Sobriquet:

A Personal Naming and Identity Management System

Robert McAdoo

A thesis submitted to the University of Dublin

for the degree of Doctor of Philosophy

School of Computer Science and Statistics

University of Dublin, Trinity College

2012

DECLARATION

Signed,

Robert McAdoo

January 11, 2012.

ABSTRACT

The Internet in its current form lacks an adequate identity infrastructure. Every relevant application must provide its own solution to the problem of authenticating and naming people. Many of these applications share common goals and duplicate this functionality. Compounding the problem is that for every application a person uses they acquire a new identifier and set of authentication credentials.

In some cases people come to depend on these names to be reachable online. When these identifiers change or become unavailable people lose contact with one another. The reliance on these identifiers also encourages provider lock-in. Regulators of the telephone system realised some time ago that mobility of identifiers is important as a means of encouraging competition between providers. Yet on the Internet no system exists to provide this functionality.

The lack of a means to authenticate people on the Internet means that people often communicate with one another insecurely; without the ability to know for sure who they are communicating with and having no means to ensure the confidentiality of their communications. Public Key Infrastructures were once thought to be the solution to this problem, but have so far failed to live up to this promise. They are costly to maintain and are not suitable for all applications where authentication is required. The area of identity management has sought to rectify the authentication problem in recent years, but generally ignores the identifier mobility issue.

In our solution, Sobriquet, we propose a global naming system for people that allows for identifier mobility, and an identity management system that aims to provide a means of authenticating people. Our solution recognises that there is in all likelihood no general solution to the problem of authenticating people. We propose a system of authentication we call "history based authentication" that authenticates a person as being the same individual as was present in a previous communications session. Authentication happens with respect to an identifier, and the history of interactions an entity has with the person that identifier represents influences the notions of identity that this entity ascribes to that identifier.

We argue that this is an adequate level of authentication for many types of online interactions. Our solution also addresses the issue of bootstrapping trust between people who have never met. We do this by reducing the economic incentives for people to engage in undesirable behaviour.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor Prof. Donal O'Mahony. He afforded me the freedom to explore many areas of interest over the course of my PhD. while ensuring I kept myself well grounded. His continued support and enthusiasm for the work was invaluable, contagious, and always appreciated.

Prof. Linda Doyle provided valuable guidance and encouragement making the transition into research, and always encouraged me to get involved in discussions even in areas outside my main area of study.

I would also like to acknowledge the help, feedback, and ideas that stemmed from many of the members of CTVR, both past and present, in particular Patroklos Argyroudis, Raja Verma, Stephen Toner, and Tim Forde.

I'd like to acknowledge the support and encouragement my friends and family afforded me over the years. They made, and continue to make, the difficult times bearable, and the good times all the more enjoyable.

Finally, but by no means least, I'd like to thank Emmanuel Stone who read many drafts of this document, and whose feedback during the write up period was invaluable.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The original use of the Internet infrastructure was rather limited by today's standards existing mainly to share resources, such as expensive mainframe computers, across research and military organisations by connecting their individual local networks into one large global network or "internetwork". These resources were located across heterogeneous networks with convenient protocol layer abstractions allowing each endpoint to be accessed across the network using the TCP/IP protocol suite regardless of its mode of attachment to the network. The different layers of abstraction are depicted in Fig. 1.1. The type of link layer may be different at two endpoints, but the network layer will ensure that data between them is routed to the proper destination. According to the end to end principle [107], which heavily influenced the design, the transport layer at each end point must look after ensuring proper data delivery itself. Individual applications will typically transmit data according to their own protocols. These applications are typically referred to as the application layer.

At each of these layers of abstraction identifiers are used to refer to an endpoint at that layer. At the link level this may be an Ethernet address, the network layer will generally use an IP address, the transport layer a port number, and the application layer identifier will depend on the individual application. We will use the term Application Specific Identifiers (ASIs) to refer to identifiers at this layer generically with examples including email addresses, usernames in web applications, and URIs (Uniform Resource Identifiers).

The most widely deployed name system at the application layer is the Domain Name System (DNS) and is used by most applications in some form or another. Names used

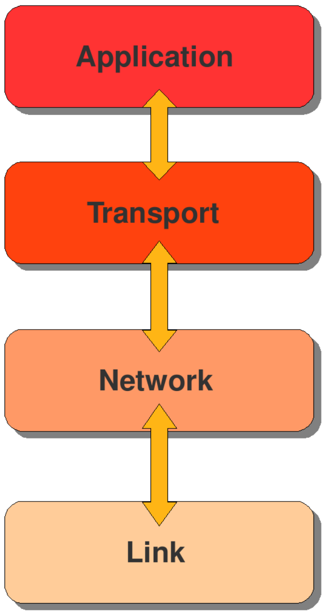

Fig. 1.1: The network protocol abstractions used in the Internet infrastructure.

in this system are referred to as Domain Names and often map to an IP address through a process known as name resolution, which is analogous to a lookup in a directory[1]. Domain names are often used by an application either as an ASI or in the creation of various ASIs. An example of this is the email address, which is often comprised of the concatenation of a username, the "@" symbol, and a domain name.

Since its explosion in popularity in the early to mid 1990's the Internet has been used by more and more people as a communication platform. Many applications that use the Internet provide a means for people to communicate with one another, or facilitate interaction between people and applications. This is increasingly the case nowadays as web applications have taken over the role of desktop applications. Examples of this include web based desktop office suites, finance applications, email clients, website builders, etc. Many of these applications include a social element to provide information sharing. The increase in the popularity of web applications is just one factor that has led to a situation where many people have accumulated lots of different ASIs. Managing

[1] In actual fact the uses of DNS have increased since its original inception, and continue to do so. We will discuss some of these further in the next chapter.

these has become a real problem.

1.1 Problem Statement

The population of the Internet has grown significantly since its early days and there are now many more applications in widespread use. The infrastructure has arguably become critical to day to day communication for many people, but the way they communicate using this infrastructure is still fragmented. People may establish connections via one type of application, e.g. email, but if they then wish to communicate using, say, an Internet Telephony system each needs to communicate their Internet Telephony identifier to the other out of band. Furthermore, the level of authentication each can rely on varies from application to application, and is often inadequate. In a practical sense this means that people who are communicating across the infrastructure are trusting insecure services like DNS to reliably provide the correct destination for their messages, and application service providers such as email service providers to ensure the delivery of their emails.

Kim Cameron, Microsoft's Identity Architect, summarises the problem in his article The Laws of Identity [43] as follows: *"The Internet was built without a way to know who and what you are connecting to. This limits what we can do with it and exposes us to growing dangers. If we do nothing, we will face rapidly proliferating episodes of theft and deception that will cumulatively erode public trust in the Internet"*.

In this thesis we will argue that there is currently no adequate solution to the problem of naming and identifying people, such that people communicating across different applications on the Internet can know who they are "connecting to". This implies providing a unified view of a person's identity across the different applications that they use, and this is indeed the problem we are trying to solve in this work. We will now highlight the need for this with a use case.

1.2 Use Case

A typical scenario that demonstrates a need for our system can be illustrated by way of talking about problems with email. We've chosen this particular communication system because it is used in the large and is familiar to most people. The problems we outline here also apply to other systems that use ASIs to identify people.

Over the lifetime of an email account a person, Alice say, may provide the corresponding email address to a relatively large number of entities. Her family, friends, and acquaintances may use her email address as a means of contacting her, she might provide it as a contact address when signing up to online services, and she may participate on various mailing lists under that identifier. She may wish to communicate with each of these entities in different ways. For instance, she may wish to communicate using telephony with her friends and family, but not with the mailing list participants.

The problem however is that access to her chosen email address may not persist for her lifetime. There are a number of reasons for this. For example, the domain hosting her account may expire, the organisation that provides the service to her may terminate her account, she may decide to retire the email account due to excessive spam or to access a better service at a different provider.

Once her email address ceases to be used by her its future is uncertain. Another entity may acquire the expired domain and eavesdrop on email sent to her account, the organisation providing the account may reallocate it to somebody else, or she may choose not to forward mail from it so that her spam problem does not plague her at her next account. Optimally she would have a mechanism for allowing those with whom she communicates to know that her email address has changed, while remaining consistently identifiable as Alice so that she remains contactable by them.

Since Alice's email address can change at any point she should be able to assure those with whom she is communicating that any given message came from her. However, it is undesirable for this to be done in a way that could infringe upon her privacy.

For instance, if Alice identifies a message as coming from her to a recipient, Bob say, then that is for Bob's benefit only and it should not be possible for a third party to determine unequivocally that this message came from Alice.

1.3 Requirements

We believe that any solution that seeks to identify people will need to meet the following requirements:

1.3.1 Provide a way to identify people

Had Alice a way to consistently identify herself no matter what her email address then she could change email address and still be identifiable as Alice to her contacts. Our first requirement is to provide such a way for Alice to be identifiable as she changes the Application Specific Identifiers, such as email addresses, that she uses. This way, no matter where she emails from she will be recognisable as Alice.

1.3.2 No Trusted Third Parties for Name Allocation

Alice's email address is tied to a domain name and an account she has with the owner of that domain. If this domain owner decides to cease providing service to her then she loses access to that identifier. This is true of any identifier that is provisioned by a third party. In order to be able to provide a consistent name for Alice, relying on third parties to identify her will always have this problem.

1.3.3 Allow Changing of ASIs

Further to being identifiable as Alice, people she communicates with also need to be made aware of her new email address if she is to be able to continue to be contactable after she changes it.

1.3.4 Privacy

If Alice is to be given a global name, under which she communicates then this shouldn't be a way for all her actions to be linkable as this would be a privacy violation. A solution to this problem should ensure that it allows Alice to prove who she is uniquely and unambiguously to whomever she is communicating with, but does not provide third parties with a means of tracking her.

1.3.5 Bootstrapping Secure Communications

If two people Alice and Bob do not have a means of communicating then they should be able to bootstrap one. If creating identities is cheap then there needs to be a way to provide assurance to one another that the identity they are claiming is not just one of many, but is tied to some real world attribute.

1.4 Contributions

The main contributions of this thesis are:

1. An analysis of the identity issue on the Internet infrastructure, with an identification of the issues involved and the requirements that a solution to this problem must fulfill.

2. A naming system designed for allocating names to people, storing and resolving mappings between the allocated names and a set of Application Specific Identifiers, and providing a means of managing these mappings.

3. An identity management and authentication infrastructure that prioritises the privacy of its users.

4. A novel set of authentication protocols that allow a person to authenticate themselves anonymously while preventing abuse of anonymity.

5. A novel approach to preventing the misuse of anonymity that relies on reducing the economic incentives to engage in malicious behaviour, such as spamming for example.

1.5 Dissertation Overview

The rest of this dissertation is structured as follows.

The second chapter provides an overview of the state of the art of identity management, authentication, and naming systems. We will analyse where they contribute ideas to solving the problem we wish to solve and where we feel they are lacking. This analysis informs our design. We will also outline existing technology that is used in our design.

Chapter three outlines the design of our system, which details its functionality and high level operation. We begin by performing a detailed analysis of the problem before outlining our design. This chapter will give the reasons for decisions we have made based on our observations in the preceding chapter.

The fourth chapter then addresses implementation specific issues. We give a description of our proof of concept implementation.

Finally, the fifth chapter concludes the dissertation and provides a direction for future work.

2. RELATED WORK

Our solution to the problem discussed in Chapter 1, which we call Sobriquet, includes two main components: one that provides a naming service suitable for assigning names to people and mapping those names to a set of results, and a component that provides identity management and authentication to Internet systems used for personal communications.

This chapter will first look at global naming systems that are in use today and will outline why these are inappropriate for the task of naming people. By showing how these systems are unsuitable for the task we will demonstrate a need for a new naming system. We will then examine the two main components of any naming system: the name resolution and name allocation components, devoting a section to discussion of each. We will first examine the type of name resolution systems in use today in Section 2.1, before progressing to debating the merits of a number of alternatives that have been proposed in Section 2.2, while Section 2.3 will explore a number of different name allocation systems. Finally we will conclude our round up of the naming issue with a look at systems that have attempted to address the issue of Personal Mobility, where naming is used as a way of allowing people to be mobile across different systems and devices.

The rest of the chapter will deal with issues pertaining to identity management, and authentication. We will examine existing approaches to these issues and where we believe they are ripe for improvement. This discussion will divide into three sections. Section 2.5 will discuss Identity Management, the area of research we feel is most relevant to this discussion. The following section (2.6) will discuss cryptography and technologies from that area that have attempted to solve the issue of global authentication. Finally, since an important requirement of our solution is to prioritise privacy, we will examine some Privacy Enhancing Technologies and discuss how we feel they may contribute towards solving our problem.

2.1 Naming Systems

Lampson defines a naming system (name service) to be that which *"maps a name of an individual, organization, or facility into a set of labelled properties, each of which is a string"* [82]. This set of labelled properties is often referred to as the results, or the result set, and the process of mapping from the name to the results as name resolution. The purpose of most naming systems is to allow the result set to change over time, while allowing it to be accessed through use of a constant identifier, the name. Most naming systems can be reduced to the following components:

1. the facility to store names in a database and map them to their corresponding result sets (name resolution),
2. means to store and control updates to the result sets (access control),
3. support for policies in relation to governance of the namespace (name allocation).

Each of these functions are typically influenced by the structure of the namespace. In the literature we have encountered two main structures: the hierarchical, and flat namespaces. A hierarchical namespace is comprised of multiple levels of logical "sub-namespaces". Each of these sub-namespaces can be considered as a leaf in a tree structure that spans from the root or the top level of the namespace. They each form a logical partition within the namespace. This allows hierarchical namespaces to scale in terms of the number of names the namespace can support while keeping the depth and breadth of the namespace tree at manageable levels. Partitioning of the namespace allows delegation of responsibility for governance of different parts of the namespace to different entities.

The partitioning of names also provides a logical way of separating the physical storage of name to result set mappings. Just as sub-namespaces may be administered by different entities, their mappings may also be physically stored on different machines. As we look at different implementations of hierarchical namespaces we will see the advantages and disadvantages of this method of partitioning the data in a namespace.

In contrast to hierarchical namespaces a flat namespace contains no such partitioning. Each name is atomic and belongs to the same namespace as every other name. These types of namespace are typically administered by a single entity, as no means of delegation of governance exists. We will look at some implementations of both of these types of naming system in the following sections.

2.1.1 The Domain Name System

The Domain Name System (DNS) came into existence in order to replace the existing Internet naming scheme, which consisted of a single text file known as the *hosts.txt* file [91]. This was stored centrally and downloaded periodically to individual machines via FTP. Eventually the number of hosts in the network grew to such a size that administration of the file became impractical [92]. Furthermore, since the file was centralised there was no way for network administrators to easily allocate names local to a given network. DNS specifies a hierarchical distributed database of global names such as *www.example.com.*. Since the original specification was released in 1983 [93] [94] DNS has become the global naming system of the Internet infrastructure, and has undergone several extensions in its lifetime, including changes to the message format [116], support for dynamically updating entries [117] [120], and extensions for authenticating DNS results [20, 22, 21].

Domain names, like most hierarchical names, are allocated top-down, in a manner that reflects how administrative entities in the hierarchy delegate control over a subset of the namespace to other entities starting from the root and moving down the hierarchy. The root node, represented by a '.' appended to domain names but often omitted in actual use, can create domains under the root node, for example *.com.* and *.ie.*, and delegate responsibility for maintaining that subset of the namespace to third parties. The entity that is allocated responsibility for administering that portion of the namespace may in turn choose to delegate responsibility for administration of a portion of its subset of the namespace to other parties and so on. Domains in the first level of hierarchy under the root do not form full domain names by themselves and are referred to as top-level domains (TLDs). The domain names that are available to the public for purchase will generally live in the third level or lower of the hierarchy, i.e. customers will generally purchase domains under a TLD. Parts of the namespace that are under administrative control of a single entity are referred to as zones. The root zone is administered by the Internet Assigned Numbers Authority, which in turn is operated by the Internet Corporation for Assigned Names and Numbers (ICANN). Top-level domains are maintained by a mixture of public entities such as government owned organisations, and private entities such as Verisign, which are responsible for administering the *.net* and *.com* domains.

There is a separation between responsibility for governance of the DNS namespace and responsibility for infrastructure. Mapping between names and their corresponding result sets, known as Resource Records (RRs) in DNS, can be assigned to entities other than those responsible for governance of the allocation procedures. Since the namespace can be carved up at each point in the hierarchy, these subsets may be physically stored on any number of different servers in any number of different geographic locations. DNS operates as a single database, though its distributed nature allows for replication to mitigate failures of any one part.

Each RR has a specified type including A records, NS records, and MX records, which specify an IP address, a name server, and a mail server respectively. When a

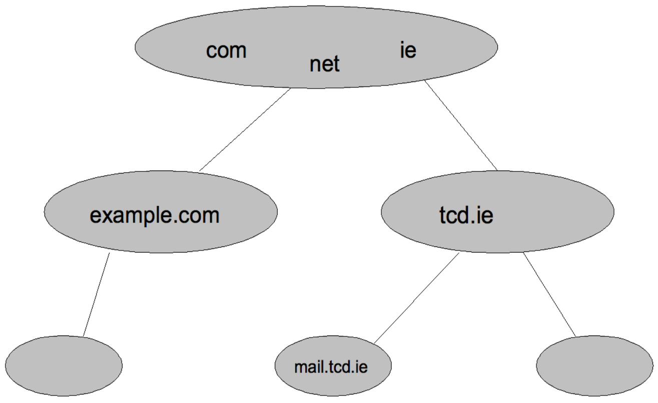

Fig. 2.1: The domain namespace hierarchy

name lookup (resolution) is performed, a query will specify the type of record that is to be returned. Many other RR types exist and new types are defined as needed. For example, to support the mapping of domain names to URIs (Uniform Resource Identifiers), the NAPTR record has been introduced. DNS can therefore be used as a directory or database for a variety of purposes and has grown beyond its original intended purpose of just mapping domain names to IP addresses. Some naming systems have used this functionality to take advantage of the infrastructure provided by DNS; we will outline some of these systems in Sections 2.1.4 and 2.3.1.

Resolution of a name such as *example.com*[1], begins at the root and follows a path of name servers specified at each point in the hierarchy as the next one to query by an NS record. So, for example, querying the root for *example.com* will result in an NS record for at least one *.com* top-level domain name server, which in turn will return an NS record that specifies at least one name server responsible for answering authoritatively for records for that domain.

The distributed database of DNS is made up of a number of nodes called name servers. These name servers may be authoritative, which means that they have been

[1] Assuming no caching.

configured to store and return results for a certain domain or zone. These name servers may be replicated through a master and slave setup where the master will be updated by an administrator and the slaves will periodically update themselves from the master. Each server may answer queries thereby achieving redundancy.

Typically a device connected to the Internet will have some software known as a resolver that is capable of talking to name servers. A name server will perform the actual queries on behalf of the resolver. To answer a query authoritatively the name server will perform a recursive resolution where it starts by querying at the root, and will continue to progress down the hierarchy of name servers specified by each NS record it receives. When the record type it is querying for is returned the name server will stop and return that as its result to the client.

Due to the fact that each of these queries may take a relatively large amount of time, a name server may answer non-authoritatively. This is where a result is returned for a server that is not authoritative for that domain. This allows caches to be deployed to store results and allow commonly queried names to be resolved more quickly. Results may be cached for a period of time specified by a TTL (Time To Live) value in the result of a query, which indicates the maximum length of time for which the results should be considered valid.

Since its original specification a number of extensions have been made to the DNS protocol. These have been necessary as more and more functionality has been demanded of the infrastructure. For instance the UPDATE function has been added in order to support dynamic updates of the result records [117]. This is needed to allow hosts that do not have static IP addresses to update their address as it changes. One of the original design goals for DNS was that both queries and replies should be small enough to fit in a single UDP packet, though TCP could be optionally used for queries. Since then the number of different types of RRs has increased and fitting results into a single packet may no longer be viable. For this reason EDNS [116] was proposed in 1999 which allowed larger packets to be used.

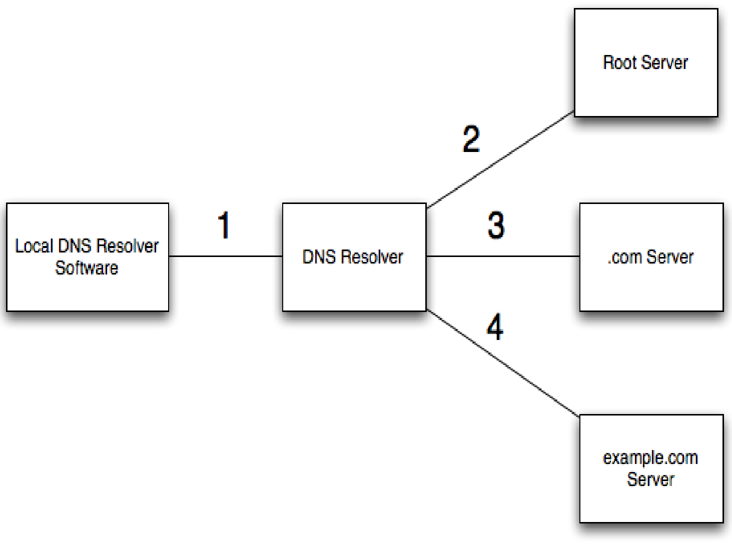

Fig. 2.2: An example DNS query

DNSSEC

For some time now the need for authenticated DNS results has been recognised. Currently most DNS results are unauthenticated and name servers are trusted to report results correctly. This leaves the name system vulnerable to a man in the middle attack where an attacker supplies a false result for a given query. This type of attack is trivial to perform when the attacker is on the same local network as their target. Since DNS is so extensively relied upon it should be secured against these attacks. DNSSEC is a set of extensions to DNS [20, 22, 21] that aim to bring security to the naming system. The extensions provide three key pieces of functionality: authentication of data stored in the database, protection of the integrity of DNS data in transit, and authenticated denial of existence of a record.

DNSSEC essentially creates a Public Key Infrastructure[2] on top of DNS. The authentication of results relies on a chain of trust. The public key of the root servers is distributed by platform vendors. The root domain may be queried for the public key record of a TLD, say, the .com domain. Each record type is signed and so the public key for the .com domain may be verified by the resolver. This will continue until the

[2] We will discuss PKIs in Section 2.6.3

resolver has verified the authentication chain of public keys that lead from the root to the domain it wishes to query. The result of a query may then be verified.

DNSSEC brings support for public keys to be stored within the database and authenticated to a domain name DNSSEC. Assuming it is eventually fully deployed, it will allow for authenticated distribution of public keys.

Identifiers using DNS

Other naming systems have been built on top of systems such as DNS as a result of its popularity. We will now look at two of those.

The first is the Uniform Resource Identifier [29]. This is the generic name for three different types of identifier, the Uniform Resource Name (URN), the Uniform Resource Locator (URL), and Uniform Resource Characteristics (URC). These different identifiers perform different functions; the URN acts as a unique identifier for an object, while the URL and URC specify the location and characteristics (metadata) of the object respectively. The decoupling of the location and characteristics of the object from the naming of it allows these details to change while not affecting the way the object is referred to. This information may instead be retrieved by resolving the URN into either a URL or a URC.

