known-correct sites, which would make them less susceptible to social engineering attacks that are designed to route users to fake services to steal their credentials. In particular, by changing from name-based identities to cryptographic-based identities we form the foundation of a solid understanding about the service the user is engaging—or, trying to engage.
- Knowing what service a user intends to access also lends itself to letting users encrypt their traffic to the known and trusted service such that imposters will be unable to steal or utilize the given information.
- Leveraging history to detect changes, as well as a user's trusted relationships with other users, would allow a quick check on the validity of a particular service, based on the history of a user's peers.
- Using out-of-band information such as a published public key that corresponds to a given service can allow the construction of passwords that will only work for the given remote address. The *pwdhash* scheme [35] uses this notion in constructing a credential from a user's password and the DNS name of the remote host. While *pwdhash* is a slick and pragmatic idea, we would strive to use the general notion that communication will simply fail to work with fraudulent sites without direct DNS coupling.
- We can also leverage face-to-face meetings for mutual authentication. For example, when authenticating a user's key fob to the bank, the token could also sign the bank's public key(s) for future use, which will allow the user to authenticate the bank at the same time as the bank authenticates the user, caching the result for future use.[4]

By combining the above techniques and methodologies we can construct a system that can warn users when they are attempting to connect to malicious or fraudulent services. However, we understand users' general annoyance with warnings and that a general "make the warning go away" attitude is quite prevalent. This is of course unsafe in that ignoring these sometimes important warnings can have dire consequences. Therefore, a goal of this project is to make service validation a necessary part of the authentication process such that if the user is trying to establish communication with an imposter the authentication simply will not work and the imposter will not gain access to any information that will in turn allow it to act as the user. In other words, credentials will be tied to a service in a way that makes authentication fail unless the given service provider is the expected and legitimate provider. However, there are no warnings and authentication only fails when it should fail (i.e., when the service is fraudulent).

[4]Of course, as stated above, while this discussion is here in terms of keys and signatures this will not be the way the system is presented to users. Rather, average users will simply get an intuitive understanding that they are enabling "secure communication".

4.3 Beyond Conversations

In many ways a basic building block of the current Internet architecture is a transport-layer connection between two endpoints. When that connection disappears the application processes on either end are terminated, as well—with any state built up being lost. Increasingly we see applications being able to transcend transport-layer connections—either to span transactions or as a method to cope with failure. For instance, within the realm of web browsing there is wide use of cookies to save state between connections (transactions) and the use of range requests to allow clients to recover from interrupted connections [14]. We see similar notions cropping up in a disparate set of applications. For instance, peer-to-peer applications (such as BitTorrent [8]) can setup a large number of transport-layer connections with a variety of peers over which chunks of data are exchanged. Yet, the transport layer connection has no application-semantics associated with it. Rather, the data is the application's focus and if some connection goes away another can be used to retrieve the same chunk of data.

These examples are of a general class of data-oriented applications [23, 10]. While treating data as a first-class citizen within the network and it's architecture makes sense, we argue that incorporating information about the data's source and the source's relationship to a given actor is also important. To this end, we argue that relationships can be leveraged in three ways. First, identity and relationships can be used to make decisions about what state to retain and for how long. A service may be willing to hang onto state for certain actors for a long period of time, as these actors have shown themselves to not be malicious. For instance, a peer may be more wiling to increase the connection timeout length (say, via the negotiation mechanism given in [12]) for users within some radius in a given social network. Second, a network that makes pervasive use of cryptographic identity and relationships can be leveraged to key stored state, rather than relying on ad-hoc application-specific mechanisms like cookies. Finally, when gathering data from an ad-hoc group of hosts (e.g., within a peer-to-peer network) the data can all be easily tied back to a common originator as the network has a way to codify identity and relationships.

A related issue is that of soft state—which is one of the successes of the current Internet architecture. Soft state calls for end hosts to obtain configuration information from the network rather than being hard-coded with the information. Such a system allows for a large degree of flexibility in terms of both resource control and re-configurability. One can envision relationships driving how soft state is handed out. For instance, a network-layer address lease might be given to a guest for a certain number of minutes, but if the server has a relationship with the requester a lease might be given in terms of hours—based on the understanding that the latter user will likely be around for a long time and is not likely to be abusing the network by unnecessarily consuming resources.

4.4 Naming

Naming in networking has always favored unique naming over usable naming. Unique naming is important such that each service and resource has a single unambiguous name. Ultimately, we need the notion that a particular email address corresponds to a given person at a given time. If a particular email address could be a sort of context-sensitive identifier that was ultimately used by multiple people we would likely have a mess when contexts started to cross. On the other hand, requiring users to deal with complicated names (user IDs, hostnames, directory hierarchies, filenames, etc.) to access a particular file is overly complicated.

As a component of relationship-oriented networking, we propose building a naming system that follows the social graph to *alias* resources. A quite preliminary sketch of such a system is [2]. In this system a resource has a unique name, but also can be aliased in a context-sensitive way by each user. Users can then expose their names to their social networks and this in turn provides ways to share information. For instance, someone could set the name “babysitter” in their personal namespace and expose the resource to a friend who is in need of child care. This name would ultimately turn into (say) a unique email address of a babysitter. This “personal naming” allows users a number of benefits in terms of dealing with the system:

- First, users are able to deal with the system without using complicated unique resource names—a major win in terms of usability.
- Second, names can easily be shared within a user’s social network—providing a context-sensitive naming system within groups of people.
- Third, access to names can be controlled in intuitive ways. For example, if only “friends” are allowed to access “babysitter”, cryptography can enforce this restriction transparently to the user because the set of “friends” includes information about every friend’s keys.
- Finally, this method of resource naming allows users more control in their relationships with their service providers since it ultimately enables provider-independent resource names. For instance, a user can set a name like “email” within their own namespace to point to their current email address. When their provider (e.g., employer) changes they can simple point the “email” name to their new address, but people in their social network can continue to simply employ the user’s “email” name to address email. This gives users the ability to choose their service provider relationships based on service level, features, etc. rather than simply being tied to a service provider to retain a given name (identity) in the system. This provides users with considerable leverage to manage a problematic tussle-space in networking.

4.5 Exposing Adjunct Services

A final aspect where we intend to leverage relationships is in terms of exposing related services or resources. Network services are often composed of a number of independent components that are designed to work together in some fashion. For instance, modern web browsing often consists of information from a variety of sources being displayed as a unit to the user (e.g., from various servers within a farm, from content-distribution networks such as [1, 39, 24, 31, 21, 17], from advertisers, etc.). Allowing such a disparate collection of information sources to indicate a relationship could be used to prevent middleboxes from silently slipping additional or modified content into the stream (as has been noted in the current network [33]). Any future Internet is not likely to move away from this modular approach to implementing functionality. In fact, future networks may be more prone to division of tasks as we are continually gaining a better understanding about how to meaningfully separate functionality.

As an example of coupling of services in future architectures, [3] calls for an “assistant” to stand in for a host when that host is not being actively used such that the host can be put to sleep to conserve energy. The vision described in [3] is that such an assistant will allow the hosts it serves to conserve energy without losing *network standing*. This means that the asleep host may be able to receive phone calls or instant messages—with a gating process run by the assistant on what is allowed to wake the host. Such a system is conceptually simple, but pragmatically difficult. How is some peer supposed to understand that the assistant is authorized to conduct operations on a particular host’s behalf at a particular time? A network that allows for the exposure of such relationships is needed to allow peers to recognize legitimate assistants.

Another area where a system of exposing relationships could help is in terms of ensuring that a particular service’s business partners are well-known and this relationship can be easily followed by a user. For instance, popular auction sites often have preferred payment brokers they recommend their buyers and sellers use to arrange payments. Yet currently, false escrow sites are a common mechanism used in auction fraud. By being able to advertise such a business relationship in a concrete and non-spoofable way, consumers can gain confidence that they are using the recommended (and legitimate) service.

5 Related Work

Our work is related to a number of projects undertaken in the research community, including:

- The community has engaged in vast range of security-related research that touches on some of the aspects we hope to address with relationships (e.g., anti-phishing techniques, anti-virus techniques). While clearly too vast to enumerate here, we note that most previous work has been conducted to benefit the current Internet architecture and usually to address only a single aspect of security in isolation (e.g., phishing). Our approach is to develop structures to advance the architecture itself such that we rely less on bolt-on techniques and more on the built-in elements of the architecture.