While these three different identifiers have different functions, they are linked by a common syntax. The general form of the URI is *scheme:scheme-specific-identifier*. The scheme of a URI defines the namespace in which that name is valid. For example, the URI *sip:user@example.com* specifies that the identifier is a SIP specific identifier, of the form *user@example.com*. The scheme specific component can, in theory, take on any form it wishes. The scheme names are taken from a well known list of identifiers that is allocated by the IANA, though numerous unofficial schemes have been seen in common usage. Since each operating system typically maintains a list of applications that support specific schemes and the default application associated with it, in theory an application can register itself as supporting any scheme it wishes.

Of the three types of URI, the URL is most familiar to people as HTTP URLs such as *http://www.example.com/index.html*. There is no resolution architecture for URNs in common use and as such it is not possible to persistently name and locate objects in the current Internet architecture. One way to achieve this would be to use the Domain Name System. In this way a query could be made by appending a specific domain to the URN and resolving in the usual way. The result set would then include NAPTR records [90], used to return a URL from a query, that point to the location of the resource. The working group for URCs concluded without ever producing a standard for URN resolution. Since then the need for URCs has largely been made obsolete by other metadata standards such as the Resource Description Framework [9].

Typically URIs such as those used in the web or SIP systems use the DNS to locate the host of the data. Arguably they have been one of the driving forces that has ensured the popularity of the DNS as the global name system. The rest of the URL will consist of the protocol to be used to get access to this data and the path, relative to some base path configured on that host, that the data can be accessed at. URLs are familiar to most web users as they are used to uniquely identify all resources in the system.

Perhaps more universally recognised than the URL is the email address. They generally take the form of *user@domain*. The email address is specified for use in SMTP, which is defined in the standards RFC 5321 [80] and 5322 [103]. The email address is essentially a less complex version of a URL, since it specifies a location that a person may be contacted at, while the scheme and port are assumed to remain static or be obtained from the context in which the identifier is used. The simplicity of the email address is a usability boon and a recent usability comparison between the URI and email addresses conducted by Yahoo! suggests that people may find the email address less confusing [15].

2.1.2 Global Name Service

In [82] Lampson describes his design for a global name service that is designed to facilitate resource location, mail addressing, and authentication in a large scale distributed system. This system was designed as a successor to the name service used in the Grapevine system [32] and formed the basis for the name service used in the Open Software Foundation's (now The Open Group) Distributed Computing Environment (DCE) [12]. The goals for the system were as follows:

- Scalability - the system should support a very large number of names and administrative organisations.
- Longevity - the system should be designed to be around for a long time with changes to the individual components and organisation of the namespace occurring over its lifetime.
- High availability - the name service is critical infrastructure and as such should be available for use.
- Fault isolation - local failures shouldn't cause the entire system to fail.
- Tolerance of mistrust - not every component will be trusted by every client; the system should accommodate this.

Like DNS, the global name service is distributed and hierarchical. In this way large scale growth could be accommodated, while accommodating failures in individual components. The client sees the system as a tree of directories much like a file system. Each directory is identified uniquely by a directory identifier (DI) and a name which can be reached from its parent. For instance the name ANSI/DEC/SRC names the root as ANSI and the directory DEC/SRC relative to it. There is support for links between directories, such that if for instance the directory ANSI/DEC/SRC was linked to ANSI/DEC/SOURCE then both directories are considered equivalent.

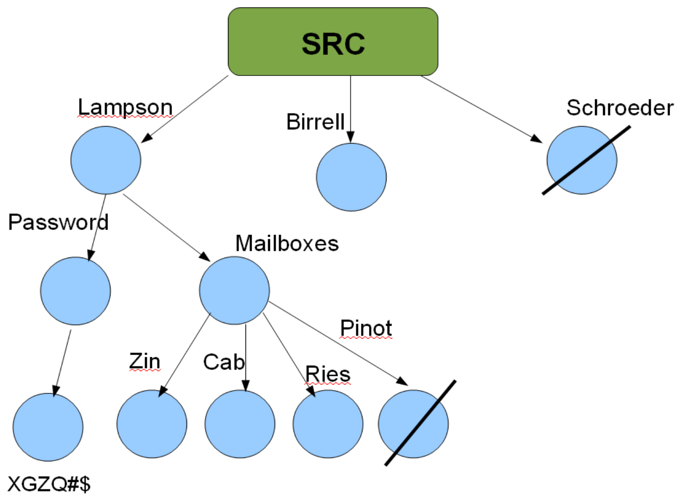

Fig. 2.3: An example of a namespace hierarchy.

Names are mapped to values, much like domain names are resolved to result sets in DNS. Values in the global name service are themselves a tree. The value returned may be a single string as is the case for "Lampson/Password" in Fig. 2.3, a set of strings such as Zin, Cab, Ries, Pinot as is the case for "Lampson/Mailboxes", or a subtree as would be returned for "Lampson".

The name service supports access control. Each update to the namespace is tagged with a timestamp. If the timestamp in the update is a larger value than that of the value it is updating then the existing one is overwritten. Each value has a master copy, which may be replicated on slave nodes. By pushing updates to the master copy only, and from there to the slaves, the directory can be updated from a number of different places without requiring synchronisation. The master performs periodic sweeps which update all the slave servers.

Access control is provided as a function of each directory. Each principal, i.e. each entity that interacts with the directories, proves its identity to the directory by proving knowledge of an encryption key. Principals are identified by a name, and the corresponding encryption key is stored in the directory in a specified value assigned to that name. Each directory stores a mapping between an entity's name, its identification key, and a set of access rights for a given path in the directory. Every time a principal wants to access the directory, the access rights are consulted, and proof of identity may be required.

To the client, the system should appear to be a single name service, which is shielded from the implementation details. Each directory however may be administered by a different entity. Again the hierarchical nature of the namespace makes this possible by delegating control from the top down. Each administrative entity on the other hand sees their individual directories as separate components to be administered. A directory may also exist as a set of copies, for redundancy purposes. Updates to a replicated directory are performed by updating first one of the directories using the timestamp method outlined above. At a designated time interval a sweep of the replicated copies is performed and the various updates are resolved to the most current version, as designated by the most up to date timestamp.

Just like DNS, caching may be deployed to reduce the need for queries to the root. Servers cache their own location with respect to the root. So if for example the ANSI/DEC/SRC server is queried for ANSI/DEC/SRC/Lampson it knows where it is in the hierarchy and may answer the query directly. Each record within the name service is assigned an expiration time and a result is considered valid until that time. This means that cached records may be considered valid until such time as they expire.

2.1.3 X.500

X.500 is a set of standards jointly defined by the ISO and the CCITT, now the ITU (International Telecommunication Union), that aims to provide a global directory capable of storing information about a large number of objects, while allowing responsibility for administration of the directory to be shared among a large number of administrative organisations [44]. This is necessary since each organisation participating in the global directory would be responsible for maintaining and updating the information in their part of the directory and should do so whenever the information changes. The type of information stored in the directory would not in theory be restricted, but the aim for the directory was that it should be consulted by people and applications when they needed to obtain information about a person, application, or entity. The directory should serve a similar function to the type of printed directories that are distributed by telephone companies around the world to their subscribers. However, being electronic and not printed, the information contained within it would be more up to date.

The user will typically access the directory using software known as a Directory User Agent according to the Directory Access Protocol (DAP). These are computer processes that provide an interface to the user and handle requests to the directory on their behalf. Individual entries are stored by nodes known as Directory System Agents (DSAs). The directory itself is comprised entirely of DSAs. These may be located anywhere and are interconnected to allow the information in the database to be accessed by any user. Each DSA stores a reference to a small number of other DSAs. Queries for information in the directory is propagated through the DSAs. If a DSA cannot fulfill a query they will forward it on to one of the other DSAs it knows about.

The Directory Information Base is the set of all information contained within the directory. Typically this will not be viewable nor of interest to all users of the directory. Entries within the DIB are stored hierarchically in a tree structure known as the Directory Information Tree (DIT). This hierarchical model allows for an intuitive distribution of both the administration of the namespace and the storage of the individual entries. From an administrative point of view, control over the maintenance directory is delegated from the top down to various Directory Management Domains (DMDs), which are individual administrative entities. Such an entity will maintain at least one subtree of the DIT, known as an autonomous administrative area. The root of this subtree is known as an administrative point and will contain a type of directory entry called an administrative entry that denotes its existence.

The directory is comprised of entries, which consist of a set of attribute value pairs.

Some attributes refer to information stored within the directory, while others pertain to the operation of the directory itself. An example of the latter are access control attributes. These are referred to as directory operation attributes.

Entries are identified by a unique identifier, the Distinguished Name (DN), which is allocated in a hierarchical fashion and reflects the path of the entry in the DIT. DNs consist of a sequence of Relative Distinguished Names, which are names that share the same root node in the hierarchy. As each RDN is unique within its subtree, so too is each DN. Distinguished Names may be aliases for other entries. Aliases function as pointers to entries and allow the same entries to be obtained under a number of different DNs.

An entry's attributes are of a certain type and a number of standard types are defined, such as "telephone number". In addition there is a type system that allows for new objects to be defined and existing object types to be subclassified. Objects may also be compared according to a set of matching rules. These rules allow attributes to be compared with criteria provided by the user and are necessary for facilitating queries and propagating requests between DSAs.

Routing of requests occurs according to the DIT. When a user queries a DSA the request is recursively sent up the hierarchy until it reaches a level where it is either fulfilled or it will descend the hierarchy until it reaches the required entry. One of the goals of the directory is to allow distribution of its contents, not just among administrative entities but also among the DSAs that store the information. A single administrative entity may have many DSAs each storing a part of its portion of the directory. The subtree that this entity is responsible for may be partitioned among different DSAs and each partition replicated any number of times.

Since the information in the directory is distributed, security becomes an issue. The X.509 portion of the standard describe how distinguished names should be certified to public keys and how public key certificates are incorporated into the directory infrastructure [74]. We will discuss this further in a later section when we discuss Public

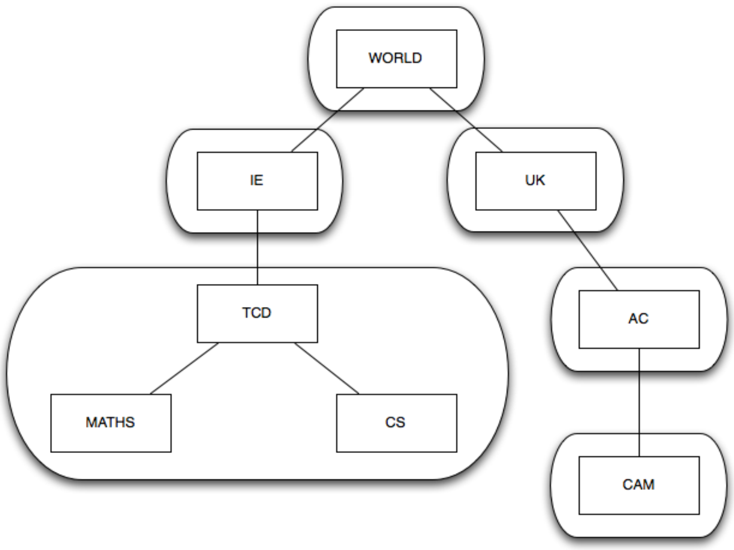

Fig. 2.4: The X.500 system.

Key Infrastructures (PKIs). However, we will just note that this functionality allows for authentication of users of the directory to the DSAs and users to verify the identity of a person they are communicating with as well as authentication of the results of a query.

Access control within the directory is provided using Access Control Lists; that is there is a list of users that are allowed access to an entry which indicates the type of access they have to that entry. By default access to entries are denied to users; that is they need to be granted explicit permission to access entries. Access rules are given a numerical precedence value, where precedence is used to resolve conflicts between rules. Conflicts can also be resolved by their specificity. That is if for example a rule exists that a user, Alice, can't read any attributes, but another rule exists that says she can read telephone number attributes, then the second rule will overrule the first since it is more specific.

These days the Lightweight Directory Access Protocol (LDAP) [124] is more often used as a directory system than X.500. One reason for this is due to LDAP not requiring the deployment of an OSI stack, while providing the same functionality as the X.500 directory services.

2.1.4 E.164

The E.164 recommendation from the International Telecommunications Union specifies the numbering plan for the telephone system [73]. The plan details which country codes are to be assigned to which country. Regulatory bodies in these countries then decide their own numbering plan for their country according to the constraints set down by the E.164 standard.

The recommendation specifies the maximum length for a telephone number at 15 digits, and the format of the number, namely *<country code><national code/area code><destination code>*. A national code and a destination code together form a nationally significant number. The country code is needed to contact international numbers. By far the bulk of the numbers used are geographic numbers, that is numbers that are tied to a geographic location by their country or area code. Often the area code is used to allocate a further set of numbers to an individual service provider. This creates a second level of the naming hierarchy. Traditionally this meant that since your phone number contained a component that was service provider specific, if you moved to another provider you had to change your telephone number. Now, however, there are a few specifications for number portability systems and a number of countries have since mandated its use and adopted one of these standards.

Number portability is normally achieved through the use of a centralised database. This database may be stored centrally by the regulatory body or a list of numbers that have been ported may be stored by the original assignee. This database may be queried either at the beginning of a call or after a service provider has, through the signalling protocols, indicated that the number is no longer serviced by them. There are four different types of service provider number portability that operate using some combinations of these actions. A review of these methods is provided in [67]. However, each of these solutions still constrains number portability to service providers within a certain geographic area. The ITU has proposed Universal Personal Telecommunications

[115] as a means of providing global number portability.

ENUM

ENUM is a system for resolving telephone numbers to URLs using the Domain Name System. The E.164 hierarchy is moved into the DNS hierarchy. Phone numbers are translated into domain names, resolved in the usual way, and a set of Resource Records is returned as a result of the resolution function. This allows for routing of calls between the regular telephone network and Internet based systems. This allows the use of telephone numbers as identifiers in Internet telephony systems, for example, thereby linking the telephone numbering system with the Internet naming system.

The resolution process is illustrated in the figure below. A new zone, e164.arpa, was allocated for use by ENUM. Resolution is performed as follows [37]:

1. Take an E.164 number in full international format, for example +35312345678 and remove the '+'.
2. Insert full stops between each of the numbers giving 3.5.3.1.2.3.4.5.6.7.8
3. Reverse the order of the resulting string leaving 8.7.6.5.4.3.2.1.3.5.3
4. Append the top level domain ".e164.arpa" to obtain 8.7.6.5.4.3.2.1.3.5.3.e164.arpa
5. Perform a regular DNS lookup treating each of the numbers as its own domain.

The result of the query is a DNS record, typically of type NAPTR, which allows for a URL to be returned. This allows E.164 identifiers to be resolved to obtain identifiers for Internet based systems. Thus, ENUM can form part of a telephony-convergence architecture.

In the example above the resolution takes at least 11 queries, one for each of the digits in the phone number. This number of queries would make resolution very inefficient. However, due to the way that E.164 numbers are allocated, there should only be a very small number of administrative entities involved, which can be used to reduce the number of queries necessary. The individual name servers will return the result for the longest query they can answer. In this case the root node could return the authoritative name server responsible for answering queries for 353, the Irish country code.

The NAPTR record type also supports the use of regular expressions on the result. So, for example, a regular expression could specify that a part of the number being queried should end up in the result; e.g. if a SIP provider allocates identifiers based on E.164 numbers, then the destination code or a part of it could be tagged onto a SIP identifier and returned by regular expression. The regular expression would be evaluated on the client side.

There are a number of ways ENUM can be deployed. One way, Public ENUM, describes an infrastructure much like the way DNS functions, where each person controls the results for their own identifier (phone number). This would allow for E.164 numbers to be used as global identifiers while a person could specify whatever they wanted in the result set. People contacting them could first query the ENUM database to obtain the most up to date identifier for that person. E.164 numbers have the advantage of being used widely as identifiers already and people are used to dealing with them and remembering them. Under this means of deployment ENUM could form the basis for a global number portability system. So long as a person can control the mapping between their phone number and its result set then they can use their telephone number, for as long as it has been allocated to them, as a unique identifier that can be resolved to obtain their real phone number or other identifiers. An example of this type of deployment is E164.org.

Another possibility is that individual providers use ENUM within their own networks, and potentially as a means of facilitating peering with other providers [84]. In this scenario the provider would maintain control over updating results for each identifier. The primary use of ENUM in this case would be call routing and the individual carriers could also use ENUM as the basis for routing with other networks that they peer with. So far public ENUM hasn't seen major deployment, but the area of Internet telephony is still in its infancy and it may yet be successful.

2.1.5 Discussion

The question remains whether these name systems, or elements of them, would be suitable for solving the naming problem we wish to solve. The main point of commonality with each of these naming systems is their use of the hierarchical structure of the namespace as a means of partitioning storage and administration responsibilities. This provides an attractive way to scale up the namespace to a large number of users. The choice of namespace structure was influenced by this requirement. However these namespaces don't address the issue of providing a scalable namespace for people that retains high mnemonic value very well. The set of names that are memorable is not that large, and systems such as DNS have experienced a land rush for the most popular names when their value was realised. This problem is hard to address in a hierarchical system as although the number of names possible in a hierarchical system is theoretically limitless, people's capacity to remember them is not. With over 6 billion people worldwide we fail to see a satisfactory way to address this problem while adhering to the design of a hierarchical namespace. Telephone numbers have addressed this issue to some extent as people have become used to memorising the 6-8 digit combinations. However, phone numbers are only easy to remember when localised and when the number of digits to remember is low. Many people who use mobile phones for communication arguably do not remember these numbers, but rather assign names locally within their address book and select people to call based on a name they can remember.

All of these name systems use distributed databases to store mappings between names and their results. Their replicated nature provides a useful level of robustness for infrastructure that is critical to the proper functioning of a distributed communications system. This is an attractive property for a naming system, and new naming systems could take advantage of the widely deployed infrastructure of DNS to achieve the same properties. Generally public DNS servers provide read access to all results for domains they host, and there is no way to authenticate the origin of a query. Since one of our goals is to protect the privacy of results, and so only have them shared with those that the user wishes, we do not see an easy way to add a layer of access control based on the identity of a user of DNS without major modifications to the protocol. For this reason we do not use DNS infrastructure as the basis for our resolution mechanism. However, we believe that DNS has value for the dissemination of data, and when DNSSEC sees wide deployment it will be all the more useful. Neither the Global Name Service nor the X.500 directory have provided usable global infrastructure. This is despite being arguably more suitable to the task of acting as generic global directories, for instance due to their support for access control.

Though we find the hierarchical model of name allocation and resolution unsuitable for use in a personal naming system, each of these systems have contributed ideas towards our design. We think it is important for replication to be used as a means of providing robustness in the face of failure. The simplicity of the replication mechanisms in DNS, the Global Name Service, and X.500 appeal to us. We think this is a strength of each of the systems. The idea that each entity should be able to use a name server of their choosing is also an important point. In DNS this allows large organisations that have more heavily loaded name servers to implement appropriate scaling mechanisms, and ensures that the mapping between name and results is fully controlled by the owner of that name. The hierarchical model is an important part of the state of the art of Internet naming systems, but there are other solutions that have their own advantages, some of which seek to correct issues with systems such as DNS. We shall now look at some of these.

2.2 Name Resolution Systems

DNS came about as a replacement for the existing method of a flat file of names, which became impractical to administer once it grew beyond a reasonable size. The old method relied on individual network administrators to find ways of transferring the entire file of name mappings from one machine to each machine within their network. This is obviously not a scalable means of resolution. Today as the .com zone file is of the order of gigabytes in size it is even less so, and given that most people will only need to access a relatively small number of results from this zone it would also be grossly inefficient. The hierarchical model of resolution is not without its problems however. In this section we will look at some systems that perform resolution in a different way.

2.2.1 Main Name System

The Main Name System [53] is a proposal to fix some of the perceived shortcomings, according to the authors, of the DNS infrastructure, namely latency of lookups and updates, complexity of administration, vulnerability to denial of service, and the lack of authentication of responses. The proposal is to "recentralise" the naming system, with a small number of high performance servers serving data for the entire naming system, like a content distribution network. The existing top-level DNS servers could be used to host the new system, avoiding the need to modify existing client software. They argue that with fewer than 100 servers each strategically placed around the globe they could serve DNS data at a much lower latency and at a higher scale to the entire Internet.

Since all requests would be fulfilled in a single hop, this would reduce the latency of queries significantly, while update latency could be reduced by eliminating the TTL caching mechanism i.e. by returning a TTL of 0 for all records. They argue that since people wouldn't have to run their own DNS servers, the system would be a lot simpler to maintain and a more user-friendly interface could be presented to users of the system. The proposal does alter the IP-based model of authentication that DNS uses however, but recognises that other proposals such as those in the DNSSEC protocols could address those issues, since DNSSEC allows the separation of the authentication of records from the manner in which they are stored. While centralisation of the DNS architecture may be possible and a good idea, since DNS makes more than enough money to be self-sufficient, other naming systems may benefit from a more centralised approach too.

The authors of the Main Name System proposal argue for the benefits of a centralised approach. When failure occurs at a single DNS server during resolution of a query this impacts the system as a whole. They explain that these failures are often due to misconfiguration of the servers, which is a problem that could be solved by centralising the infrastructure.

However while the centralisation of the DNS infrastructure would be sustainable due to the fact that the DNS system as a whole is financially solvent, this may not be the case for other types of naming system. The centralisation of this important piece of Internet infrastructure would make the maintainer of that infrastructure a target for attackers, would put too much power in the hands of one entity, and would create privacy concerns. We will now look at some systems that advocate a more decentralised approach based on overlay networks.

2.2.2 Overlay Networks

An Overlay network is a virtual network that is layered on top of an existing network infrastructure. Application-layer overlay networks have, for example, been used in resource location and discovery to provide a decentralised resolution infrastructure. Some of these have the goal of being able to function without the need for a centralised entity, with the advantage of being more resilient to attack than a centralised infrastructure. In this section we will outline the operation of a number of such overlay networks. Of the systems we will look at, there are two main categories, namely flooding networks and distributed hash tables (DHTs). Flooding networks are simpler in design, but do not provide any guarantees about the number of hops required for resolution and may require a large number of messages to be exchanged in order to respond to a query. On the other hand, DHTs are generally more complex, but provide significantly better performance guarantees i.e. a bounded number of hops to resolve a query.

2.2.3 Gnutella

Gnutella is a flooding style network. Its protocol and operation is described in [16]. Each node is connected to a relatively small number of neighbours or peers. To join the network, a joining node must know the IP address of a bootstrap node, one that is already in the network. The joining node will query the bootstrap node for a list of other nodes in the network, randomly select a number of them, and connect to them as neighbours. When a node wishes to query for a resource it will query each of its neighbours, who will forward the query their neighbours, and so on. Since this form of query is exponential, potentially a large percentage of the nodes in the network may be queried. A node that has the requested resource can reply to the origin of the query directly. To avoid sending too many messages, each query will include a counter called the time-to-live (TTL) value. As a node forwards a query they decrement this counter, discarding any requests that reach zero.

In later versions of Gnutella, to achieve better scalability, the concept of ultrapeers (or supernodes) was introduced [108]. A regular node could promote itself to the status of supernode if it met certain requirements, such as adequate CPU speed, high-speed connectivity, etc. Each node would maintain a connection to a number of supernodes and would route queries through the supernodes. The supernodes would then route queries to other supernodes and regular nodes they peered with. This two level hierarchy reduced the number of messages that needed to be exchanged, but placed a larger burden on the supernodes.

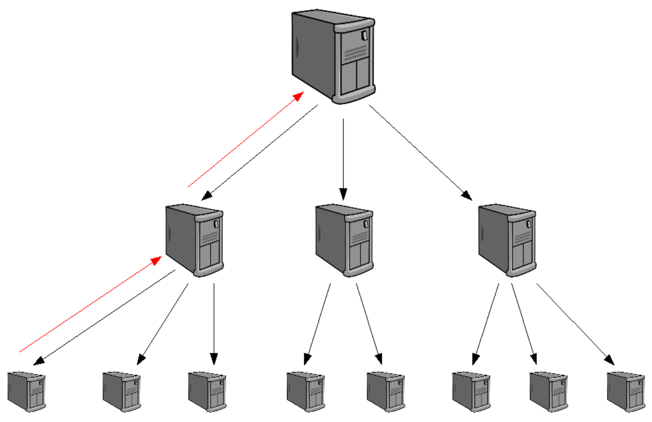

Fig. 2.5: Illustration of query propagation in Gnutella. The black arrows depict the query path, with the red arrows showing the path of the matched query.

2.2.4 Skype

Skype uses a Gnutella style network with supernodes to store information about its users. Although the precise operation of the Skype network is unavailable, due to its proprietary nature, a number of studies have been undertaken, such as [28], [71], and [35], that give an insight into the operation of the network and its protocols. In addition to its decentralised Gnutella network, the Skype software uses a centralised login server. This server ensures that usernames are unique and authenticates peers on login. On successful authentication of a username and password pair, the login server issues the client with digital certificates that allow them to prove their identity to other peers with a public key. These certificates are stored in the Gnutella-style network by the individual Skype clients and may be returned as the result of queries.

Skype also uses peers in the network to provide NAT traversal for nodes in the network that need it, presumably using some variant of the STUN (Session Traversal Utilities for NAT) protocol [105]. Skype is also one of the first Internet Telephony systems to include encryption by default. An analysis of the implementation of the cryptography protocols was undertaken by Tom Berson of Anagram Laboratories in

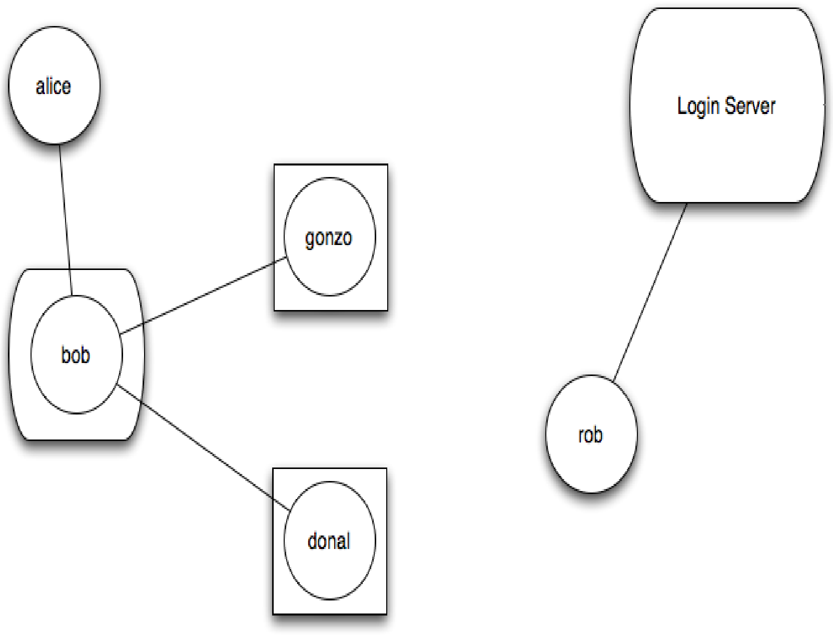

Fig. 2.6: The Skype system. Boxed nodes like Donal and Gonzo are NAT'ed nodes and rely on Supernodes such as Bob to route traffic on their behalf. Authentication happens by contacting the Login Server directly

2005 on a snapshot of the Skype code. He concluded that the implementation of the various protocols were correct and that the security architecture of Skype was well designed [30].

2.2.5 Chord

Chord is a distributed hash table that was developed at MIT [112]. The DHT has been subsequently used in the design of a number of distributed systems, such as the DHash block storage system and the Cooperative File System (CFS). The basic structure of the DHT is a ring, where each node in the ring maintains a connection with its successor. Each node in the ring is assigned a number, which determines its location within the ring. A node's number in the ring is assigned based on the hash of some key. Consistent hashing is used to assign keys to nodes. Typically a secure hash function, such as SHA-1 [96], would be used as the base hash function for this purpose.

The maximum number of nodes allowed in a given hash table is some value $n = 2^m$.

In order to achieve $O(\log n)$, or m steps, worst case lookup a finger table is maintained. A node's finger table consists of a list of nodes that are distances of successive powers of 2 from that node. For example, say we have a node n, then the finger table will contain information about nodes that are at a distance $n+2^{k-2} \bmod 2^m$ for $1 \leq k \leq m-1$ away from the node in the ring. Traversing the network to find a given key thus involves locating the nearest node to that key in the ring. The first node chooses either a node from its finger table or its successor depending on which node is closest to the key and forwards the query there. Each node the query is forwarded to recursively forwards queries in this fashion until the destination node is reached.

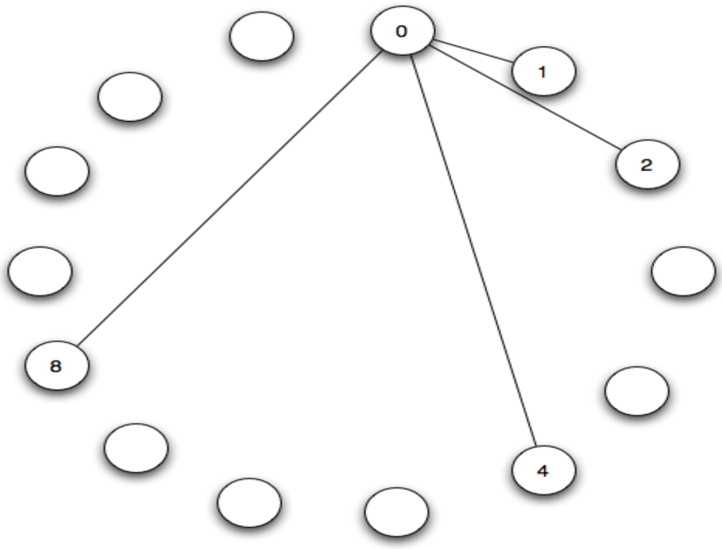

Fig. 2.7: The chord topology.

This method assumes that the hash table is full, as each node in the network requires a successor and a full finger table in order to achieve the $O(\log n)$ lookup time. In the case where the table isn't full, which is generally assumed to be the case, nodes will take over the function of those missing in the following manner. When the node joins the network it will attempt to locate its successor. If that fails then it will locate the nearest node to its successor. The joining node will then take over the duty for all nodes between it and its successor. When other nodes attempt to query a missing node, that node will respond on its behalf, and thus queries get routed either to their destination node further down the chain or that node responds that the query doesn't exist.

2.2.6 Manifold

Manifold [60] is a decentralised resource location and discovery system, that is designed for use in a dynamic ad hoc environment, where there is no fixed infrastructure. Manifold defines two overlay networks, one to perform exact searching, while the other supports inexact queries. Exact search is performed by a DHT known as Manifold-g, while inexact search is performed by the Gnutella style network Manifold-b.

The systems supports exact search on a global scale, where people will typically want to obtain specific resources, such as a certain document, or the identifiers. The topology of the network is based on an N-dimensional hypercube. A hypercube is a general abstract shape of which, for example, a square and a cube are instantiations in two and three dimensions respectively. Taking the example of a cube, we can easily verify empirically that each of the corners (nodes) is at most three nodes away from any other node in the cube. This property holds in N dimensions. Also a hypercube may be labelled algorithmically in a way that defines a Gray code. So transitioning from one node in the cube to another requires the flipping of a single bit. In this way routing can be performed using an XOR and a bit rotation at each node. The search performed is done on the exact search query. This is due to the way that resources are named in the DHT. Each of the nodes in the network use a pre-decided hash function to decide the addresses of nodes in the overlay. The name of a resource is then named according to the hash of the query string that is used to identify it. So for example the Trinity College homepage would be addressed in the networks as *H(http://www.tcd.ie)*, where *H* is the hash function used in the network. Since a slight variation on the address would give a different hash value it is not possible to perform inexact search using Manifold-g. Like Chord the Manifold-g routing algorithm, as outlined above, assumes that the hash table is complete. In order to facilitate routing in an incomplete network the idea of shadow nodes is introduced. These nodes take over the routing function for missing nodes.

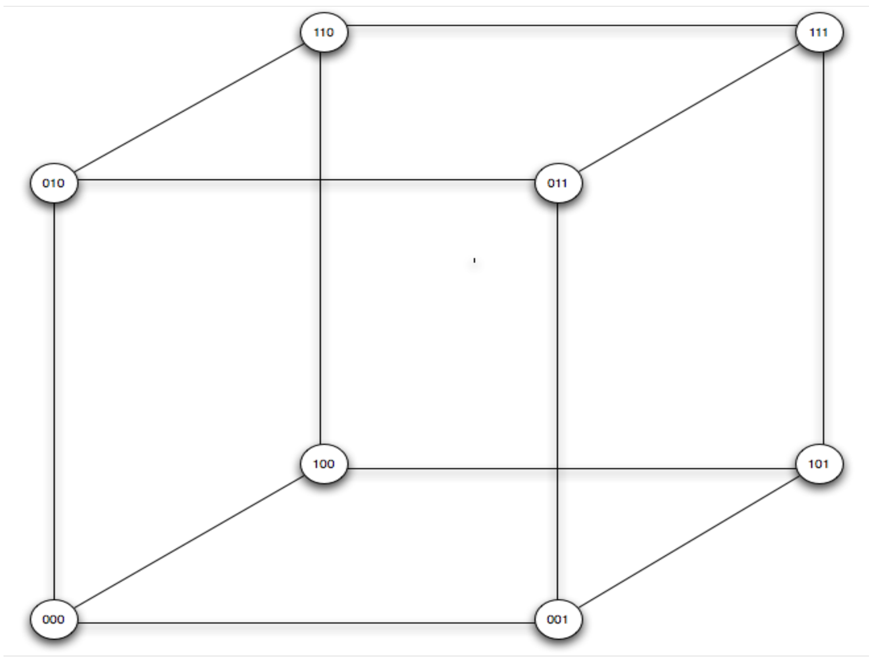

Fig. 2.8: Manifold queries follow a path along a hypercube. This figure depicts a query showing a complete hypercube.

For this reason a TTL-controlled flooding network based on the Gnutella overlay, known as Manifold-b, was defined. This is used for queries that may for example be based on the properties of the resource being sought, e.g. "any colour printer". This also allows partial matches of objects to be returned. For example, the query "colour printer" might return black and white printers if no colour ones are available. The node that performed the query could then decide, upon receiving no results for colour printers, that the black and white printer will suffice. Clearly if inexact search can be performed by Manifold-b, so could exact search. However, Manifold-g exchanges far fewer messages during resolution and so the two overlays are used in conjunction with each other to achieve the best of both worlds.

2.2.7 DNS over P2P

Cox et al. [50] investigate the feasibility of using a distributed hash table to serve DNS data using a system they call DDNS (Distributed DNS), not to be confused with Dynamic DNS that also uses this abbreviation. They argue that the DNS security extensions proposed by the DNSSEC effort separate the authentication of the result set from the storage and serving of the data. They propose using a decentralised solution for a number of reasons. Firstly, in [52] the authors conclude that the reason for a large number of DNS failures is due to the complexity of maintaining servers. Their proposal would reduce this complexity as instead of each organisation or individual having to maintain their own server or procure one from an ISP or elsewhere they would simply run the DDNS software, which would contribute resources to the global DHT with presumably more or less similar configuration across each of the hosts in the network. Secondly, a distributed solution should more evenly distribute traffic among the participating nodes rather than the current situation where it has been shown that up to 18% of traffic goes to the root nodes [76]. The decentralised nature of the infrastructure would also make it less susceptible to denial of service attacks, since data could be replicated across a large number of nodes thereby increasing the number of nodes that would need to be attacked. While DHTs provide a low number of hops in comparison to the number of nodes that the network can scale to, usually $\log n$, the number of hops is still usually too large for latency sensitive applications. Cox et al. [50] conclude, therefore, that serving DNS over a DHT would be bad for latency, and since name lookups should have low latency this would make such a system impractical without some kind of replication mechanism.

2.2.8 Beehive and CoDoNs

Beehive [101] is a proactive replication mechanism for distributed hash tables that seeks to reduce the latency for name lookups by reducing the number of hops needed for resolution in the average case for DHTs based on prefix routing, such as Chord [112], Pastry [106], Kademlia [89], and the Manifold-g component of the Manifold overlay [60]. In these systems nodes that provide shared storage are addressed according to a fixed length bit string. Data to be inserted is also given an address, which may for example be determined by the hash of its name or contents. This address is then stored at the node whose address shares the longest prefix in common with the address of the data. When a lookup is to be performed, the current address is changed bit by bit from left to right at each hop, with the query being forwarded all the while, to get the next hop until the node that stores the data is reached. This style of routing makes use of the fact that each additional bit doubles the possible number of addresses in the network, while increasing the number of hops by one.

The replication algorithm makes use of the fact that if a result is replicated by all possible previous hops then the number of hops to get to that data is reduced by one on average. Clearly replicating all the data would be inefficient if the data set is large since that would mean all nodes in the network would need a large proportion of storage. However, if the popularity of files in the data set follows a power law distribution then the most popular data may be replicated with a relatively small overhead to reduce the number of hops for a large amount of the queries. Beehive allows the degree of replication to be configured but seeks to provide a one hop lookup on average. The system figures out which data is the most popular at any given time and proactively replicates it across the network. This allows it to handle flash crowds, where large numbers of requests are made for a certain resource in a relatively small time frame, and adapt to distribute the load caused by such events across the network.

CoDoNs [102], the Cooperative Domain Name System, is a system that resolves DNS queries using Beehive and the Pastry overlay. The system has been proposed as an alternative resolution mechanism to the current DNS system. The authors claim that it would be more resilient to attack, quicker to resolve queries on average, would update changes more quickly, and would distribute the load more fairly across the different storage nodes. Additionally the overhead of maintaining a server would be reduced as just a CoDoNs node would need to be maintained, which would eliminate the need to, for example, deploy a DNS cache.

2.2.9 Freenet

Freenet is a peer to peer distributed storage system developed originally by Ian Clarke and outlined in [47]. The motivation for Freenet was to develop an anonymising network which is resistant to censorship. While the main function of Freenet is to store files, a working email system has been developed on it also. Currently there are two modes of operation of Freenet, one called opennet and the other darknet. Opennet is the global Freenet network and allows anyone to access the data stored by it. Darknet on the other hand requires that people be manually added to the network and is designed for small-scale networks, where the participants generally belong to the same social network. Anonymity in the network is maintained by putting chains of nodes between the source and destination endpoints, while each node in the chain knows only the IP address and Freenet identifier of the node prior to and after it in the chain.

Files are stored in the network according to a key, which is based on the hash of either a persistent identifier for that file or the file contents itself. Routing in the network initially takes place as a random walk. Each node upon joining will obtain a list of possible neighbours, which will be a subset of the total nodes in the network, and will initiate a connection with a small number of them. When that node then wants to query a key it will select a random neighbour and forward the query to them. If the query comes back negative it will forward the query to a different node. Nodes can detect recent duplicate queries and all queries contain a time to live, which when it expires causes the query to be returned false if it hasn't already matched a result. Queries are recursively forwarded through the network in this fashion. Results that come back along a certain path are replicated at each node in that path in order to increase the availability of that file, with popular queries becoming highly replicated and thus more likely to be found quickly. Additionally nodes remember which neighbour responded positively to previous queries and forwards similar queries down that path. Thus, clusters of files which have similar keys should cluster together in the network.

2.2.10 Discussion

The centralised approach of the Main Name system offers clear performance benefits over the large numbers of hops typical in a DHT. If the infrastructure is centralised then load balancing can be performed on a more deterministic and fine-grained level. The decentralised nature of P2P means that the resources on the network are heterogeneous and unknown. The overlay network itself must take into consideration that some nodes may be slower than others, and existing approaches, such as the supernodes in Gnutella, rely on the nodes themselves to determine their suitability for certain operations within the network. This is undesirable as it increases the burden on certain nodes and may have real implications for their ability to use resources.

We also find the security of the P2P approach to be lacking. In [23] we performed a threat analysis of overlay networks, including the ability for nodes within the network to subvert the routing process and intercept large numbers of queries. These issues are not addressed in any existing systems we are aware of, though there are attempts to solve some of these problems in the literature. The ad hoc nature of these networks would make robust solutions to these problems difficult to engineer. While these systems are suitable for certain applications where these security issues are deemed to be an acceptable trade off for the benefits of a decentralised name lookup infrastructure, we do not feel that they are suitable for our solution.

2.3 Name Registration Systems

In the previous section we took a look at alternative name lookup systems. This section will outline another aspect of naming systems by detailing different types of name registration mechanisms. Each of these differ from the hierarchical approach taken by the systems outlined in the first section.

2.3.1 Host Identity Protocol

The Host Identity Protocol (HIP) [95] seeks to provide a means to identify hosts as they move between different physical points of attachment to the network, and therefore different IP addresses. To achieve this HIP introduces a new layer between the transport and network layer with its own namespace based on cryptographic identifiers. The transport layer is decoupled from the network layer and uses the host identifier, known as the Host Identity Tag (HIT), instead of an IP address. The HIT is stored in the host's DNS entry and is obtained on resolution. Since the HIT is the hash of a public key, end points may be authenticated and HIP provides this functionality.

When HIP is deployed a client application will resolve the domain name of the target to obtain its HIT. The client application will then attempt to send information to the HIT. The HIP layer may use a rendezvous service, i.e. a third party that serves as an initial point of contact for clients [75], to obtain the IP address associated with that HIT. They will then perform the authentication function and continue. At the end of the authenticated key establishment protocol each client will know the other's HIT and a session key will have been established. When one of the hosts changes its IP address, the rendezvous service will facilitate the transition from one IP to the other with each endpoint's HIP layer ensuring this happens transparently at the application layer.

Apart from mobility, multihoming is another service offered by the HIP. This is possible since the HIP layer is responsible for determining the destination of packets. The resolution of a HIT can also return a set of IP addresses. The HIP layer will then send data to each of the hosts in the result set.

2.3.2 Semantic Free Referencing

Semantic Free Referencing (SFR) as outlined by Balakrishnan et al. [119] seeks to propose an alternative way for referring to and locating objects on the web. Citing DNS issues that have impacted the web such as legal challenges to domain name ownership, lack of faith in the governance of the namespace, and the lack of persistent identifiers for objects, they propose a naming solution based on names with low mnemonic value. They propose removing all semantic information about the object from the name in order to decouple information such as location in the network from its identifier. Furthermore a resolution architecture is proposed.

The proposal is to base the namespace on flat identifiers, known as SFRTags, which resolve to o-records, which contain the location and metadata for an object. The resolution architecture is based on a DHT, in this case Chord [112], and a storage system that layers on top of it known as DHash [51]. The authors propose that the resolution infrastructure could either be maintained by a not for profit organisation such as a governmental body, or be a commercial offering where different providers compete with each other, with updates being propagated between the different providers.

The name service has the property that each record is self-certifying. Since the SFRTag is a hash of the public key of the provider of the object, the record itself may be signed with the corresponding private key. The public key is then distributed as part of the o-record. In this way, when a person resolves a given SFRTag they are assured that only the person who knows the private key can update its location.

Two issues that the authors have identified with a DHT based infrastructure, rather than a DNS style infrastructure are those of fate-sharing and latency. Fate-sharing deals with the question of who is affected by outages. For instance in DNS when an organisation's Internet connection goes down then it is desirable that local machines are still accessible. In a DHT, if a node fails and it's responsible for serving information about other nodes, then this is not possible. To remove this issue SFR specifies that a separate cache for records specific to an organisation should be put in place. Latency on the other hand is an issue associated with the relatively large number of hops to perform resolution using a DHT. To reduce this latency the authors propose two mechanisms. First, records are cached in the DHT along the query path. This should result in popular records being distributed throughout the DHT, thus reducing the number of hops needed to access them. Secondly, caching is proposed on an organisational level. The same cache that handles caching to avoid the fate-sharing issues would cache popular records. Records are given an associated time to live value. Once this is expired the record is purged from the cache. In this model organisations provide a SFR server for the organisation, similar to the way they run a DNS server today.

2.3.3 Layered Naming Architecture

The Layered Naming Architecture [27] is a proposal to provide three features to the Internet infrastructure, namely the ability to persistently name objects, support for mobility and multihoming, and the problems that middleboxes, devices such as NAT boxes that perform routing or policy enforcement at the transport layer thereby breaking the end to end principle, cause in the current architecture. Seeing the core infrastructure as untouchable, citing the fact that IPv6 still has not been successfully deployed after more than a decade as evidence of this, the authors of the Layered Naming Architecture proposal have instead elected to tackle these issues by altering the naming infrastructure. They propose replacing the current two namespace model, where domain names are mapped directly to IP addresses, with a model that has four types of namespace. Their work builds on that undertaken by HIP and SFR, which we discuss in sections 2.3.1 and 2.3.2 respectively, and the Internet Indirection Infrastructure, which is outlined in [113].

The first type of name is called the user-level descriptor (ULD). This corresponds to any type of application specific identifier, such as email addresses, SIP URIs, search terms, etc that a user currently encounters while using Internet services. These identifiers may then be resolved to a service identifier (SID) using any method deemed appropriate. The SID remains persistent for each object, so for example a specific file may have its own SID and that SID will always act as a locator for that specific file. This addresses the first goal for the architecture.

In a typical use case scenario the SID corresponds to an application layer identifier, such as an URI. Applications will still need to resolve this to a network layer identifier in order to be able to route to the endpoint that the resource is located at. When a SID is resolved, another new identifier the endpoint identifier (EID) is returned. This identifier is used by the transport layer as a handle that it can send packets to and as a level of abstraction for the network layer so that the address of the endpoint may change. The EID must thus resolve to at least one network layer identifier. If in the middle of a session the destination host changes its network layer address then the EID may be resolved to obtain the new identifier.