- A large number of web portals have been developed around the notion of connecting users with their social network. This allows users to easily expose information and communicate with their friends as a group. Sites like MySpace [28], Facebook [13], and LinkedIn [25] have shown that people find this style of communication to be powerful and useful in their everyday lives. Social networks developed within these sites are only useful within the confines of the given system, and deliberately lack privacy from the view of the social network operator. Our goal is decidedly different. Rather than simply connecting users within some targeted realm, our goal is to build the notion of relationships into the very fabric of the network to allow the use of relationships broadly and deeply across protocols, layers, services, components and applications.

- Google's OpenSocial [18] aims to take social networking outside the specific realms developed at individual sites like Facebook or MySpace. Using OpenSocial, one can establish a social network that is generic across applications and web sites. A variety of applications have been developed around the OpenSocial API that allow users to do a variety of tasks from playing games with their friends to sharing book recommendations. OpenSocial's goal of allowing users to build social networks and applications that leverage these social networks in a generic fashion that is not tied to any particular web portal has similarity with our approach. However, we offer a number of advantages beyond OpenSocial. First, OpenSocial is focused on applications, whereas we envision incorporating relationships pervasively throughout the network architecture. For instance, on the opposite end of the spectrum from applications we could envision that relationships could be used to gain access to the wireless network in a friend's house. In addition, OpenSocial is focused on relationships between people, whereas our work is broader such that relationships can be developed between and with people, institutions, hosts, networks, services and resources.

- There has been considerable interest in ease of use in cryptography, such as SSH's "leap of faith" [29], Durfee et.al.'s Usable PKI [9, 6], and techniques for leveraging small trusted devices [30]. However, most of these projects are not concerned with creating a consistent and global cryptographic identity which can be used to convey relationship information but securing individual transactions in an easy-to-use fashion.

- We also note that the driving theme behind the Host Identity Protocol (HIP) is that separate identifiers are needed for a host itself and its *location* within the network [26, 27]. For this latter, an IP address suffices. However, there are valid reasons to also have a concrete notion of the former (e.g., for access control, state retention across location changes, etc.). While the driving spirit behind HIP is in some ways similar to our proposed system, we also seek to broaden the notion of decoupling identity from the details of data transfer beyond the host/location dimension to also encompass users, data and services.

- Finally several projects have considered how to leverage social networks in the context of networks and network architecture. For instance, MIT's Unmanaged Internet Architecture (UIA) [15, 16] project focuses on architectural structures to name personal devices (e.g., "Jane's phone"), introduce devices to each other and ultimately route traffic across a set of devices that encompasses a user's social network [15, 16]. In addition, the work described in [7] is also concerned with routing across social networks. Our work is distinct from these efforts in that (i) our naming scheme works with all current and future namespaces and is not scoped to only hosts/devices and (ii) while the previous work has focused on routing across social networks we concentrate on leveraging these networks for a broad set of uses (access control, moving beyond transactions, service validation, etc.) to build the kinds of social structures in the network that are prevalent and useful outside of networking. We do not preclude using the relationships formed within our architecture from being used for routing information, but it is not our primary focus.

While these projects described above (and other similar efforts) have lead to useful technology, we believe that taking an architectural approach that does not pre-suppose the purpose or the scope of how relationships can be utilized will ultimately be a valuable contribution to the future Internet architecture.

References

[1] Akamai Technologies. http://www.akamai.com/html/technology/index.html.

[2] Mark Allman. Personal Namespaces. In *Proc. HotNets*, November 2007.

[3] Mark Allman, Ken Christensen, Bruce Nordman, and Vern Paxson. Enabling an Energy-Efficient Future Internet Through Selectively Connected End Systems. In *Proc. HotNets*, November 2007.

[4] Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, and Nicholas Weaver. The Strengths of Weaker Identities: Opportunistic Personas. In *Proc. of USENIX Workshop on Hot Topics in Security (HotSec)*, August 2007.

[5] Alma Witten and Doug Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Poceedings of the USENIX Security Symposium*, 1999.

[6] D. Balfanz, G. Durfee, R. Grinter, D. K. Smetters, and P. Stewart. Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute. In *Proc. Usenix Security Symposium*, 2004.