Resolution of the EID may result in multiple IP addresses being returned. In this way multihoming may be achieved. The EIP abstraction also allows hosts to change IP address in the middle of a communications session. The IP layer can detect the change through timeouts or ICMP error messages and resolve the EIP again to obtain the host's new address. To handle middleboxes the authors propose allowing the EIP to point to a source route. Since the issue with middleboxes is that the IP addresses behind the middlebox are usually private IP addresses, they cannot be addressed globally. A source route would solve this issue, as the path through the middlebox to the destination would be specified.

Since all of the identifiers used in the Layered Naming Architecture are based on a flat namespace, the obvious choice for resolution of these names is a Distributed Hash Table, which we will discuss further in a later section, and that is the approach that the authors have opted for. The authors don't address the latency issue with their choice of DHT, but suggest that the latency issues can be addressed in other ways, for example by aggressive caching.

2.3.4 INS

The Intentional Naming System [19] is a resource location and discovery system, where resources are named by their attributes. The goal is to allow queries for resources such as "least busy printer in the building". INS integrates the naming of resources with message routing, using a default late-binding mechanism, where the location of the resource isn't specified until time of message delivery. This allows the node to change its network address at any time. The routing system allows for multicast as well as unicast of messages, so that messages may be routed to all nodes that match a given request, e.g. "all printers in the building". The system uses an application layer routing system that pushes its deployment requirements to the edges of the network.

Names in INS are specified as a hierarchal structure of attribute-value pairs. The attribute specifies the category that an object belongs to, while the value specifies its type within that category, e.g. "shape=circle" specifies a shape of type circle. The hierarchy is structured such that child nodes are dependent on their parent nodes. The final node in each branch of the hierarchy points to a name record that satisfies that query. To resolve a name each of the attribute value pairs in the query are located in the hierarchy and the intersection of the name records below them is returned.

Resolution in INS is performed by a number of resolvers, known as Intentional Name Resolvers (INRs) that arrange themselves, within a local network, in a spanning tree overlay topology. The overlay network is tolerant to failures and individual INRs may leave or join the network at any time. Services advertise their names periodically and INRs listen on a well known port for these advertisements. These advertisements also serve to refresh information stored in the INR, which may expire after a given length of time. This allows nodes to recover from failures and alleviates the requirement that they maintain state between reboots. INRs also share information about services between themselves.

The Intentional Name System is of interest because of its query based naming design. This allows for a single object to potentially be located according to a number of different criteria. This approach is interesting from the point of view of naming people since different people have different and often incomplete information about a person when they try to locate them. For instance, one person might know a person's first name, surname, and country of residence, while another may only know their surname and the area they live in.

2.3.5 Semantic Email Addressing

The goal of semantic email addressing [79] is to provide an infrastructure that can route email based on semantic information rather than a particular address. For instance, a person could choose to send an email to a particular person as described by their position and place of work. SEA also allows a "recipient" to be a group of people with a common interest, such as "people who like film noir". People can specify topics they are willing to receive email about and the SEA system will route email that matches those topics to them accordingly.

People opt in to receiving email from the SEA system by specifying their email address, their place of work, job title, interests, etc. in a document using a semantic web standard, such as FOAF (Friend of a Friend) [41]. They make this information public in their FOAF document, which may for example, be linked to from their website. The SEA system then indexes this information and stores it centrally in a database. They may update this document to remove or add information as it becomes obsolete, which will then be updated in the database whenever the system indexes it.

When a person wants to send email using SEA they use a special SEA aware server known as a SEAmail server. This will be hosted by the same entity that has indexed the semantic information. They will then construct a semantic email address. This is a query that matches information about a subset of the people indexed by the system. The queries match on the semantic information contained within the document and the email address specified within is used to deliver the email. The SEA provider will then deliver the email address to the appropriate recipients.

The SEA system provides a novel way to deliver email and because it uses semantic information rather than a particular address to route messages it separates the delivery of a message from the way it is addressed. This could be used to provide identifier portability since a semantic description could be used as a person's persistent identifier, while their actual address may change. However, the SEA system does not address privacy issues. The authors do not provide a way for people to filter based on sender. They argue that this guarantee is not provided by the current system and so SEA provides more flexibility than the legacy system. However they acknowledge that SEA could be abused to send untargeted email by specifying an address that matches a large number of people. Also, since a person's FOAF data is made public it would be possible for a spammer to index their email address without much difficulty.

2.3.6 Discussion

This section saw discussion of three different approaches to naming that each advocated names that lacked mnemonic value. None of these systems rely on a centralised authority to perform name allocation thereby requiring little or no financial cost to be incurred in the running of such an authority. Names are allocated in a way that allows claims of ownership to be proven. For instance, an identifier in the form of a hash of a file name can be used to verify that, when located, the file hashes to the correct value and is therefore the correct one.

Names of this form are not memorable, but they do not always need to be used or seen by people. Many URLs used on the Web today are long and unmemorable, but they are not meant to be remembered. Instead they are only used to locate resources. Other ways of exchanging and storing these URLs may be used such that people don't have to. For instance, most browsers available today store bookmarks, which are essentially memorable names for unwieldy URLs, Google is an efficient method of finding resources when needed, and many social bookmarking sites aim to index interesting content.

The approach taken in the latter two systems, SEA and INS, is based on attributes of the resource or person being named. While it is true that a person's characteristics may identify them uniquely, this approach is not appropriate for a personal naming system. Systems such as SEA and INS require these attributes to be searchable and only function if they are. Thus participation in the naming system requires information about a person to be shared. While some people may want to share this information it may not be an appropriate way to identify all people and doesn't address our requirement of protecting privacy. An example of this is that parents may not, for instance, want information such as their children's email address to be public.

The Layered Naming Architecture explicitly supports and recognises the need for middleboxes. Middleboxes break the end to end principle [107], but are often required to add functionality missing from the existing Internet infrastructure, where neither modifying client software nor core infrastructure is possible. One example of the need for middleboxes is in Network Address Translation where the middlebox is responsible for mapping private network addresses to a global one. There are a number of strategies for doing this, and they are covered in some depth in [110]. Middleboxes have been advocated by personal mobility solutions, and it would seem that they are an unavoidable part of the infrastructure.

2.4 Personal Mobility and Naming

2.4.1 Unmanaged Internet Architecture

The goal of the Unmanaged Internet Architecture [65] (UIA) is to enable zero-configuration connectivity between mobile devices. In UIA each person maintains and runs their own namespace, from which they assign names to their devices and people they know [66]. When people meet they engage in an introduction protocol where they share their namespaces with each other. In this way they can each refer to devices in each other's namespaces, providing an intuitive way to access them. For instance Bob might want to access Alice's iPod. Since Alice probably only has one she might name it *ipod.alice*. As Alice's devices move to different points of the network she will distribute changes in the mapping between their UIA names and their locations using a gossip protocol [54], where changes are sent to all devices Alice's device encounters. During the introduction process Alice will have suggested to Bob that he refer to her by the name *alice*. Bob may choose to accept this name if it is appropriate for him and doesn't conflict with another entry in his namespace.

These names act as a memorable identifier for people like Alice and Bob to refer to one another. In reality each device will identify itself with a public key, and each person maintains an SPKI (as defined in Section 2.6.4) infrastructure for authentication purposes. These namespaces are hierarchical as a natural hierarchy emerges from a the relationships between the devices, their owners, and the people they communicate with.

The UIA also allows a person to form groups of devices. In this case each device will ensure replication of the namespace across each of the other members in the group. This is again achieved using a gossip protocol, where devices share updates with each other opportunistically. There is a merge protocol that specifies how devices are to be added to the same group.

Groups are used to simplify management of who has access to what device. For instance, if Bob has a laptop and an iPod he might choose to merge the two into a group. Then when Alice meets Bob they both introduce their laptops to each other. This process allows Alice implicit access to Bob's iPod since it is in the same group as the laptop. UIA also allows for the concept of shared groups where two or more people can add people and devices to the group, allowing Alice and Bob to create a shared group for some of their devices.

Resolution of UIA names results in an Endpoint Identifier (EID), similar in function to the EID in HIP. The EID represents a device identifier separated from its location in the network. The EID may then be resolved in order to obtain the current location of the device. Unlike in HIP this takes place using an overlay network, whereas HIP uses DNS for this function. An EID is resolved by flooding a request through an overlay network until a path to the device is determined. UIA also proposes incorporating support for NATs through the use of hole punching or forwarding across the middlebox.

2.4.2 Eduroam

Eduroam is a service that enables university students and staff to move between universities and make use of network access at any participating university. Although originally a European initiative, universities from outside that continent have since joined.

The service consists of a hierarchy of RADIUS servers. Each university maintains its own RADIUS servers, with a local user database. These are connected through the hierarchy to a federation on the national level. Each national federation can, in turn, be connected through a larger federation.

While roaming, users can authenticate themselves to their home institution to get network access. Since each user is identified by a username of the form *username@realm* their username can be used to route their authentication requests back along the hierarchy to their home institution.

Access points in institutions use the 802.1x standard that includes the EAP standard of authentication. To authenticate themselves to the network a user will communicate with their home institution over a secure tunnel. They may then either provide a username and password using either of the EAP-TTLS or PEAP authentication protocols, or they can use EAP-TLS for mutual authentication between the network and the user using X.509 certificates. The visited institution handles authorisation for access to resources.

2.4.3 HINTS

The Historic Name Trail Service (HINTS) [86] is an attempt to solve the issue of personal-identity mobility, where people can change the online identifiers that are used to contact them, e.g. email address, SIP URI, etc. The goal of HINTS is to allow people to link the different identifiers they use over time, in a sort of name history. In this way, when Alice wishes to contact Bob, she can check the trail of identifiers to see what identifier he currently uses or what identifier he used at a particular time. In this way the service can give the illusion of a persistent identity. Since the linking of identifiers is completely voluntary a person may dispose of this identity at any time by not linking to their new identifier.

They describe two instantiations of the service. The first requires each user of the service to trust a single name historian, that allocates persistent identifiers to each person, maintains the links between each person's identifiers, and certifies the current identifiers used by a person at a given time when requested. Each person that wishes to use the service signs up for an account with the historian, obtaining a unique identifier from the historian in the process. They then voluntarily inform the service of the identifier they are currently using and update this when they stop using it. The service verifies ownership of identifiers using a challenge-response mechanism. In this way the service may operate without the involvement of the individual namespace providers.

The second instantiation of the service attempts to provide a more secure and robust solution. To achieve this they change to cryptographic identifiers, rather than identifiers allocated by the historian. The need to trust the historian is alleviated, as each of the namespace providers are expected to certify the owner of a given identifier to a public key. Proof of ownership of an identity is represented using an identity certificate, which certifies the identifier in question, for example an email address, the start time, end time, and public key of the owner. This certificate is allocated to the owner of the identifier in an out of band manner. The end time in this case corresponds to the expiry date of the certificate and may be extended by the service provider producing a new certificate. A given provider may end ownership of that identifier by issuing a revocation certificate or allowing the identity certificate to expire. The name historian can specify the duration of an association by use of a link certificate that specifies the actual length of time that the person was using that identifier for or a severance certificate that specifies the last date the association was valid.

HINTS should allow for identifier portability by clients with no alteration to the core Internet infrastructure, and a minor alteration to the client software. Specifically, the client needs to keep track of what identifiers a person is using at a given time and inform the historian when this changes. The authors don't specify a means of achieving this, but rather note that this could easily be achieved by the individual applications that are used or by maintaining a local database of identifiers that each client could alter and refer to. Clients that use HINTS are expected to check the name historian periodically for updates to a person's address. The authors expect that once a week should be often enough so as to maintain freshness while alleviating the load of the historian, but, by their own admission, don't have any data to back that up.

2.4.4 Universal Personal Telecommunications

Aside from telephone numbers that have been allocated to certain countries, the ITU has specified that certain numbers should be non-geographic. Universal Personal Telecommunications (UPT) is a proposal by the ITU to provide what it calls "mobile numbers" [115]. The idea is that each person would be allocated a UPT number, which would remain theirs regardless of the service provider they used to contact people. Each person would subscribe to the service at a UPT service provider. They would then be allocated an E.164 number under the 878 country code, with the area code identifying the UPT provider. All calls to the person's number would be routed through the UPT provider. Since the UPT number is non-geographic it should in theory be useful for providing global number portability.

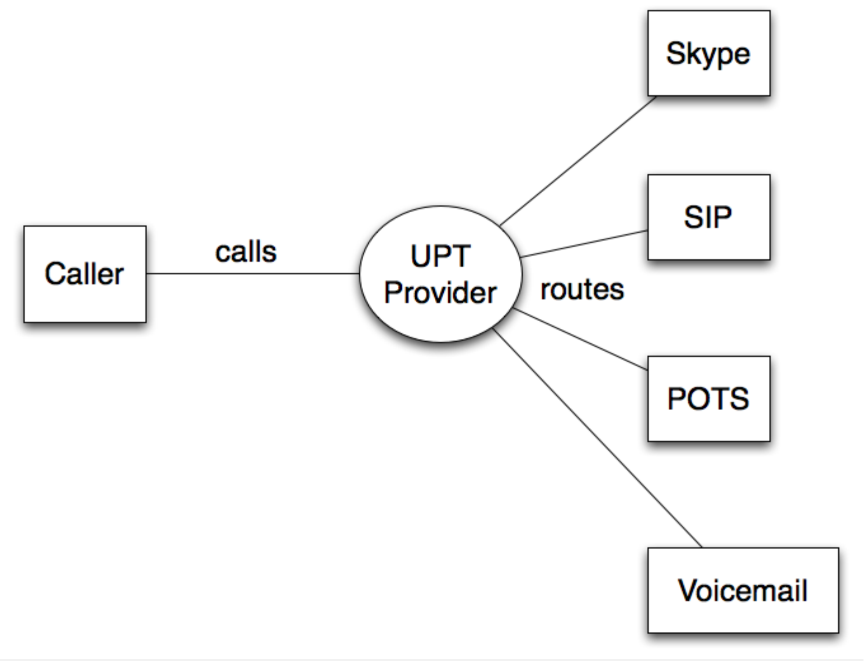

Fig. 2.9: The architecture of the Universal Personal Telecommunications system.

The subscriber must create a service profile, which dictates where their calls should be routed to. The service provider could support a number of different methods of routing calls. Service providers could, for example, decide to support the SIP protocol, in which case the UPT provider could route incoming calls over an IP network to the end point. Thus, UPT allows for the convergence between the legacy telephone network and the Internet. The service profile also allows the subscriber to specify their call routing preferences. This is important as it isn't a feature of the legacy telephone network. Unfortunately UPT hasn't been successful and only one provider has been allocated a number under the UPT country code. This provider, at the time of writing, seems to have abandoned plans for providing a UPT service and instead now offers an ENUM service using 878 numbers as the identifiers.

UPT in conjunction with a public ENUM system would allow for a non-geographic namespace of identifiers to be used not just on the legacy telephone network, but also on the Internet, and to provide a means of enabling convergence between them.

2.4.5 Universal Communications Identifier

The Universal Communications Identifier is an effort by the European Telecommunications Standards Institute to enable people to identify themselves uniquely and specify any special needs they have while communicating [63]. The standard specifies three main components; the identifier itself, a Personal User Agent (PUA), and a Service Agent. The identifier is a globally unique value and is composed of up to three components; a globally unique number, a set of optional labels, and an optional data field that can be used to provide metadata about the information in the labels. The Personal User Agent routes incoming calls according to user-defined preferences, while the Service Agent acts as a layer of abstraction to the PA from the details of the network implementation.

The identifier itself is composed of three components, a handle which is a name that the user suggests for themselves, a globally unique number, and a list of optional preferences. The preferences are there to facilitate special features; for example, if a person is disabled then they may be used to specify whatever requirements they may need. The unique number is allocated by a global body to an individual and is considered theirs for life. To date UCI has not seen deployment and as far as we know none have been allocated, but this may change in the future with the deployment of next generation telephony architectures.

Like UPT the UCI specifies the use of profiles as a means of improving the user's control over their call routing preferences. The UCI system was proposed in the late nineties around the time of two other personal communications architectures, the Mobile People architecture and the Universal Inbox which was a component of the ICEBERG project, which we will discuss in the next two sections. The goal of these was to enable a shift from telephony architectures where call routing happened in the core of the network where it could mostly not be influenced by the individual to more people-centric systems where routing preferences could also be specified by people at the edges.

2.4.6 Mobile People Architecture

The Mobile People Architecture [87] is a solution for personal mobility that originated in Stanford University. The architecture has a number of goals including:

1. Allowing people to be reached regardless of the communications device they are using or the constraints this devices places on the type of communications it can support.
2. Maintaining the privacy of the individual with respect to their current location.
3. Facilitating measures against spam and other such annoyances. This includes allowing the user to specify what types of communications should be directed to them and who should be allowed to contact them.
4. Converting between different message types. If one device doesn't support a particular type of message or a person wants a message in a certain format then the architecture should facilitate its conversion to the appropriate type where possible.
5. Ensure that there aren't any constraints placed on the architecture that would prevent it from handling future network protocols. This includes deploying well defined interfaces in order to provide a way for these protocols to hook into the architecture.

The architecture introduces two main components to the existing Internet architecture. The first is the concept of the person layer, which provides a new level of identifier that identifies a person. This mainly consists of a name service that resolves a person's unique identifier (POID) to an application specific address (PASA). The second concept is the personal proxy. The proxy is responsible for routing, filtering, and

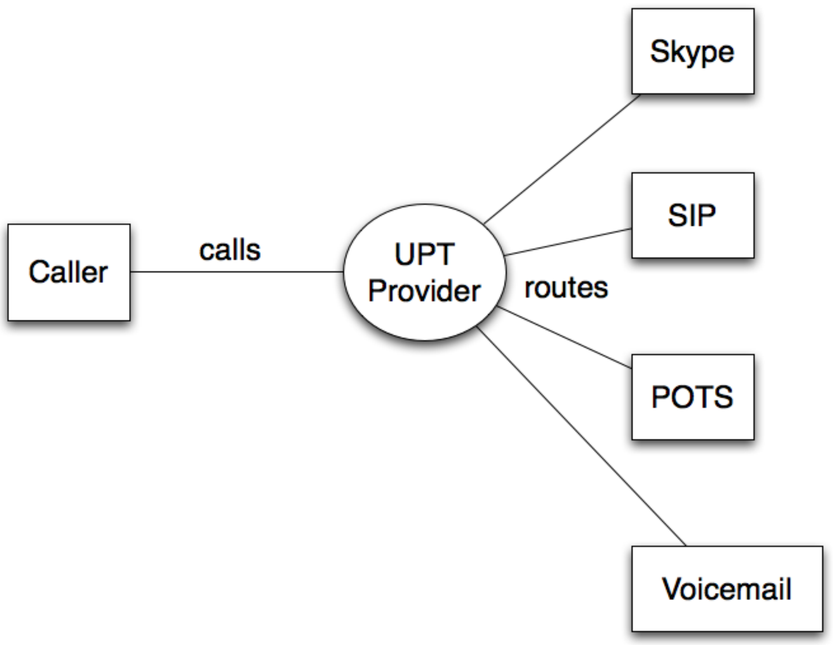

Fig. 2.10: The Mobile People Architecture.

transcoding messages. The proxy also allows a person to hide their location in order to protect their privacy. The PASA points to a personal proxy using an application specific identifier, such as a SIP URI or an email address. The personal proxy has a rules engine that specifies how each of the incoming communications are to be filtered.

2.4.7 Universal Inbox

The Universal Inbox [100] is a component of the ICEBERG project undertaken at Berkeley. The idea behind the universal inbox is that a person should be able to be mobile across different services and devices and have their communications follow them. The high level goals of the universal inbox are similar to that of the MPA's. However the requirements and design of the two differ substantially.

In the ICEBERG proposal each person can specify their location and routing preferences for their incoming communications. Centralised network elements handle the routing and protocol translation that allow sessions to be routed based on a person's device, location, and the identity of the person communicating with them. These elements also allow for things like codec translations and documents to be translated from one type to another. The MPA, in contrast, extends the existing architecture at the edges, whereas the ICEBERG proposal is more of a network convergence solution, where individual gateways, network elements, translate between different protocols. So for instance, the Skype network and the Google Talk network would be connected via a number of middleboxes that would translate between the Skype protocol and the Google Talk protocol. The ICEBERG architecture also specifies that a centralised database would be put in place to keep track of the current mode of access of each person. This is in direct violation of the MPA's privacy goal. Also, unlike MPA centralised network components provide translation between different file types, which increases the cost of deployment substantially compared to MPA.

2.4.8 GrandCentral (Google Voice)

GrandCentral, now Google Voice [1], is an Internet based voice service that allows a person to link one or more numbers to one or more devices. Additionally a number of telephony services are provided by the GrandCentral service, such as voice mail, call forwarding, and call screening. The GrandCentral system functions similarly to a UPT system. All calls to a given number are routed through the service provider. The user may specify a set of rules that take variables like time of day or incoming phone number into account in order to route calls to a given device or number of devices.

GrandCentral has been cited as an example of a "Telephony 2.0" service, since it marries telephony to the web. Users of the service can access the service via the web in order to create, modify, and delete call routing rules. They can additionally access their voicemail and monitor their calls via the web. GrandCentral also allows for the interoperability of Internet telephony and the PSTN as users can access the service via a SIP client.

2.4.9 PhoneGnome

The PhoneGnome is a telephony device that allows a regular telephone to be used to contact Internet telephony services, via SIP, and also Skype. The device has a socket for a regular telephone to plug into it as well as an Ethernet cable [7]. The device can be configured by the user to connect to a number of different Internet telephony services. People that sign up to the PhoneGnome service can then define contacts and the SIP URIs, telephone numbers, or Skype IDs at which they are accessible. These preferences are stored online. All user calls are routed through the PhoneGnome service, which decides the best way to route that call to its destination. The metric for this decision is generally based on cost. For example, if a person's contact has a PSTN connection and a Skype ID registered with the PhoneGnome service, the Skype ID will be tried first since Skype to Skype calls can be made at no cost to the user. If this fails then the system will resort to the next identifier in the list, in this case a PSTN number. The PhoneGnome service also provides a number of other services, such as call recording, voicemail, and voicemail messaging, where a voicemail message can be sent via email to a person.

2.4.10 Discussion

This section outlined a number of existing attempts to solve the problem of personal mobility and naming. The first two approaches were decentralised naming systems that attempted to solve issues related to personal naming and authentication. The Unmanaged Internet Architecture builds a private namespace for a person's devices and then allows them to share that namespace with others. In this way a relationship between the person, their devices, and their relationships with others may be expressed through the naming system. People can assign their own names and meanings to these relationships. This has obvious advantages for personal naming as a person's relationship to devices and people are expressed. Thus, the names are likely to be of high mnemonic value.

HINTS takes a different approach in that it attempts to link a person's ASIs, thus creating a historic trail of identifiers that will ultimately lead to that person's newest identifier. Thus, if Alice knew that Bob's email address used to be *bob@example.com* during a certain time period she can find his most up to date email address by using this and the HINTS system. This type of system sidesteps the issue of creating an identifier for people and instead allows their various identifiers to be linked together.

Both of these systems have their advantages. However in the case of UIA the naming system is only suitable for naming devices, and can only be used when a person has one of their devices with them. This may not always be the case. Furthermore, the naming system assumes that people can meet and exchange identifiers face to face. However often people will email or otherwise communicate with people they only know online. The system does not provide a way to bootstrap an identifier exchange in these kinds of situation.

One reason people may wish to change identifier is so that they can start afresh. For instance if an email account has become plagued by spam then changing email addresses allows the recipient to reduce the level of spam they receive. In these instances it would be desirable to allow everyone that knew your old email address legitimately to learn the new one. In HINTS this is not possible as the trail exists from one to the other without any kind of access control. Knowledge of a person's identifier at one time would lead to their current one.

Each of the other systems we outlined fell into the broad category of telecommunications mobility solutions. They each advocated some kind of middlebox to be put in place to route incoming calls according to some user-defined profile. In the case of MPA and ICEBERG these middleboxes even performed content analysis and transcoding. These two solutions differed in terms of where they required the middlebox functionality to be placed. ICEBERG incorporates the functionality into its core architecture while MPA advocates placing it in a dedicated server on the edges of its network architecture.

The UPT and UCI solutions were proposed by standards bodies, which would have the political clout to implement them. Yet these solutions remain largely unused. We believe this is because they require user involvement and would cost the user money. The functionality provided by them has become largely unnecessary for the vast majority of people since the implementation of number portability in a lot of countries.

Of the solutions proposed by companies outside of the established telecommunications industry, both Google Voice and Phonegnome are promising. They both have the advantage that they provide a means for people to make cheap calls, and have other tangible user benefits such as the ability to route incoming calls according to their preferences. The advantage they have over UPT and UCI is that through providing convergence between the Internet telephone systems and the telecommunications infrastructure these solutions bring obvious benefits to the user and could sustain themselves based on their business models.

2.5 Identity Management

The area of Identity Management covers a body of research into issues relating to the identity life cycle. This covers a lot of different types of research including lightweight web-based authentication and authorisation solutions, the issue of authentication and authorisation across different organisations, usability issues, and privacy issues. This section will outline some of these solutions and their relevance to Sobriquet.

2.5.1 OpenID

Recently there have been a number of systems that attempt to solve the identity management issue for the web. These systems can be described as lightweight systems, because unlike the enterprise solutions like Liberty Alliance [3], they attempt only to solve the problem of authenticating an individual at multiple endpoints using a single account. Some systems that fall into this category include Yadis [14], OpenID [6], and LID [4]. While the individual solutions differ in their implementations, the general design is similar in each case.

OpenID [6] is a decentralised identity management system that was proposed by Brad Fitzpatrick of LiveJournal to allow people to allow people to login to different websites using the same identity. The goal is to eliminate the need for a separate username and password for each website to which a person wants to authenticate themselves. An OpenID identifier is generally a URL, which includes among its components the domain of the identity provider, the protocol that the authentication will take place over, and the username of the person at that identity provider. When people sign up to a new service using an OpenID they are telling the service provider which provider will handle authentication on their behalf.

The goal for OpenID is to be backwards compatible with existing web browsers. In other words an Identity Provider should be able to enable OpenID without requiring the user to download a new web browser for it to work. Analysis of OpenID has revealed a number of flaws [83], such as the lack of requirement to perform authentication over a secure protocol, attacks based on Domain Name System threats, susceptibility to phishing attacks, and the privacy issues that result from authentication passing through a single Identity Provider.

2.5.2 OAuth

OAuth is a protocol that emerged from a community of web developers with a view to solving the problem of allowing access to private resources to third parties [72]. Facebook use OAuth in their Facebook Connect service. The protocol typically involves a client, a server, and a resource owner. The client attempts to access server resources

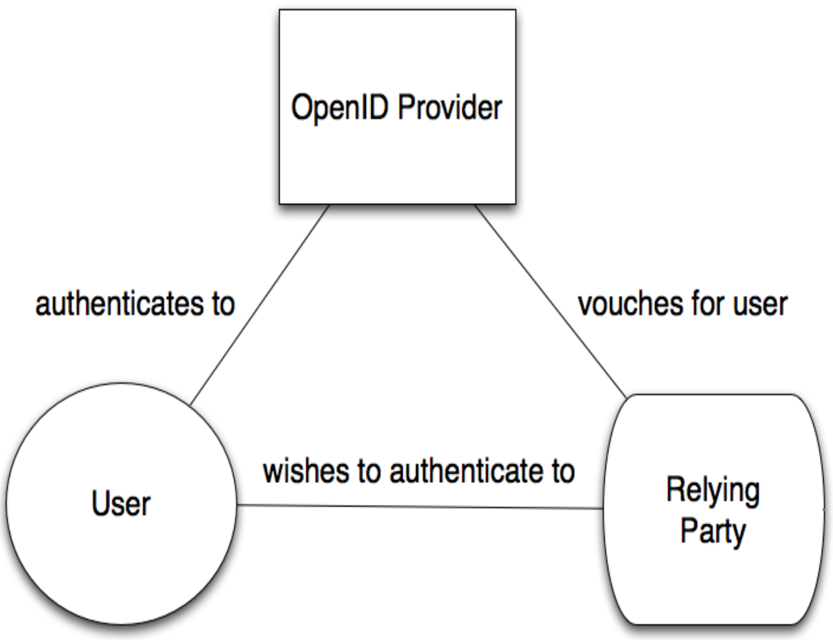

Fig. 2.11: The relationship between OpenID Provider, User, and Relying Party.

on behalf of the resource owner. The protocol has two components. The first is a process that allows the resource owner to authenticate themselves to a server so they can delegate access to the client. The second is a means for performing authenticated HTTP requests using two sets of credentials, one that authenticate the client to the server and one that identify the resource owner on whose behalf the request for access is being made.

As an illustration of the need for the protocol its designers give the example of a printing service that allows users to print from a photo sharing service. OAuth allows the user, the resource owner, to delegate access to the printing service that allows it to access photos in their account at the sharing service without divulging the credentials, e.g. username and password, they, the user, provide to authenticate themself to the photo sharing service. If OAuth were to be used in this scenario then the client, in this case the printing service, would need to have signed up for a set of credentials for accessing the photo sharing service, ahead of time. To access the photos the printer will first make a request to the sharing site to obtain a token for accessing the user's photos. The printer will then redirect the user, using HTTP redirects, to the sharing site sending the credential and a callback URI along with the request. The user will authenticate themselves to the sharing site and, when prompted, will approve the request to access their photos. The sharing site will then redirect the user back to the callback URI and their session on the printer site can continue.

2.5.3 Federated Systems

A federation is a group of entities that agree to use a common set of standards and to trust each other's authentication of entities within the system. This is necessary in order to allow users of one service access to the other. As an example, consider a situation where a student of one university wishes to take out a library book in a different university's library. In this scenario each university would put an authentication procedure for their students in place. Then when a student roams from university A to university B, university A can authenticate the roaming student on behalf of university B. So in order to gain access to the resources of the library in university B a student from university A would authenticate themselves using whatever authentication credentials they received from university A.

Extending the example a little further, if we wish to allow roaming lecturers more access than students we must also define the role of each at their respective universities. In this case there needs to be an agreement between the two universities about what the various terms mean. For example, one university might give lecturer status to guest lecturers whereas the other may not. Since this information may factor into decisions about access control a common definition is needed. One way to do this is to establish a contract between each university outlining these definitions. However, this creates a lot of complexity in terms of management of these relationships and may make the authentication procedure unmanageable. A federation where each university signs up to a set of common definitions and policies allows for this process to be streamlined.

2.5.4 SAML

The Security Assertion Markup Language [99] is an XML based standard for describing and exchanging authentication and authorisation information between different administrative domains. At the time of writing the latest version is SAML 2.0, and is generally used in web based systems, with transactions taking place over HTTP [98], typically using SOAP [118]. In a system using SAML each user is assumed to be enrolled at an Identity Provider. As with other Identity Management systems, the user will authenticate themselves to their Identity Provider, obtain a session token that allows them to assert that they have recently authenticated themselves to their provider, and have the provider vouch for their identity to individual service providers.

A typical use case is as follows. A user will request a resource at the service provider. The service provider will then check whether that resource requires authentication or not and whether the user already has a security context at the provider, i.e. that they already have an authenticated session established. If not the provider will request that the user authenticates themselves with a unique identifier for that transaction. The user will then present this identifier to their identity provider along with a request that they vouch for them. The identity provider checks whether the user has a valid security context with them. If not the user must first authenticate themselves to the Identity Provider. The Identity Provider, upon receiving assurance of the user's identity, will issue a document containing the assertion that that user has authenticated themselves and is who they claim to be. The user will then forward this document to the service provider, specifically the service that processes security assertions. Upon validation of the assertion a security context between the service provider and the user is established. The resource may then be requested and once the user has the authorisation to access the resource the request will be processed successfully.

SAML defines four main components. *Assertions* define the security information to be exchanged or requested. These are exchanged according to a given *protocol*. SAML protocols are typically comprised of a request and a response. *Bindings* define how the individual SAML messages are bound to communications protocols, such as SOAP requests over HTTP. Finally *profiles* define how a given set of assertions, protocols, and bindings operate together to perform a given use case, such as the one outlined above. A number of standard profiles have been defined. Individual assertions relate to a given subject. In version 1.0 of the standard these were identified using email addresses. In version 2.0 support for pseudonyms was introduced allowing a person to present a different identifier to the service provider each time. These pseudonyms are generated by the individual service providers and they may thus link different transactions to the pseudonym and in turn to the identity of the subject.

Liberty Alliance

The Liberty Alliance [3] is an open standards group comprised of a number of large technology companies. Their goal is to develop standards that allow people to conduct online transactions while protecting both their security and privacy. To date they have released a number of standards concerning the deployment and operation of federations. Their standards cover everything from the language used to exchange authentication and authorisation information, SAML, to the methods used to establish trust between entities in a federation.

Shibboleth

Shibboleth [10] is an identity management framework in development under the Internet2 project that builds on the SAML standard. The primary goal of the framework is to provide access to resources across different administrative domains. To achieve this goal it provides two main pieces of functionality. Firstly it allows for cross domain single sign on. This allows a person from institution A to authenticate themself as being from institution B. In this way people can travel between different organisations and authenticate themselves using a single identity provider. Secondly, in order to gain access to resources Shibboleth allows information about the user to be released on an as needed basis. This may be controlled both at the identity provider level and also by the user.

Shibboleth adopts a federated approach where each member of the federation has agreed to a set of procedures, definitions, and protocols to be used for the exchange of identity information. As such, when a person wishes to gain access to a resource they must first authenticate themselves in order to prove that they are a member of the federation. Additionally they may need to release some information about themselves, such as their role within their home organisation, in order to prove they have a right to access a given resource.

When an unauthenticated user wishes to access a protected resource they will first be redirected to a service known as the *Where Are You From?* service. Here they will be presented with a list of possible home institutions, i.e. other members of the federation. Since Shibboleth is based on the SAML standard, which is used mainly for web transactions, this redirection will typically take place within a web browser. Once the user has selected their home institution they will be redirected there for authentication. This authentication process may be defined by the individual institutions to meet their own criteria; however the authentication information exchanged between the institutions will be of a standard type. Once the authentication process has taken place at the home institution the user will be redirected back to the resource they requested.

2.5.5 Metasystems

In his article, The Laws of Identity [43], Cameron, Microsoft's "Identity Architect", attempts to outline certain fundamental truths about identity and identity management. One of the claims he makes is that there is not likely to be one identity system that is suitable for all applications since each application has its own view of what identity is. Instead he proposes that we develop identity systems as needed and provide a familiar interface to the user and developers of these systems. This has been the approach adopted by two systems, Cardspace from Microsoft and the Higgins identity framework that is now a part of the Eclipse project. Cameron coined the term Identity Metasystem to describe this approach, as such a framework attempts to provide useful features to a generic identity system, rather than adopting a single approach.

Both Cardspace and the Higgin's framework take a user-centric approach providing an analogy to users in the form of cards, known as i-cards in the Higgin's framework and InfoCards in Cardspace. This is meant to be analogous to a person presenting an identity card in order to gain access or privileges in the real world. These cards reflect a person's individual identities and means of identification. Each implementation allows for plugins that interface with the individual identity systems and expose generic interfaces to the user interface component. This may then enumerate the various identifiers a person has and allow them to perform a set number of actions on them.

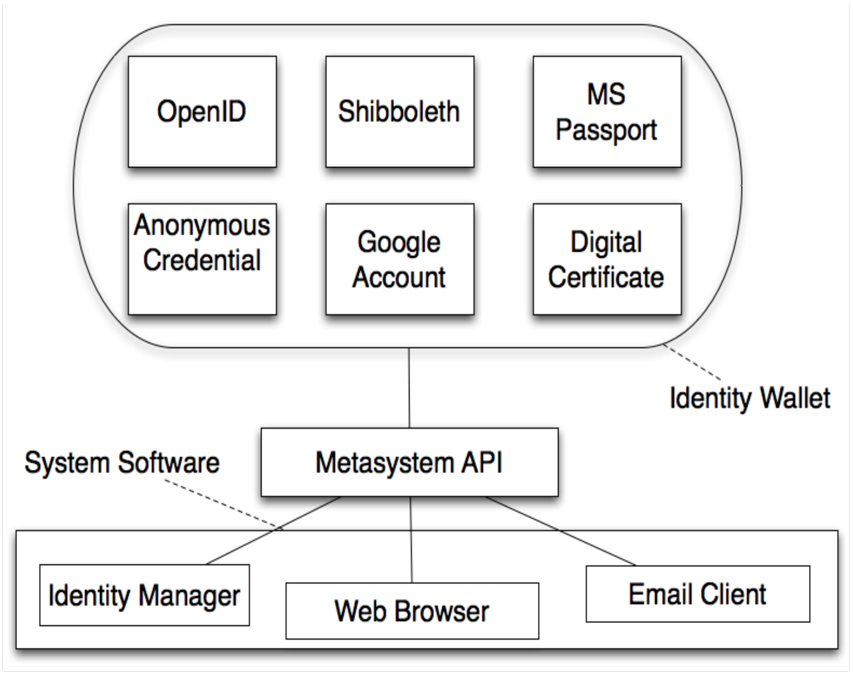

Fig. 2.12: General architecture of an identity metasystem.

2.5.6 Discussion

The Metasystem approach recognises the complexity of the authentication issue and acknowledges that there is never likely to be one omnipresent authentication solution on the web. For this reason it aims to interoperate generically with many different types and present a common interface to its users. We have outlined a number of different approaches to identity in this section. Of these we have divided them into subsections that fulfill different requirements. For example, we discussed Federated Systems which are designed to provide the means for expressing, exchanging and authenticating identity information and how this should be managed. OpenID on the other hand provides a more lightweight approach that allows people to authenticate themselves to different service providers under a single account.

Several large service providers, such as Google and Yahoo, have adopted OpenID as a means of letting their users log into third party applications without needing to sign up for a new account. This will allow web applications that accept OpenID logins to provide a lower barrier to usage for the large number of people with, for example, Google and Yahoo accounts. They can also potentially outsource their authentication to these large service providers and do away with the need to store usernames and passwords themselves. Similarly OAuth is being used to provide authorisation tokens that allow limited access to user resources between service providers. Since it would seem unlikely that OpenID or OAuth would be replaced for these tasks by something like Liberty Alliance given that many systems already support and use OpenID today, the assumption that multiple types of identity systems will prevail would seem reasonable.

However the main issue with both of these solutions is that the centralised Identity Provider has total control over a user's identity. If that Identity Provider ceases to exist, bans a user, or is not accessible globally then that user will lose the ability to log into any account they have linked to that identity. This is a real concern as, for example, Facebook is currently blocked in several countries worldwide including The

People's Republic of China [18] and so Facebook Connect users would be unable to use their accounts there.

Furthermore, neither the OpenID nor the Federated Systems attempt to address the problem of privacy in a convincing way. We are not aware of any efforts underway within OpenID to correct this, and the Federated Systems have so far taken a weak approach to privacy. The crux of the issue is that all logins are required to go through the Identity Provider, which can link every session by a given user. Next we will outline the evolution of public key cryptography, which has provided technologies that could be used to bring more privacy to these types of authentication systems.

2.6 Public Key Cryptography

2.6.1 Overview and History

Public key cryptography was first publicly proposed in 1976 by Diffie and Hellman [56]. This idea redefined the field of cryptography as it allowed for new key exchange and authentication protocols to be built, which would not have been possible otherwise. In 1978 Rivest, Shamir, and Adleman became the first to publish an algorithm for public key encryption, which came to be known as RSA.

2.6.2 Public Key Authentication

According to Diffie and Hellman's vision an entity's public key would be made available to the world in a public directory along with their name and address. When a person, Alice say, wants to send a message to her friend Bob, she would consult this directory to find his public key. At the time it was assumed that the directory would be read only. They assumed that the public file could be trusted, and they did not provide a way for secure dissemination of public keys.

In 1978 Loren Kohnfelder [81], in his bachelor's thesis, outlined his idea of a digital certificate. Under his scheme all the contents of the public file would be digitally signed by a trusted third party, which would ensure its results could be transferred without an attacker altering them in transit. This formed the basis for the development of Public Key Infrastructures.

2.6.3 Public Key Infrastructure

An X.509 public key infrastructure [49] provides an infrastructure for the certification of public keys. At the root there are one or more Certification Authorities (CA) that are deemed to be universally trusted. Such an authority is tasked with the certification of identities and public keys. This may require the entity to present themselves at the office of the CA with valid identity documents. Once they have completed the registration procedure the CA will certify their public key, a process that will produce a digital certificate with the public key and identity information corresponding to that entity, signed with the CA's own private key. Anyone who knows the public key of that CA and trusts it to certify other public keys, may then verify the authenticity of the certificate and securely access that entity's public key. This certificate could then be distributed in order to facilitate the exchange of public keys.

The PKI model is the basis of public key certification for a number of technologies, perhaps most notably the Secure Sockets Layer (SSL) [68] and its successor the Transport Layer Security (TLS) [55] protocols. However, the vision of a global directory of public keys envisioned by Diffie and Hellman's paper has yet to be realised with a PKI.

2.6.4 SPKI/SDSI

In [62] Carl Ellison detailed several issues with the notion of identity as used in PKI terminology. This work formed the basis of a collaboration with several others, including Ronald Rivest and Butler Lampson, that eventually led to the SPKI/SDSI effort

[61].

Naming in SPKI/SDSI is motivated by the contention that the dream of a global X.509 PKI is unlikely to ever exist. They argue that the information that was envisioned to be made public in such a system would never be made public in reality as it is valuable and confidential. They further argue that the idea of a global name is unlikely and instead that names, in so far as they are necessary, should be local.

They argue that keys are what computers should use to verify identity. The key would act as an index to some database of permissions that the key holder is authorised for. This is different to the X.509 notion that a digital certificate should map a key to a name, which forms the basis for deciding permissions.

Names they contend should be used by people to ascribe their notions of a key holder's identity to a key. So a person Alice might decide the key belonging to her friend Bob should be given that name. This is represented in SDSI's s-expression syntax as so "(name Bob)". This says that the namespace of Alice contains the name Bob. Alice in turn can share her namespace with her friend Fred, who may choose to incorporate it into his namespace and will therefore refer to Bob as "(name Alice Bob)". In this manner hierarchies may be built in a bottom up fashion, which is different to X.500's top-down approach that stems from the root.

SPKI hasn't seen large scale deployment, though it was a part of the UPnP Security DCP 1.0 standard. That standard did not see the deployment that was anticipated. In an update to the standard the authors mentioned that SPKI's lack of support in the industry likely contributed to the lack of deployment of UPnP Security 1.0 [85]. This lack of industry support is likely a hindrance to SPKI's deployment in general.

2.6.5 Web of Trust

Like SPKI the web of trust model [97], as proposed by Phil Zimmerman the creator of PGP for use in that system, is a decentralised method of certifying identities to public keys. In the web of trust model any person may certify the public key of a person they trust. In this manner a web-like social network structure is created and we can traverse this web to find the public keys of people who are trusted by people we trust. In theory provided the graph is fully connected we could traverse the web and find the public key of anyone who uses the system.

In reality questions have been raised about the transitivity of trust, if Alice trusts Bob and Bob trusts Carol, should Alice trust Carol? In the real world this may not always be the case. Furthermore, unlike in a PKI there is usually no policy specifying how a certification is to be performed. Often this will happen at key signing parties, where a group of people meet up and certify each others' public keys. However, this results in far less accountability than in a PKI, where the Registration Authority is responsible for ensuring certification adheres to its policies.

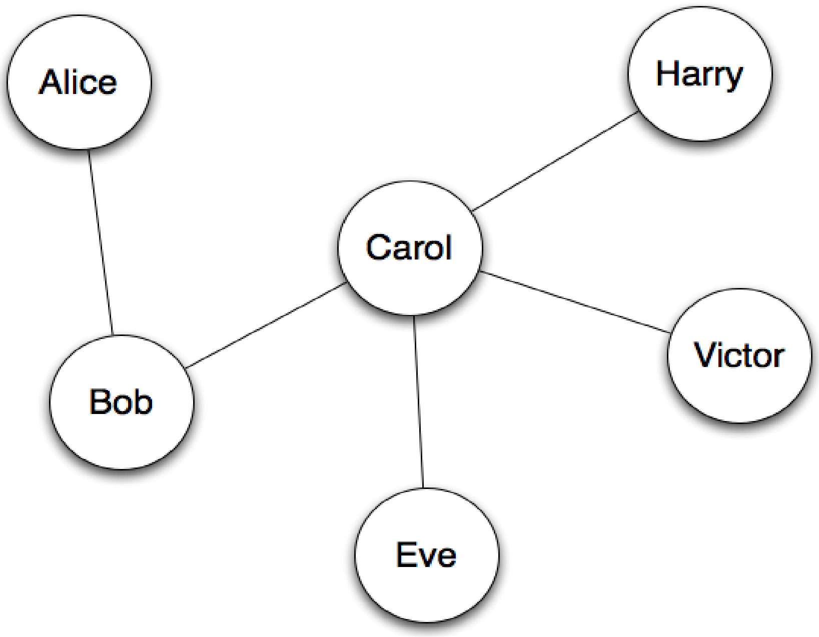

Fig. 2.13: A web of trust. In this instance Carol is a hub. Bob must trust that Carol correctly certifies public keys for Harry, Victor, and Eve for the web to function. Alice must trust that Bob's trust in Carol is not misplaced for her to obtain the same public keys.

2.6.6 Secure-Shell Authentication Protocol

In the Secure Shell (SSH) [123] public key authentication model, when there is no PKI available public keys are exchanged without authentication. There is a leap-of-faith taken in the assumption that there is no active attacker capable of interfering with the exchange by substituting his own public keys for those exchanged. Under the SSH model, public keys should be cached and bound to either the network address or the domain name of the other party. Then, if in a later session the public key changes, authentication will fail. The SSH model also guarantees that if there was an attacker present for the initial exchange of public keys, that the attacker must be present for all subsequent sessions or the two parties to the exchange will notice that the public keys have changed.