[7] Lerone Banks, Shaozhi Ye, and Felix Wu. Davis Social Links: Integrating Social Networks with Internet Routing. In *ACM SIGCOMM Large Scale Attack Defense Workshop*, August 2007.

[8] Bram Cohen. The BitTorrent Protocol Specification, http://www.bittorrent.org/beps/bep_0003.html.

[9] D. Balfanz and G. Durfee and D. K. Smetters. *Making the Implossible Easy: Usable PKI*, chapter 13, pages 319–333. O'Reilly Media, 2005.

[10] Michael Demmer, Kevin Fall, Teemu Koponen, and Scott Shenker. Towards a Modern Communications API. In *ACM SIGCOMM HotNets*, November 2007.

[11] Rudra Dutta, George N. Rouskas, Ilia Baldine, Arnold Bragg, and Dan Stevenson. The SILO Architecture for Services Integration, control, and Optimization for the Future Internet. In *Proc. IEEE International Conference on Communications (ICC)*, June 2007.

[12] L. Eggert and F. Gont. TCP User Timeout Option, November 2007. Internet-Draft draft-ietf-tcpm-tcp-uto-08 (work in progress).

[13] http://www.facebook.com/.

[14] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. RFC 2616.

[15] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Fans Kaashoek, and Robert Morris. User-Relative Names for Globally Connected Personal Devices. In 5^{th} *International Workshop on Peer-to-Peer Systems*, February 2006.

[16] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. Persistent personal names for globally connected mobile devices. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, November 2006.

[17] Michael J. Freedman, Eric Freudenthal, and David Mazires. Democratizing Content Publication with Coral. In *USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2004.

[18] OpenSocial Google Code, http://code.google.com/apis/opensocial/.

[19] P Gutmann. PKI: It's Not Dead, Just Resting. *IEEE Computer*, 35:41–49, August 2002.

[20] R. Housley. Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP), December 2005. RFC 4309.

[21] Intelligent content distribution service. AT&T Inc. http://www.business.att.com/service_fam_overview.jsp?repoid=ProductSub-Category&repoitem=eb_intelligent _content_distribution&serv_port=eb_hosting_storage_and _it&serv_fam=eb_intelligent_content_distribution &segment=ent_biz.

[22] S. Kent and K. Seo. Security Architecture for the Internet Protocol, December 2005. RFC 4301.

[23] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, and I. Stoica. A Data-Oriented (and Beyond) Network Architecture. In *ACM SIGCOMM*, 2007.

[24] Limelight networks content delivery. http://www.limelightnetworks.com/contentdelive ry.html.

[25] http://www.linkedin.com/.

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

[27] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol, October 2007. Internet-Draft draft-ietf-hip-base-10.txt (work in progress).

[28] http://www.myspace.com/.

[29] OpenSSH Secure Shell, http://www.openssh.org.

[30] A. Oprea, D. Balfanz, G. Durfee, and D. K. Smetters. Securing a Remote Terminal Application with a Mobile Trusted Device. In *Annual Computer Security Application's Conference*, 2004.

[31] RapidEdge content delivery network. http://www.rapidedgecdn.com/.

[32] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM'01*, August 2001.

[33] Charlie Reis, Steven D. Gribble, Tadayoshi Kohno, and Nicholas Weaver. Detecting In Flight Page Changes with Web Tripwires. In *Proc. of the 5th USENIX Symposium on Networked Systems Design & Implementation*, 2008. To appear.

[34] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *SIGCOMM*, 2005.

[35] Blake Ross, Collin Jackson, Nicholas Miyake, Dan Boneh, and John C. Mitchell. Stronger Password Authentication Using Browser Extensions. In *Poceedings of the USENIX Security Symposium*, 2005.

[36] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[37] RSA Alert: New Universal MITM Phishing Kit Discovered, http://www.rsa.com/press_release.aspx?id=7667.

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

[39] SAVVIS content delivery services (cnd). http://www.savvis.net/corp/Products+Services/Digi tal+Content+Services/Content+Delivery+Services/.

[40] Walter Soldierer. Why Should I Sign my Own Public Key?, http://www.iusmentis.com/technology/remailers/selfsign.html.

[41] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM'01*, August 2001.

[42] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, U. C. Berkeley, April 2001.