The SSH trade-off between requiring strict authentication of public keys, as in the Public Key Infrastructure model, as opposed to allowing people to assert their identity on first contact and assuming that the channel is free of attackers allows for a more economic approach to security. As we mentioned in the section on Public Key Infrastructures, the major cost involved in running and maintaining these systems is the certification of identity, which often requires real world presence. SSH on the other hand allows for public key authentication with looser constraints on the binding to identity. However, there may be a cost associated with rekeying as the identifier presented, for example, by a server to its users may change, and the effect of such a change will be indistinguishable to a man in the middle attack. This change will need to be communicated to the users somehow. In general SSH can bring security to applications where running and maintaining a PKI is either not viable or not desirable, though the protocol can make use of a PKI when one is available.

2.6.7 Petnames

Petnames [111] are a compromise between the need for a global flat namespace, usually based on cryptographic identifiers, and a memorable name. They function as a memorable name for a long, difficult to remember identifier. The memorable name is referred to as the petname and only has meaning for the person who invented it. Thus, petnames are not themselves resolved or shared, but rather act as a convenient way of referring to an entity. An analogue is an address book in a mobile phone, where a person assigns a name to a phone number that they don't wish to remember.

They were proposed as an alternative means of mapping public keys to names. The first time a person encounters an entity they cache that entity's public key. This may be certified by a CA or not. The user then assigns a petname to it. When a host identifies themselves using that key in subsequent interactions it will be mapped to its petname and presented to the user in this way by the petname software. For instance the Firefox petname plugin shows the name in green beside the address bar if a host has properly identified themselves. If a user has previously established a secure association with their bank say, then if an attacker attempts to launch a phishing attack the petname software will not identify the phishing site as the bank.

This also presents a usable workaround for sites that use secure HTTP (HTTPS) connections to establish a confidential channel for login sessions. If the HTTPS connection fails due to an expired certificate, for example, but the host's public key is still functional then the petname software would still indicate to the user that the identity of the site they are communicating with is the same as the one in every previous session. This provides more information to the user than the method used in many browsers today, where the browser prompts the user to risk establishing the connection but does not indicate whether the public key of the site has changed.

2.6.8 Better Than Nothing Security

The goal of the Better Than Nothing Security (BTNS) working group within the IETF was to produce a standard for using IPsec between nodes on the Internet without the need to rely on an authentication infrastructure [114]. As implied by the name the theory is that any level of authentication and confidentiality is better than having none at all. This is achieved by noting that cryptographic protocols, such as Diffie-Hellman, are secure in a scenario where there is no active attacker. This is similar to the observation made by the designers of SSH.

2.6.9 Key Continuity Management

Key Continuity Management [69] aims to create a more usable encryption and authentication system. The approach they advocate is to automate the generation of public/private key pairs and automatically perform all signing and encryption on behalf of its users. Instead of relying on infrastructure for key distribution the system attaches self-signed certificates to outgoing email. This allows other email users to verify emails sent by them. The first time the system encounters a public key certificate it caches it. If emails come from the same source as a certificate but are not signed with the correct public key it indicates this to the user by flagging it. This prevents forging of emails from known sources. The system does not, however, detect forging of emails from previously unknown sources.

2.6.10 Cryptographically Generated Addresses

The work due to Aura [25] on Cryptographcally Generated Addresses (CGA) describes a method for binding public keys to IPv6 addresses in the Secure Neighbour Discovery (SEND) protocol. Addresses are generated in part, from the hash of a public key, which forms the rightmost 64 bits of the IPv6 address. The corresponding private key signs messages originating from that address. Ownership of a public key can be asserted and verified by checking the truncated hash of the public key against the 64 leftmost bits of the address.

Addresses in CGA are self-certifying, that is the authenticity of the associated public key can be verified by virtue of the fact that the IPv6 address is known. However, in cases where the IPv6 address of a device is not known ahead of time, CGA does not provide a way of securely obtaining this information about a given device.

2.6.11 Discussion

We briefly introduced the history of public key cryptography in this section. We showed how the evolution of research led to the development of PKI systems, as a solution to the problem of public key authentication. We then showed some efforts that sought to make the solution to this problem more practical. We also looked at the SPKI approach of using public keys as identifiers and allowing people to associate their notions of the identity of the key holder to that key. This allows the same key to be associated with multiple names and reflects the reality that everyone may not have the same notion of any given person's identity as everyone else. The Web of Trust model of public key authentication aims to decentralise the functionality of the CA in a Public Key Infrastructure. However public keys are still bound to identifiers.

Regardless of their mode of certification each of these systems bind information to a public key through the use of a certificate, which is made public for retrieval by interested parties. The SSH and BTNS approaches adopt a means of exchanging public keys that assumes that on first communication there will be no active attacker present on a channel. This allows them to eliminate the need for a directory of public key certificates since each participant provides their public key directly. This assumption allows for public key authentication infrastructures in situations where providing a PKI is not practical and the associated risks are deemed acceptable.

2.7 Privacy Enhancing Technologies

In 1985 Chaum highlighted the issues involved with identification and privacy in networked environments [46]. He used the term "dossier society" to describe the situation where information about individuals and their interactions with other people and entities in such environments could be monitored and mined in order to infer information about their lifestyles, habits, location, and relationships. In this article he outlined a number of technologies which could facilitate the authentication required in such an environment, while protecting the privacy of its users. Chaum is responsible for introducing both the Anonymous Credential [46] and the Group Signature Scheme [45] in the literature. This section takes a look at the state of the art of these systems.

2.7.1 Anonymous Credentials

Anonymous credentials are a type of cryptographic token which allow people to reveal identity information about themselves anonymously. They were first proposed by Chaum in 1985 [46]. Chaum outlined a number of issues with the use of digital certificates for authentication purposes. Authenticated transactions may be linked, leading to a situation where a person leaves a trail of information about themselves wherever they go. Each entity they authenticate themselves to sees the full details of their identity in the digital certificate presented. Sometimes this is necessary, but often it is not and neither is it desirable. To buy something online, for example, it may not be necessary for the online retailer to know anything about a customer other than his or her ability to pay for the goods. In the real world, cash provides a level of anonymity to customers, but before anonymous credentials no such analogue existed.

Modern anonymous credential systems, such as those proposed by Brands [38][39], allow for the selective disclosure of the information contained within a credential.

For instance, say that Alice obtains a credential from a governmental organisation that contains her name, age, and an identification number that identifies her uniquely between governmental organisations. This credential has value to others as it has her age, but Alice may be reluctant to show it to third parties, since it also contains a number that identifies her uniquely. Anonymous credentials allow her to reveal the age field, without disclosing the other values, while providing the requisite proofs that identify her as a valid holder of that credential.

One issue that is raised with Anonymous Credential systems is that of transferability. While this property is beneficial in some applications it may not be in others. Say, for instance that governments issue credentials specifying the age of a given person as an anonymous credential. Some age restricted websites, such as those pertaining to the sale of alcohol may wish to require such a credential before providing access to their site. In theory if Alice meets the age requirements and Bob doesn't, she could give the credential to him to allow him access. Since the credential is anonymous there is technically no way for the issuing authority to find out who had lent it to Bob if the fraud were uncovered.

To tackle this issue, proponents of anonymous credential systems propose encoding sensitive information, such as biometric information, into the credential. When a credential is to be presented, part of the proof of ownership requires a proof of knowledge of each of the fields of the credential. The assumption is that while Alice may have no qualms about providing Bob with her age credential she may not wish him to have access to sensitive information about her.

The system due to Brands also provides a way for credentials to be blacklisted. By maintaining a revocation list at the issuing authority the holder of the credential can be required to prove that they are not on this list before a credential will be accepted in a transaction.

Credentica is a company that formed out of the research undertaken by Brands for his PhD thesis with the aim of bringing his anonymous credential system to market as the product U-Prove [5]. The company is now under the ownership of Microsoft. The U-Prove technology was an implementation of the anonymous credential system described in Brands's thesis and provided features such as the selective disclosure of credential fields, anonymous revocation of a credential, and the ability to prove binary relations on fields in a credential.

Brands Credentials

Under the Brands scheme [38][39] credentials are issued to applicants by trusted third parties known as Credential Authorities. Digital credential holders *show* credentials to verifiers. Credential *issuing* and showing are specified by protocols within the scheme. We will now describe the scheme from the point of view of Alice, the credential holder, and Bob the credential verifier.

We assume the existence of a Certification Authority (CA), that is a trusted third party that has the authority to certify certain attributes about Alice. These attributes can include any data. The CA binds a set of attributes to a public key, for which only Alice knows the corresponding secret key. When she is showing her digital credential Alice will use this secret key to sign a nonce and prove that she is the holder of the credential and prevent others from claiming ownership. Alice can selectively reveal to Bob any number of the attributes in her issued credential during the showing protocol. The showing protocol consists of a proof of knowledge of the secret key corresponding to her public key, and that the attributes in her digital credential have the properties she claims.

In order to maintain the privacy of the credential holder the CA should not be able to link the signature on the credential and the public key Alice supplies to the set of attributes it certifies. The assurance the scheme provides is that assuming those attributes are not somehow unique to Alice then even if Bob and the CA collude they cannot link the credential shown to Bob to the one issued to Alice, even if they compare the issuing protocol transcript at the CA to the showing protocol transcript she undertook with Bob.

To achieve this Brands makes use of a protocol known as restrictive blinding. The scheme defines an issuing protocol that provides the credential holder, Alice, with a blinded public key and a blinded certificate. The blinding scheme prevents the issuer from linking these to the showing protocol. It also prevents Alice from cheating and obtaining a certificate on a different public key than the one she provides to the issuer, or changing the encoded attributes of the credential.

2.7.2 Group Signatures

Gsroup signatures are defined as follows by Camenisch and Groth in their paper [42]:

*A group signature scheme involves three types of parties: members, non-members and a group manager. It further consists of five algorithms KeyGen, Join, Sign, Verify, Open, and Revoke. The key generation algorithm produces (**vk**, **gmsk**) ← KeyGen() as output, where **vk** is a public verification key and **gmsk** is the group manager's secret key. If the group of members is fixed, we may assume that the algorithm also outputs a vector **sk** of secret keys to be used by the members. If, however, the group of members is dynamic, KeyGen does not output secret keys for the members. Instead, the Join protocol can be used to let non-members join the group. As the end of this protocol, a new member obtains a secret key sk_i , while the group manager obtains some information Y_i related to the new member that he includes into his secret key **gmsk**. To sign a message **m** the member runs **ÏČ** ← Sign(sk_i , **m**). To verify a signature **ÏČ** on message **m** one computes Verify(**vk**, **m**, **ÏČ**). Furthermore, given a signature **ÏČ** on **m**, the group manager can identify the originating member by computing Open(**gmsk**, **m**, **ÏČ**), which outputs the identity of the member who created the signature. Finally, using the Revoke algorithm (**vk**, **gmsk**) ← Revoke(**gmsk**, Y_i), the group manager can exclude the member relating to Y_i from the group.*

Since the original schemes proposed by Chaum many schemes have been proposed, including [24] and [36]. The latter is a short group signature scheme, which makes use of bilinear maps to reduce the signature size. The former was extended in work due to Camenisch and Groth [42]. This scheme includes the ability to efficiently revoke, as defined above, a group member's ability to sign messages, but does not make their old signatures invalid. The revocation process requires a small update to the group's public key and each group member must perform a local computation. The work on group signature schemes has been applied to creating the Direct Anonymous Attestation protocol used for authentication of Trusted Platform Modules in version 1.2 of the TPM specifications [40]. This specification is implemented in, for instance, Intel Core 2 chipsets [2].

2.8 Trust Management

As defined in [34] Trust Management is the field that studies "security policies, security credentials, and trust relationships". Work due to Blaze et al. on the PolicyMaker system introduced the area. According to RFC2704, which details the Keynote trust management system, [33] a trust management system has 5 components:

1. "A language for describing 'actions', which are operations with security consequences that are to be controlled by the system."
2. "A mechanism for identifying 'principals', which are entities that can be authorized to perform actions."
3. "A language for specifying application 'policies', which govern the actions that principals are authorized to perform."
4. "A language for specifying 'credentials', which allow principals to delegate authorization to other principals."
5. "A 'compliance checker', which provides a service to applications for determining how an action requested by principals should be handled, given a policy and a set of credentials."

All applications that use a trust management system can ask the compliance checker whether a specific action is allowed under the configured policies. Since every application that uses a trust management system shares a common language for describing policies, these policies can be managed centrally and distributed across networks.

Recently, the scope of the trust management field has broadened to include related topics such as trust metrics, trust establishment, and automated trust negotiation systems. Trust metrics aim to quantify trust. Marsh [88] has undertaken work in formalising the concept of trust to allow agents to make trust-based decisions. In reputation systems such as eBay's [17], feedback scores provide quantifiable measures of trust of actors based on their past interactions. Bhargav-Spantzal et al [31] describe a system for managing and sharing transaction histories, that may be used to establish trust between actors based on their certified interactions with others in the past. Work on Automated Trust Negotiation, such as [121], aims to allow the requirements for trust to be established between two actors, while avoiding the disclosure of unnecessary and sensitive information.

2.9 Conclusion

This chapter introduced a number of systems related to both naming and identity management. Each section introduced a different area, and discussed how the systems in that area related to the problems posed by personal naming and identity management. Additionally we gave an overview of relevant systems that we will make use of in our solution, or which informed the design somewhat.

3. SOBRIQUET DESIGN

In the first chapter we set out the problem description and the requirements for its solution. The second chapter covered related systems, and some systems which we will make use of in our solution. We will now discuss briefly the high-level design and operation of Sobriquet and argue that it meets the requirements we set out in the first chapter.

This chapter is divided into a number of sections. The first section will reiterate the requirements for our solution as set out in Section 1.3. We will then give a brief overview of the operation of the system in the second section. We have divided our discussion of the design into six main sections, those that discuss: Sobriquet names, security protocols, identifier exchange, the Data Store, name resolution, and trust establishment. The Sobriquet names section will talk about the different identifiers in use in Sobriquet, their function, and how they are managed. The security protocols section will detail our use of cryptography, while the identifier exchange section discusses the use of some of the outlined protocols. We will then talk about the Data Store and its function, including its application to name resolution, which we will discuss in the seventh section. The final piece of the design is the trust establishment section which discusses a novel means of bootstrapping an association between people who do not know one another. We will then go on to analyse our solution before concluding the chapter.

3.1 Requirements

In chapter one we outlined a number of requirements for our solution. These were based on a discussion of a use case, and are as follows:

1. Provide a way to securely identify people

2. No Trusted Third Parties for name allocation

3. Application Specific Identifier portability

4. Do not compromise the privacy of users in the way they identify themselves

5. Provide a means to establish secure communications between people who have never met

We will now discuss these requirements in the context of specific problems.

3.1.1 Identifier Allocation

The most important criterion for our identifiers is that they are completely independent of any identity provider. In essence what we want is to allow each person to be their own identity provider with full control over the life cycle of an identifier. The system should automate the identifier management process.

No Reliance on Certification of Identity by Third Parties

In the case of federated identity management systems it is more appropriate to rely on centralised identity providers to assert the identity of its users given that they must allow sharing of information about the identity of members of institutions with other members of the federation. Since each authenticate their own members for services internally anyway, it is logical to allow each institution to act as identity provider for each of their members when they roam to another institution. Identities within such a system exist as long as these institutions exist and so long as it is appropriate to continue the federation.

However for the case where an identity provider's role is mainly to provide an identifier and facilitate authentication, such as in systems like OpenID, and when that identifier should be long-lived, then relying on identity providers becomes problematic. Identity providers introduce two significant problems that are avoidable in the design of an identity management system. The first problem is that customers rely on those providers to access resources of value. In doing this they must trust the identity provider to provide them with this access. In certain systems, such as OpenID, the Identity Provider can grant themselves access to their customers' resources. Furthermore, if the identity provider goes out of business then their customers lose the ability to login to services where they previously relied on that identity provider for authentication. This is not without precedent, in the case of Vox, an OpenID provider, where its parent company shut down the service after its acquisition [70].

Also, these systems typically do not provide a way to port identifiers from one provider to another. In their report on deregulation of the telephone industry, the FCC noted that "Number portability is one of the obligations that Congress imposed on all local exchange carriers, both incumbents and new entrants, in order to promote the pro-competitive, deregulatory markets it envisioned" [48]. We assume that identifier portability on the Internet would encourage similar competition.

While OpenID does support identifier portability through a user discovery mechanism, this requires each person to own their own domain and host a file that specifies the identity provider they use. Domain names cost money and achieving identifier portability this way would require the configuration of software in a manner that is arguably too complicated for a large number of users. This seems like a complex solution and it is not clear whether most users would be prepared to use this feature. While it would be possible for individual applications and service providers to allow people to change the identifier they use as a configuration option, if this is enforced by design then this functionality will have to be provided.

One of our requirements is that we cannot rely on third parties for the allocation of identifiers. Our choice of public keys as the primary identifiers in Sobriquet is due to this design requirement as they are globally unique[1], can be used for authentication, and may be created as needed. By providing the user with full control over management of their identifiers we aim to avoid the ill-effects of the identity silo approach of systems such as OpenID.

Trusted third parties, such as the CA in the X.509 standard for PKIs, bring complexity and overhead in the management of the system. The operation of a PKI is always going to require a trade off between the quality of certification that may be performed and the price of certificates. There is no guarantee that a CA that issues certificates will maintain their infrastructure to an acceptable level in the face of new threats. Trusted third parties are an attractive entity to attack. If an attacker manages to compromise one, as has already happened [109] in the case of the Internet PKI infrastructure, then they may produce a valid certificate for any domain they want and so long as endpoints trust that CA they will accept those certificates as being valid. If attackers can create their own certificates then there is no way to distinguish which certificates ought to be trusted from those that were maliciously created. This is arguably worse than having no certification at all as client software would show no indication of the compromise and users would assume their connections were secure.

However, exchanging public keys without having a Trusted Third Party (TTP) present means that there must be some other way to verify the exchange has taken place correctly. We will outline a few possible solutions for this in 3.5. The shortcomings of TTPs listed above does not however negate their usefulness. However, for the purposes of Sobriquet we wish to avoid having them involved in the process of certifying a person's public identity to a long term public key to reduce the complexity of the system and avoid having a person's public identity certified to an identifier that may

[1] To a very high probability, assuming they are chosen using large random numbers

be used to uniquely identify them. A compromise, however, is to make use of channels that themselves make use of public keys that are certified by TTPs in order to protect the confidentiality and integrity of the exchange. In the case that the TTP is secure and has not been compromised this brings additional security to the exchange. In the case that the TTP has been compromised, then this is no worse than the leap of faith taken in the exchange of public keys that may be used by SSH in the absence of a PKI, and that we are willing to accept in the exchange of public keys in Sobriquet.

Unique and Unambiguous Identifiers

People are not easy to identify in a systematic way in the real world. Surname and first name tuples are not unique in a large number of cases. Incorporating other data about a person into their identifier is problematic because either that information doesn't provide a unique identifier or it can change, which means the identifier lacks consistency, or the resulting identifier could leak private information.

As noted in the SPKI literature [61] the idea of a global infrastructure of names that would be suitable for people is not likely to materialise. The approach advocated in the X.500 standards is to bind a public key to personal information through a process known as certification. People can then use this information to find a person's public key in a global directory. However certifying such information about a person is expensive as it would require obtaining proof of identity from each and every participant. Also, it is not desirable that people's personal information be made publicly available.

A person's view of another's identity can change over time and people's relationship to one another can change over time. Alice might, for instance, meet Bob at a conference. In the beginning they would regard each other as acquaintances. This has practical implications for how each might wish to be contacted by the other. For instance Alice may not want Bob to be able to contact her at home, and would like to only communicate with him over email. Over time however perhaps they become collaborators. Now Alice and Bob may wish to allow each other to share drafts of a paper so they need an additional way to communicate. For this reason local names are most appropriate. These local names can change over time, if necessary, to reflect the view of their relationship that both Alice and Bob have.

As introduced in the previous chapter, the petname model represents a compromise that allows a naming system to be decentralised with securely unique identifiers, while having memorable names assigned to its participants. Allowing people to assign names of their choosing allows for a namespace of names with high mnemonic value. If collisions in the namespace occur the user can resolve them by themselves. For instance if Alice knows two Bobs she will probably have a way of naming each of them that differentiates between the two. Finally, using public keys as names allows identifiers to be globally unique and provides the basis for an authentication infrastructure.

Usability

Having a strong notion of identity incorporated into the Internet infrastructure would result in a number of benefits from the point of view of usability. First of all people would be more easily able to filter and manage their online interactions. They could, for example, route emails according to the identity of the sender, or redirect VoIP calls by caller. As spam spreads to other applications, such as IM and VoIP, the ability to securely identify people we communicate with will become all the more important.

Sobriquet has the potential to integrate with existing applications as an authentication solution. This would reduce the complexity in signing up for accounts with different applications. This would be a boon for Internet application providers as it would lower the barrier to entry for users of their application by simplifying the registration process considerably. At the same time Sobriquet would reduce the complexity in managing different accounts across different providers. To make these potential benefits a reality Sobriquet should manage the creation and management of identifiers.

3.1.2 Name Resolution

Name resolution is the function of a naming system whereby a name is mapped to its corresponding result set. Resolution is always performed by the user and returns ASIs, which may be used by Internet applications on their devices. Frequently the result set contains the information required to establish a communications path. Since this is the primary function of the system it is important that the resolution mechanism does not introduce much latency and the database the results are stored in is resilient to failure.

Low Overhead

The first priority for our resolution architecture is that it should not, when possible, introduce much overhead into the establishment of communications sessions. To this end it should attempt to eliminate sources of latency while keeping the resources it uses to a minimum. In the last chapter we saw how latency in systems such as DHTs increases substantially as the number of hops increase. The arguments of the creators of the Main Name System for the "recentralisation" of DNS show how centralising the resolution infrastructure can lead to lower total overhead from both a latency and a management point of view [53]. We take this approach in Sobriquet, while giving the user control over the information stored in the centralised Data Store, making it easy to move between different Data Stores or to use more than one. Furthermore, we anticipate Sobriquet result sets being quite small as a direct result of a user's social network being quite small. We can take advantage of these facts to design a caching algorithm that results in low overhead in name resolution.

Resilience to Failure

Sobriquet aims to be resilient to failures of individual name servers in the system. Configuration options for many DNS servers can be quite complex, which can lead to failures in name resolution. Compounding the problem in DNS is implicit dependencies on other servers for resolution, thereby creating additional points of failure. Sobriquet aims for simplicity in this respect. Servers should be simple to install and there should be few configuration options required from users of the software.

Intelligent Caching

Related to the previous two requirements is the need for an intelligent caching algorithm. Sobriquet can take advantage of the fact that the people who will be resolving identifiers are known to each other ahead of time, since a mutual exchange of identifiers must take place in order for them to be able to perform resolution in the first place. This means that we can push updates to the result set to those who will need them. Our caching algorithm should reduce the resolution overhead, and play a large part in mitigating the effects felt by the failure of individual name servers.

3.1.3 Authentication

Privacy Preserving

The goal for our authentication protocols is to ensure that they do not allow tracking of users under a single global identifier. Central to this is the fact that Sobriquet identifiers should not be tied to a real world identity unless the user chooses for this to be so themselves. As such our goal is not total anonymity, but rather to prevent others from proving that messages came from a certain source, while ensuring that those messages are adequately authenticated. We aim to use the privacy enhancing technologies described in Section 2.7 in this effort.

Application Independent

We have identified a number of different high level use cases. We intend to design protocols to fit these rather than design for each application directly. Our intention is that the protocols will be self-contained and independent of the underlying application that uses them. To this end the design of the authentication protocols follows two abstract models. The first is a security protocol for store and forward systems, such as email. This model assumes that there may be significant delays between when messages are sent, when they are delivered, and when they are replied to. The protocol for this type of system is limited to a single message. The second model is that of an interactive session. Here both parties are present and exchange messages in real time. This is the model for most types of interactive communication, such as VoIP. Here the authentication protocol is not limited to a single message and both parties are present. We believe these two high-level models allow our protocols to be used within the majority of internet communications systems. We will describe the messages involved in these protocols in 3.4.

3.1.4 Trust Establishment

The trust establishment component attempts to provide a means for two people, Alice and Bob say, to establish an authenticated session without requiring that they have an existing relationship or any means of meeting to exchange identifiers. Alice and Bob have never met, and so neither Alice nor Bob have any way of knowing who the other is. Furthermore, it is possible that one wishes to keep their identity hidden from the other. Still, it is desirable that neither party can invent identifiers for themselves without incurring some real world cost. Authentication in this instance would not be to an identity, but to something that provides both a level of assurance that the other person is limited in their capacity to perform malicious actions, such as sending spam. The design of our trust establishment component is motivated by a number of observations:

1. For certain types of Internet based malicious behaviour to be economical, it needs to occur in high volume. A good example of this type of behaviour is spam,

where spammers need to send large volumes of messages to turn a profit. This observation is supported by investigations into spammer activity such as [78].

2. Proof of Work schemes such as Hashcash [26] aim to deter abuse of communications systems by forcing the initiator of a communications session, e.g. the sender of an email, to perform a certain amount of work that is computationally expensive enough to limit the rate the sender's system can send email at. The recipient of the email can easily verify that the required computation has been performed without expending too many resources. A major problem with this approach is the fact that spam may be sent from botnets where many thousands or millions of machines are deployed, meaning that even though the throughput of each machine could be slowed down in sending each email, since the attacker isn't paying for the CPU time on the hijacked machines they don't care as much about the computational cost of sending individual emails.

3. If we can tie malicious actions that occur online instead to a real world task that takes up a person's time then we can make it expensive for them to undertake high volume transactions maliciously. If the number of malicious acts they can engage in is sufficiently slowed down then it becomes uneconomical to partake in them. This is similar to the motivation behind introducing delays between unsuccessful logins to systems as is common with, for example, remote shell software. However, our approach would seek not to waste a user's time but rather to acknowledge that their time has been spent productively and that they are likely to be a real person.

While a person's identity is protected by this component they are authenticated as being an individual as attested to by a mutually trusted third party. This affirmation is not done through a traditional certification process, i.e. it is not tied to a specific identity, but rather to a set of actions that themselves take a sufficient length of time to undertake and need to be performed by humans. We intend for these to be actions that an individual would undertake anyway. For instance, the operator of an online forum knows that a specific user has been active on their forums for a given length of time. They know that the person has contributed to discussions on the site, and that those actions take time. However since the user of a site has an interest in contributing to it this is time they would invest anyway. So to a legitimate user this investment of time is not seen as an inconvenience. However, to someone whose only goal is to gain access to a Sobriquet Reference[2] performing these actions often take far more time than the reference is worth for them to obtain. Admittedly though, this approach may not prevent spear phishing attacks, where attackers target so called high value individuals, such as celebrities, given that the reward for compromising their communications has significant value. Even so, compromise of a certain Sobriquet reference does not guarantee a sufficient reward in and of itself.

3.2 Sobriquet Overview

In the first chapter we demonstrated a need for our system by taking the example of someone using email, and showing the issues that relate to identity and the use of ASIs to identify people within that system. We will now give an overview of the operation of Sobriquet at a high level. While we have not yet introduced the various components, presenting an overview of the operation of the system first will make the role of the various components clearer as they are introduced.

In the use case for Sobriquet, Alice would give out her email address to friends. They would then use this to contact her. This step has changed. Now Alice provides her Global Name and the URI of a Data Store, where her contacts may obtain an authenticated list of Application Specific Identifiers she uses, to her friends via the Secure Identifier Exchange protocol. This may involve obtaining a reference if the person Alice is exchanging identifiers with is not someone she has contacted before.

[2] We use the term reference because Sobriquet References have a similar goal to real world references as provided by employers.

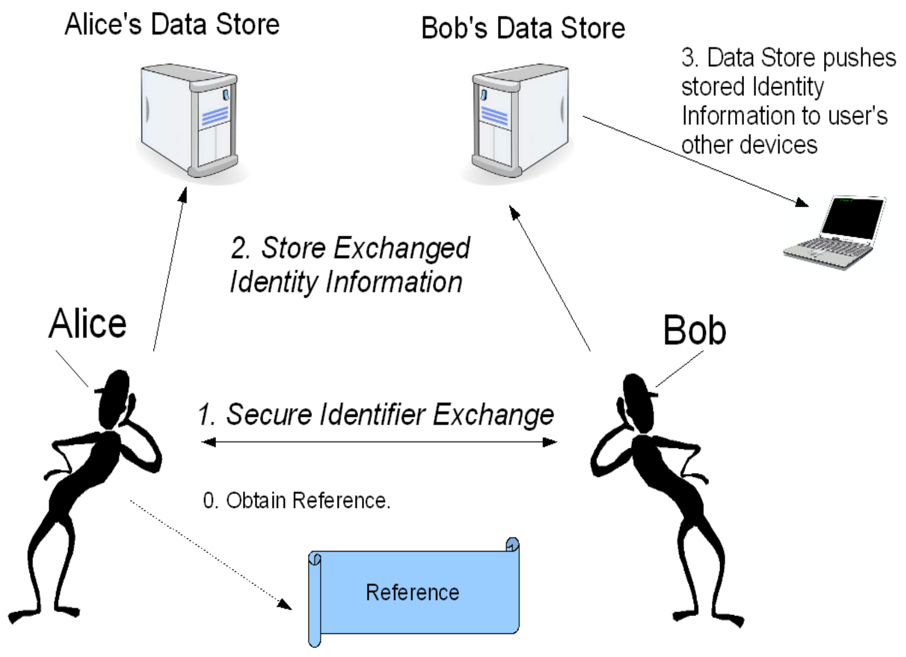
Alice's Data Store
Bob's Data Store
3. Data Store pushes stored Identity Information to user's other devices
2. Store Exchanged Identity Information
Alice
Bob
1. Secure Identifier Exchange
0. Obtain Reference.
Reference

Fig. 3.1: The Secure Identifier Exchange and interaction with the Data Store.

Her friends can perform a lookup on Alice's global name to obtain, among other ASIs, her email address. This step is shown in Fig 3.1. We see that Alice stores information she exchanges with others in a Data Store. This is a central point that all her devices can use to access information needed to resolve global names to ASIs and authenticate people she communicates with.

When Alice wants to send an email to someone she has exchanged Sobriquet identifiers with, she performs a lookup on their Sobriquet Global Name, which identifies them uniquely. In Fig 3.2 we see that she first tries a local cache, and if that look up fails she then queries her Data Store. Sobriquet makes use of client side caching to reduce the performance impact of performing a lookup to the Data Store.

When Alice wants to change one of her ASIs she pushes updates to her Data Store. Since she has a local cache on each device, her local caches ensure they are synchronised to what is most up to date in the Data Store. Any updates to the local caches are pushed to the Data Store, which is how updates to the namespace on any one of Alice's

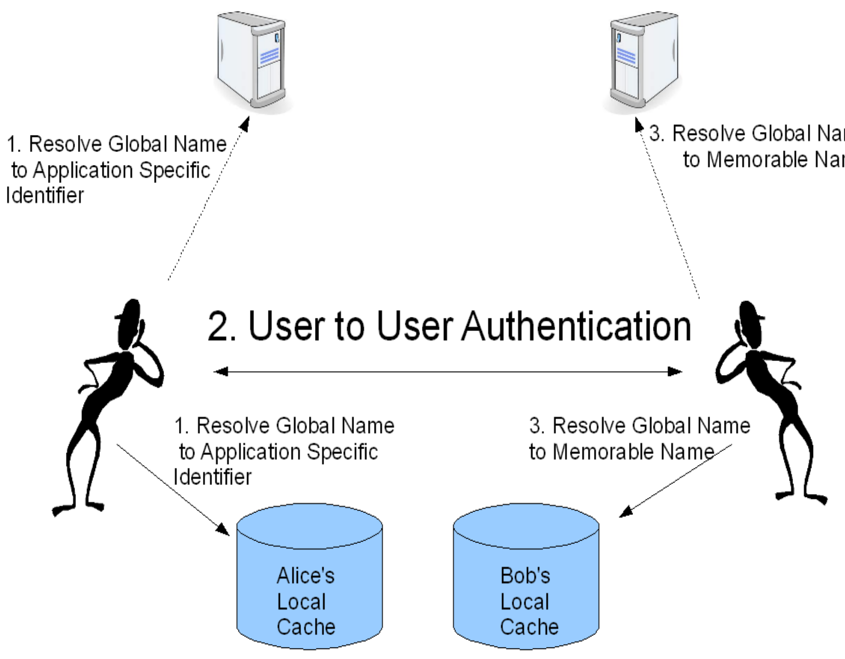
1. Resolve Global Name to Application Specific Identifier
3. Resolve Global Name to Memorable Name
2. User to User Authentication
1. Resolve Global Name to Application Specific Identifier
3. Resolve Global Name to Memorable Name
Alice's Local Cache
Bob's Local Cache

Fig. 3.2: Name resolution and authentication.

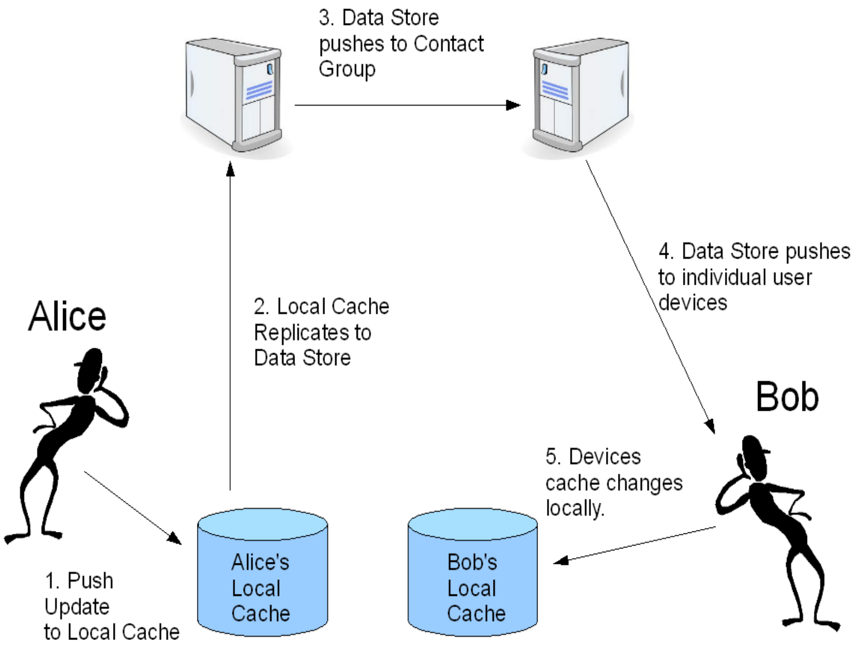
3. Data Store pushes to Contact Group
4. Data Store pushes to individual user devices
Alice
2. Local Cache Replicates to Data Store
Bob
5. Devices cache changes locally.
1. Push Update to Local Cache
Alice's Local Cache
Bob's Local Cache

Fig. 3.3: Updating Sobriquet Namespace.

devices propagates to the rest of them.

3.3 Sobriquet Names

Our naming system concerns itself with two types of name; the global name and the petname, which we call the Sobriquet Global Name (SGN) and Sobriquet Memorable Name (SMN) respectively. Sobriquet makes use of a group signature scheme, as introduced in Section 2.7.2. The SGN is a public verification key created under this scheme. In the group signature scheme we use, due to Camenisch and Groth [42], public keys are large randomly selected numbers, which are by their nature difficult to remember. We chose to use a cryptographic namespace because of our requirement that the solution should not rely on an identity provider.

To create an SGN a Sobriquet user will run the KeyGen function of the group signature scheme; the role of this function in the group signature scheme is briefly described in Section 2.7.2. As noted there this will result in two quantities, a public verification key and a group manager secret key. The public verification key is used as the SGN. From this point on we will refer to the public verification key as the SGN. The Sobriquet user will additionally store the group manager secret key as she will need this to admit people to her contact groups at a later date. This means that there is a one to one correspondence between a Contact Group and an SGN, and so an SGN identifies a particular Contact Group.

There are two ways in which an SGN is used. The first is during resolution. Here Alice, say, will take an SGN belonging to Bob, and perform a name resolution on it to obtain a list of Application Specific Identifiers that Bob has associated with that identifier. The exact nature of the resolution function will be discussed in Section 3.6. The second way an SGN is used is in the security protocols outlined in Section 3.4. Here an SGN, as a public verification key, will verify that a given person belongs to a certain contact group after they prove themselves to be the holder of a group member private key, that is a quantity under the chosen group signature scheme that is used to prove membership of a particular group.

The choice of SMN is left up to the user. This is a memorable string that is assigned to, and has a one to one correspondence with, an SGN. For instance in Fig. 3.4 we see that in the set of Richard's names there is an entry for a Memorable Name "Alice" that maps to the SGN beginning with "7e48c3a696e4", whereas in the set of Harry's names that same Global Name is assigned the Sobriquet Memorable Name "Alice from ICNP".

The SMN is used in the same two Sobriquet functions as the SGN. The first is during name resolution. Here Alice, on deciding that she wishes to contact Bob, enters the SMN into whatever Internet application she is using. The SMN is used in the same way as a DNS domain name in Internet applications; for example in a web browser a domain name is resolved to an IP address and the browser assumes there is an endpoint at that IP address that can speak HTTP. In the same way, when a user types an SMN into a Sobriquet aware application, the application will perform a resolution on it using the Sobriquet client software. The client software can map an SMN to an SGN and perform a resolution on that SGN to obtain Bob's ASIs, which will ultimately be used by the Internet application Alice is using to contact him.

As discussed in Section 3.5, identifier exchange will take place over whatever channels are available, and will inherit the security properties of that channel. The method of exchange used will impact each person's notion of the other's identity and this will be likely reflected in their choice of SMN. For this reason we refer to Sobriquet identifiers as being "history-based". Since the weakest form of exchange takes place over a totally insecure channel the participants in the exchange cannot assume the exchange succeeded. However over time that exchanged identifier will come to be associated with the history of interactions the two participants undertake together. If later on they choose to verify their identifiers, for instance using voice or video in conjunction with a secure hash of the identifier, then they will have a decent level of assurance of each

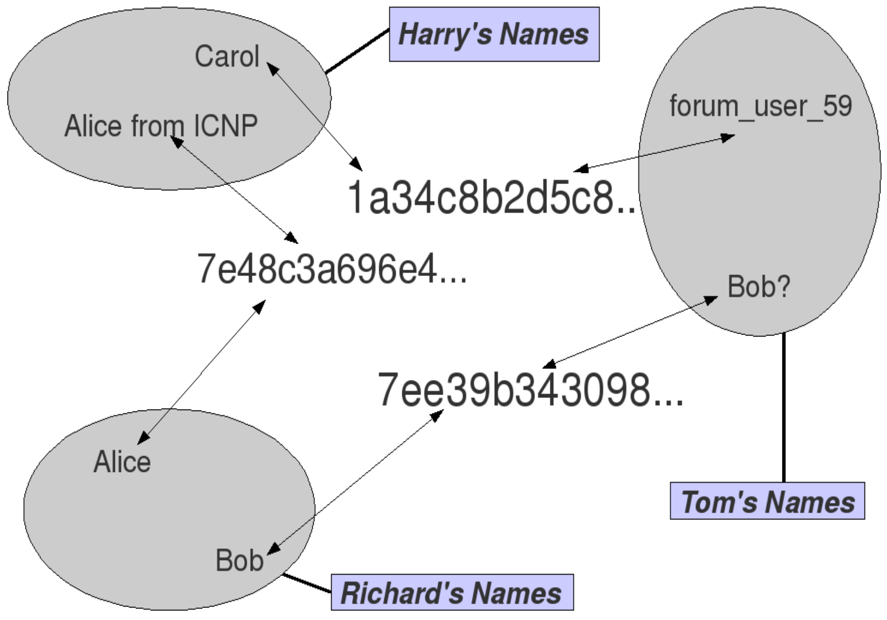

Fig. 3.4: Relationship between Memorable Names and Global Names.

others' identities. In addition to verification techniques we advocate involving service providers in the identifier exchange.

3.4 Security Protocols

In this section we describe the security protocols in use in Sobriquet. We use existing security protocols, such as the Station-to-Station protocol [57]. Here we will outline them as they are integrated with the group signature scheme we use. In subsequent sections we will refer back to these protocols in our descriptions of the various components that use them.

We consider the following three scenarios, where authentication is required in Sobriquet:

- User to User interactive - This protocol is used between two people who wish to communicate interactively, during communications session establishment.

- User to Server interactive - This is used in authentication to the Data Store and also between users and hosted Internet applications that use Sobriquet.
- Non-interactive - Used in store and forward situations. This is a single message, and is used by the identifier exchange protocol.

3.4.1 Interactive Protocols

Both of these protocols are based on the Station-to-Station (STS) protocol [57]. Like STS they provide perfect forward secrecy, mutual authentication, and result in a shared key being established. During Sobriquet identifier exchange, which we will outline in 3.5, each party, acting as group manager, executes the join function themselves and generates all the required parameters. This means that the group manager can sign messages on behalf of any user in the group, which destroys the "no framing" guarantee of the scheme. However since the group signatures are only meant to be used in authentication protocols with the group manager, who in our case is a Sobriquet user, it doesn't matter if the group manager can forge signatures for members of their Contact Group.

In fact this brings an additional property of deniability to the protocols. Since any message could potentially have been signed by two people: the group manager and the member, it is not possible for the group manager to prove to a third party that any message was signed by a particular contact of theirs since the group manager could equally have signed it. That means that if Alice and Bob are contacts and communicate using Sobriquet there is no way for Bob or a third party to enforce or claim the property of non-repudiation on a signature from Alice, since Bob can produce signatures on her behalf. However, since he can only produce signatures that can be verified by the group public verification key that he is using as his SGN, only messages ever destined for him could be forged by him. In normal operation of Sobriquet Bob will never produce these signatures and so this property does not introduce a reflection attack.

In other words, since Alice knows the private key that Bob will be using to authenticate himself to her, she can sign messages on his behalf. Since any message with Bob's signature could have been signed by Alice there is no way for Alice to prove to a third party that Bob signed it and not her. However since Alice knows which messages she has signed she can still authenticate Bob.

Protocol 1 - User to User

Before we outline our protocols we introduce some notation that we will use throughout this section. We refer also to definitions outlined in 2.7.2. Here the following symbols are ascribed meanings as follows:

- H - is a hash function.
- $|$ denotes concatenation
- $S_{AliceBob}$ denotes a group signature
- p is a prime, and g is a generator for the cyclic group p. These are the Diffie-Hellman parameters referred to in [57], and as the basic protocol here is STS, these and other relevant system parameters should be chosen as such.
- x is a random number
- $S_{AliceBob}(t)$ denotes a signature produced on t by Bob using his group member private key for the group public verification key of Alice.
- $Cert_{Owner}$ indicates a public key certificate that has been issued to the Owner.

We assume that the choice of H, g and p has been made ahead of time. The protocol follows these steps:

- Alice sends: $g^x \mod p$, $H(SGN_{Bob})$.

- Bob sends: $g^y \mod p, E_K(S_{AliceBob}(g^y \mod p, g^x \mod p))$
- Alice sends: $E_K(S_{BobAlice}(g^x \mod p, g^y \mod p))$
- Bob opens the signature produced by $S_{BobAlice}$ and verifies that the member ID returned matches Alice's.
- Alice opens the signature produced by $S_{AliceBob}$ and verifies that the member ID returned matches Bob's.

A group signature indicates that a person has obtained membership in a given group. We indicate this in the preceding authentication flow as follows $S_{ManagerMember}$. So for example the signature $S_{BobAlice}$ is produced by Alice using the group member private key that she received from Bob. Since a person may operate different groups in the first step, Alice sends the hash of the SGN she wants to authenticate to and uses $g^x \mod p$ as a salt. Since the secure hash of this value may be computed quickly it should not add much overhead assuming Bob has a limited number of public keys to check this hash against. Since each public key corresponds to a contact group he manages, this is a reasonable assumption. At the end of execution of this protocol both parties end up with a shared key $K = g^{x^y} = g^{y^x} = g^{xy}$. The shared key is then put through a key derivation function to obtain a symmetric encryption key that each will use to preserve the confidentiality of the channel.

Protocol 2 - User to Server

- Alice sends: $g^x \mod p$.
- The server sends: $g^y \mod p, Cert_{Server}, E_K(S_{Server}(g^y \mod p, g^x \mod p))$
- Alice sends: $E_K(S_{ServerAlice}(g^x \mod p, g^y \mod p))$
- The server opens the signature produced by $S_{BobAlice}$ and verifies that the member ID returned matches Alice's.

In this variation the server sends a digital certificate that attests to its public key. Alice can verify this information in the usual way and will verify the signature produced by the server S_{Server} as a regular public key signature.

3.4.2 Non-interactive Protocol

For non-interactive protocols Alice can simply sign messages. Her contact can then open the signature and obtain her identity. The non-interactive protocol can be used to send public encryption keys to contacts to establish confidential communications. We assume that Alice has a public encryption key, and a public key certificate C. Alice generates a signature $SSC = S_{BobAlice}(C)$. If Alice's certificate C is signed by a mutually trusted CA she can use that, or she can simply use a self-signed certificate. Her signature on C ensures the certificate will be accepted and trusted by Bob in either case. Assuming Bob has sent her a message of the form containing his public encryption key (PK_{Bob}) ahead of time she can then sign and encrypt messages to him as follows:

- Alice chooses a random session key K.
- She encrypts the message intended for Bob with K, $EM = E_K(M)$
- She uses Bob's public encryption key to encrypt the session key $E_{PK_{Bob}}(K)$ and prepends this to EM.
- Using her group member private key in Bob's namespace she signs the message $E_{PK_{Bob}}(K)|EM$.

When Bob receives the message he will first open the signature to obtain Alice's identity and map this to her SGN, and the public encryption key he has sent her. He will then decrypt the session key using his private decryption key.

3.5 Identifier Exchange

At this point we assume that each of the participants, Alice and Bob, have SGNs created as described in 3.3. Each has decided that they would like to make their SGN available to the other so that each can use Sobriquet identifiers to refer to the other. We assume the existence of a secure channel over which the SGN exchange will take place. While Sobriquet doesn't rely on the presence of any existing authentication infrastructure or secure channels, it should make use of them for SGN exchange if they exist. This is similar to how SSH makes use of a PKI to ensure that public keys are exchanged securely, but in the absence of the PKI falls back to exchanging them over a less secure channel and in doing so accepts the risk that there may be an active attacker present. What follows are recommendations for the types of channel to use for SGN exchange. These are not indicated in any order of preference:

1. Exchange the SGNs using an existing secure channel provided by a Sobriquet aware application. An example here would be to tunnel the exchange protocol over SSH between Alice and Bob. Since it's impossible in general for the Sobriquet client software to know what the security properties of a channel it's using are, it's up to a Sobriquet aware application to make the decision about whether the channel has the desired properties.

2. Alternatively the confidentiality of the channel may be augmented by the use of a location limited channel. An example of this would be the use of Near Field Communication (NFC) [13], which allows short range contact between two users. The use of such a channel does not guarantee the absence of an attacker, but greatly reduces the likelihood of one being present for the exchange.

3. An insecure key exchange can be made more secure through out of band verification of the exchanged quantities. Here, for instance, each user could ring the other up and verify the hash of a key established using a key exchange protocol.

If each recognises the voice of the other and are satisfied that the hashes match then they can assume the exchange took place securely.

4. Use an insecure key exchange without out of band verification. This assumes that there are no active attackers, which is the type of assumption that makes more sense in some protocols than in others.

The risk from compromise of the SGN exchange is that the attacker may be able to send messages, for example, from Alice to Bob authenticated as her and vice versa. If either party detects that this has happened they can revoke the exchanged identifiers and retry the exchange at a later date. They may notice that this has happened, when for example, they go to authenticate each other at a later date and find that the SGNs each are trying to authenticate to don't match any existing ones. If an active attacker manages to compromise the exchange then they must be present on every channel that both Alice and Bob communicate over from that point onwards, otherwise they will notice that each others SGNs have changed.

Each participant, Alice and Bob, run the group signature scheme's Join protocol, as outlined in 2.7, in turn and store the output. They also each choose an SMN and send each other their Data Store URI. By running this protocol each end up with the following quantities:

1. **Sobriquet Global Name** - Alice and Bob will obtain an SGN for each other. When the protocol is run interactively they will use this to authenticate each other for the exchange of private information.

2. **Sobriquet Memorable Name** - assigned by each person individually. They have a one-to-one mapping to global names and serve only as a memorable identifier for non-memorable global names. As such they are chosen by each person and ought to reflect the relationship that they have with the owner of the corresponding SGN, for example "John from Work" or "The Boss".

3. **Group Member Private Key** - This allows a participant to authenticate themselves as a member of a particular Contact Group. In practice this is the group member private key created by execution of the join protocol of the underlying group signature scheme[3]. In such a scheme proof of ownership of a group member private key allows a person to be authenticated as a member of the group and allows the group manager to open any signature signed on behalf of its group, and obtain a unique identifier for that group member.

   This introduces a level of directionality into the authentication protocols. If Alice provides Bob with group membership then he may use this to authenticate himself as Bob to Alice alone. Since Bob does not authenticate himself to any global identifier the authentication procedure cannot compromise his real world identity to any eavesdroppers on the channel the authentication takes place over.

4. **Data Store URI** - Alice will notify Bob of the URI of the data store she uses and Bob will let her know of his. This URI will be used during the resolution process. We will discuss this further in the next section.

As we have stated already, our system does not rely on the certification of identifiers to real world identities. Instead we use the trade off between a lower amount of assurance of a person's identity, and attempt to mitigate the risk that the initial exchange of identifiers may be compromised through other means, such as voice verification or location limited channels. If such verification occurs then both Alice and Bob will have a decent level of assurance that the exchange was not compromised. However, if they don't undertake this verification process then the key is effectively exchanged over an insecure channel. The exchange of identifiers itself is broken up into two separate one-message exchanges. The protocol is not interactive because it is assumed that it may be take place over non-interactive channels such as store and forward systems like email.

[3] We discussed Group Signature Schemes in the second chapter.

We further assume that when Sobriquet is deployed in a situation for authenticating a user to a service provider then the service provider will provide the user with group membership, but not vice versa. This is because the identity of the service provider will most likely be public and as such does not need to be protected. The service provider can in this case identify itself using a digital certificate.

3.5.1 Secure Identifier Exchange Protocol

The secure identifier exchange protocol builds on the protocol in 3.4.2. To exchange identifiers Alice and Bob are assumed to have generated public encryption keys. To exchange the necessary values securely, as listed above, they perform the following message exchange.

1. Alice sends Bob her public encryption key in the clear. We discussed the type of channels this should be exchanged over earlier in this section.

2. Bob performs the *join* function of the group signature scheme to generate the necessary group member private key. He then encrypts this and his data store URI as the message in 3.4.2. He signs this using his group manager private key. He sends this message to Alice along with his SGN (group public key), and public encryption key.

3. Alice performs the *join* function of the group signature scheme. She then encrypts Bob's group member private key and her own Data Store URI according to the protocol in 3.4.2. She signs this using her group manager private key and sends the message along with her SGN to Bob.

After the protocol has run each has the required state to contact and authenticate one another.

3.6 Data Store

The Data Store has two main functions. The first function is to store information about a user's namespace, their ASIs, and their contact's ASIs. Secondly the Data Store is a point where a user's contacts can resolve the user's SGN to its result set. This section will primarily deal with the first function, while the second function is discussed in the subsequent section.

The identifier exchange we outlined in 3.5 results in the exchange of Sobriquet Global Names and also the addition of each party to the other's namespace. When two people communicate there will be a step where each participant in the communication session authenticates the other. This authentication step requires knowledge of private key values, which are needed by the user on the machine they are communicating from. Since this machine can be any machine they use, and may change frequently, they need access to this information and their contacts' ASIs from everywhere they communicate from.

This information is divided into two tables in the Data Store's database. The first table is the user's table and this contains the information that is tied to a specific SGN. This information includes the SGN itself, the group manager's private key that allows new members into a Contact Group, a privacy flag, and the result set for that SGN. The group manager's private key and the encryption keys are stored in encrypted form. We will detail how these records are encrypted, and how keys are managed in Section 3.6.1.

Access to this information is accessible via the name resolution mechanism to clients. Access is controlled through the use of the "Private" flag as shown in the diagram. If this flag is set then a user must authenticate themselves using the protocol outlined in Section 3.4.1 in order to gain access to this information. Since the Data Store can verify messages signed by clients using their group member private key, this authentication can take place. However, the Data Store will be unable to open the signature and

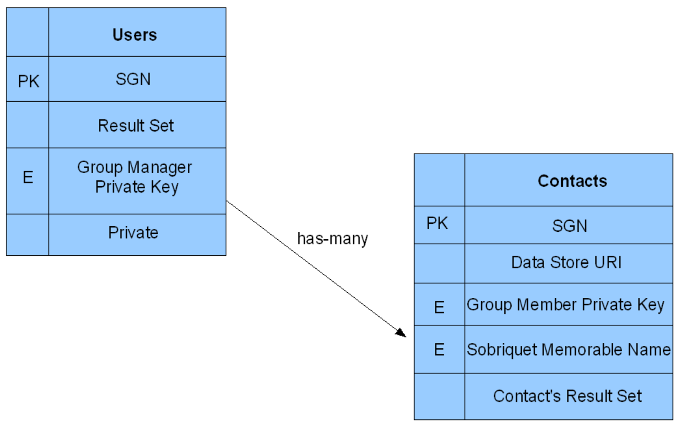

Fig. 3.5: The data stored by the Data Store. PK signifies a primary key, and E signifies an encrypted record.

obtain the identity of this client.

The second database table of significance used by the Data Store is the contacts database. There is a has-many relationship between users and contacts; that is a user has many contacts. This relationship is depicted in Fig. 3.5. The information stored in this table is as follows: the SGN of the contact, their Data Store URI, the group member's private key that allows the user to prove they are a member of the corresponding Contact Group to the SGN, the Sobriquet Memorable Name that has been assigned to the contact, and the contact's result set. Out of this information the tuple of the private key and the memorable name are stored in encrypted form. The contact table is only available to the user, and only once they have sufficiently authenticated themselves as outlined in Section 3.6.1.

On the client side the user will have a Data Store Browser, which is a piece of software that communicates with the Data Store, or multiple Data Stores if the user so wishes, and is responsible for synchronising the state on a device with the most up to date state in each of the Data Stores as indicated by a counter value that increments each time an update takes place and which is stored in the Data Store alongside each

SGN to ASI set mapping. The Data Store Browser will maintain a long-term cache on devices the user owns and will maintain a temporary cache on devices the user does not own. To access the Data Store the user will supply the Data Store Browser with a username and password. The username will be in the form of an email address, which the software will parse into username and host components by splitting the string on the "@" character.

Each Data Store Browser maintains a persistent connection to the Data Store. When this is not possible it may periodically poll. When a connection to the Data Store is opened the Browser will send the last timestamp it received from the Store. If there are any updates it will send a new timestamp along with the updated information. If there are no updates the Store will reply with the timestamp sent. The Data Store will push information to the Browser as updates occur. Since each item stored in the Data Store is indexed according to a SGN, each update will be also. The full new record will be sent with each update and will be overwritten if the timestamp of the received update is greater than the one present in the cache. The Data Store treats encrypted data as a binary string. This string will be encoded according to a suitable encoding scheme, such as Base64, before it is sent between the Store and the Browser.

3.6.1 Authentication and Encrypted Storage

The password the user enters to the Data Store Browser will be put through a key derivation function, such as PBKDF2 [77], from which an authentication token and an encryption key will be derived. Passwords should be chosen to be of sufficient length and randomness. While remembering longer passwords is difficult for most people we believe that this is an acceptable trade off given that it is the one piece of state that the user needs to replicate themselves. The authentication token will be used to authenticate the user at the Data Store and the encryption key will be used to encrypt the records stored there. To obtain these two values the key derivation function will be applied twice. The encryption key will be derived from a value obtained by iterating the key derivation function m times and the authentication token will be derived by iterating n times. So long as $m < n$ the encryption key cannot be derived from the authentication token, assuming the key derivation function uses an adequate one-way function for each iteration. The values of m and n should be defined as global constants and their values are seen as an implementation issue that depends on the key derivation function used. The salt for the key derivation function is stored at the Data Store and obtained during the authentication process. We assume that the authentication procedure takes place over a secure transport protocol such as TLS. We advocate, though do not mandate, that the Secure Remote Password protocol [122] is used for authentication over this channel. The authentication token is used as the SRP password. Even if the TLS channel is compromised the attacker will learn nothing of interest from the login exchange.

3.6.2 Data Store Portability

Since we do not want the Data Store to introduce a component that the user is tied to, we allow them to change to a new Data Store at any time. This process requires that they put a special record in their user table under their SGNs, which points to their new Data Store. This record is signed with their group manager private key. This is performed once the new Data Store tables have been populated with the existing data from one of the user's devices. Since this process requires user intervention there is no need to ensure that multiple devices are not updating the Data Store at the same time.

3.6.3 Scalability

Data Store usage should scale linearly. Entries in the Data Store are indexed and queried by the SGN. That allows for easy sharding of the data along that database column, with no cross shard queries, which would allow for very linear scaling. Additionally the amount of state stored per user is not very large. Since each query ought to be carried out at roughly the same frequency per user it's unlikely that intelligent caching would be possible to fit the most heavily queried data in memory. However, storing a subset of the data, namely the vector clock value and a hash of the SGN would heavily reduce the amount of data needed to fit in memory. A 16 or 32 bit integer value should suffice for the vector clock value and so those values for 1 billion users could easily fit in memory alongside a hash of the corresponding SGN (assuming 160 bits), for a total of 36 bytes, on a single machine. By adding more machines this would scale linearly. Reads and writes could additionally benefit from the addition of solid state storage and the use of database replication. Expecting updates to happen at a frequency of about one per day per user should be an overestimation of the requirements since pushing an update requires user involvement in changing their identifiers or adding a new contact, which it seems unlikely would happen on a daily basis. To support 10^9 users performing one update a day, assuming an even spread, would require an infrastructure capable of handling about 11600 requests per second. This figure should be easy to support with a manageable number of web servers and databases on current hardware.

3.7 Name Resolution

The main function of the resolution component is to map names to their corresponding result sets. To reduce the overhead, the resolution component makes use of proactive caching with Data Stores replicating name to result set mappings. This replication makes use of the fact that each person will likely only interact with a small number of people. The overhead for replication should be quite low as a result.

However, recognising that this replication will not always succeed the component also provides an interface to hosts to access its result set. This functionality is only accessed when a person deems their result set to be out of date. In this way the latency involved in resolution is reduced, while replication failures are handled gracefully.

In Fig. 3.6 we see the basic message flow of the resolution component. Updates flow through the Data Stores from user devices. The use of the Data Store as a means of sharing state between user devices was outlined in the previous section.

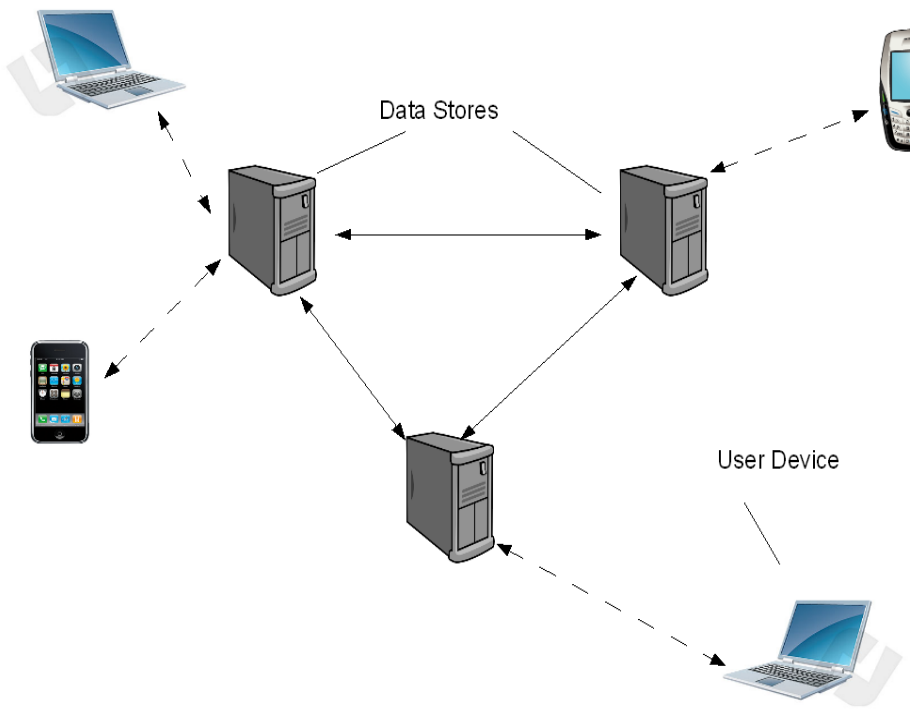

Fig. 3.6: The name resolution message flow

3.7.1 Resolving Identifiers

Name to result set mappings in Sobriquet may require authentication before they can be viewed. This is to prevent harvesting of Application Specific Identifiers through the resolution function. If authentication is required before a given person can query results then this is stored in plain text in the privacy field at the Data Store alongside the SGN and its results. If the user has chosen to require authentication before its result set is provided to a client, then the Data Store must require authentication from the client issuing the query. The authentication protocol used between a client and the

Data Store is outlined in Section 3.4. Additionally the Data Store must not override the preferences regarding requiring authentication from the client.

Each entry in the Data Store is indexed by its SGN. Each SGN maps to a single set of results. While updates are generally pushed from the Data Stores to each of a user's devices, a query may also be performed by a user device to the appropriate Data Store directly. If there is an updated result then the result is synchronised between the user device and its Data Store. This ensures the updates are disseminated to all of the other devices belonging to the user.

We use a vector clock to determine the latest update, with each update tagged with a counter value that is incremented by the client when it is pushing an update to the Data Store. The Data Store can determine if each client has an up to date result set at each point. In an update the client pushes a complete result set, as opposed to just the changes, and so the Data Store can just overwrite one if it is newer than the one it currently has. If the Data Store sees two clients try to push different updates with the same counter value then it can simply accept the first one, and reject the second, forcing that client to retrieve an update and retry pushing its changes later.

The Data Store is responsible for pushing updates to result sets to each of the user's contacts. Since these are signed using the group manager's private key corresponding to the SGN the result set is tied to, we can trust the Data Store to do this on behalf of the user. Since only the owner of an SGN can produce signatures for that SGN, only they may push signed updates for that SGN. This signature is verified by each SGN before the updated results are written to the relevant tables.

When an update is pushed from the client, the Data Store can perform a lookup in its database to obtain the list of contacts associated with the SGN that the user's result set has changed for. This is the list of clients that need to be sent the updated result set. The Data Store will contact the Data Store of each member of the Contact Group corresponding to the updated SGN namespace and push the new result set to them. In doing this it will send the contact's Data Store the SGN of the contact so their Data Store knows which user table to lookup, and the updated result set. The Data Store can then update the contacts table for the relevant users in its database. If multiple members of a Contact Group use the same Data Store then this request can be batched by sending multiple target SGNs in the update. The Data Store receiving the pushed update will then update each of those tables.

There are two scenarios which can result in a result set being queried from the Data Store as opposed to the local cache. The first is when the cached copy on the user's device becomes stale. This occurs after a predetermined amount of time or if the cache is invalidated by the user. In this case a query occurs in order to freshen the cached copy on the device with the most up to date version at the Data Store.

The second scenario is when the user attempts to initiate a communication session with one of their contacts and a failure occurs at the authentication step. The user initiating the session will then check their Data Store for an update from the person they are attempting to contact. If they determine that there is none then authentication is deemed to have failed. It is up to the initiator whether they want to retry later. An active attacker could in this instance cause a denial of service by interfering with all authentication sessions between the two users; however we believe this is an acceptable risk. Such a failure does not result in a new identifier exchange and so the attacker cannot use a denial of service in this case to force new identifiers to be created and attempt a man in the middle attack at that point.

In Fig. 3.7 we see how Sobriquet fits into the data flow of a typical internet communications application. Our naming system performs resolution on an identifier that represents a person and maps it to a set of application specific identifiers. On contacting each other each party can authenticate each other to their global identifiers.

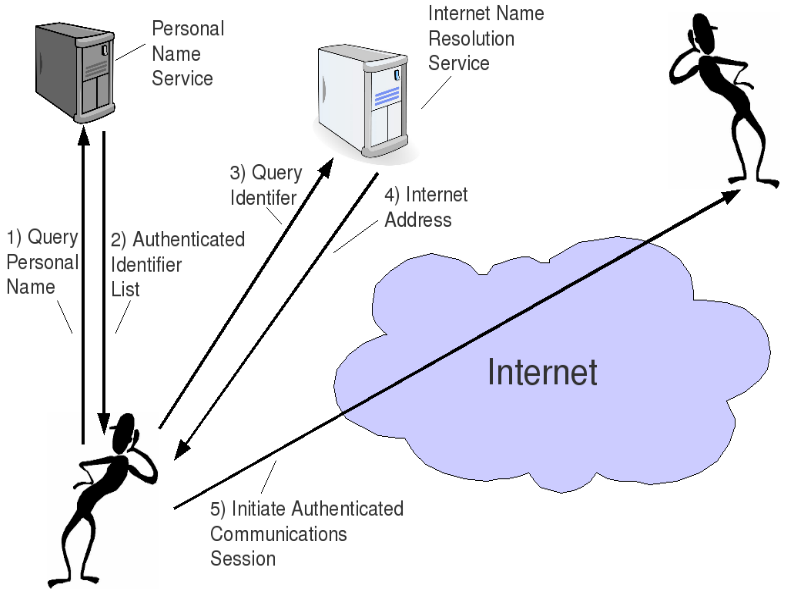
Personal Name Service
Internet Name Resolution Service
1) Query Personal Name
2) Authenticated Identifier List
3) Query Identifer
4) Internet Address
Internet
5) Initiate Authenticated Communications Session

Fig. 3.7: A typical internet communications session with Sobriquet.

3.8 Trust Establishment

We think it is desirable to allow people who have not met to contact one another, while providing some form of "authentication" between them. This process should prevent users from undertaking a Sybil attack [59] by authenticating them to some real world attribute. For instance, if Bob wishes to be contacted via email by people, he might decide to put his SGN on his website. However Bob may not know everyone who wishes to contact him this way and some of them may be spammers, scammers, or otherwise have ill intent. The ideal situation for Bob is to know which people are worth trusting and which are not. However this is not realistic without some form of authentication given that anyone that can send email has the potential to be a spammer.

Our solution to this problem bears similarities to that of the Receipt Management proposal of Bhargav-Spantzel et al. [31]. Like them we issue credentials that testify to a concrete transaction that the bearer has undertaken. We also intend for these credentials to be used in future transactions to bootstrap trust relationships with other parties. However, whereas they proposed an architecture for the issuing of receipts, which included the ability to return revoke receipts for returned items, like receipts in the real world allow people to return items to a shop, our proposal does not require this functionality.

Figure 3.8 gives an example of the use of Sobriquet references. Here Alice has undertaken a transaction, at some point, with the service provider "Friendface". After this transaction takes place Friendface indicates its willingness to issue a reference for Alice based on her participation in that transaction. Since Alice and Bob both trust Friendface, perhaps because both have undertaken transactions with that provider before, Bob is willing to accept references from Friendface. Alice obtains a reference that is bound to Bob's Sobriquet Global Name, so that she can only use that reference to authenticate herself to Bob. We assume she has obtained his public key through out of band means. She presents the reference to him along with a proof that it was issued to her. Should Alice successfully prove ownership of the credential the transaction may continue. If subsequently she abuses the reference and engages in malicious behaviour Bob can issue a complaint to Friendface, which will then affect Alice's ability to obtain references from it in the future. Friendface will have policies for dealing with complaints, which will most likely amount to refusing to issue references to her for a set time period.

The design of Sobriquet's references is motivated by the observation that, quite often, malicious transactions on the internet need to occur in a large volume in order for them to be profitable, which means that the frequency at which they can occur is important. For instance a spammer will need to send a large number of emails to obtain a return on his investment [78]. Our solution to these problems is thus to reduce the frequency at which they can occur, thereby reducing the profit, and hopefully significantly reducing the incentive for this malicious behaviour.

The transactions we are interested in providing references for are limited to those that have a real world cost, whether this cost is in terms of time or money. For Bob, purchasing an item on an e-commerce site has a real world monetary cost. Similarly

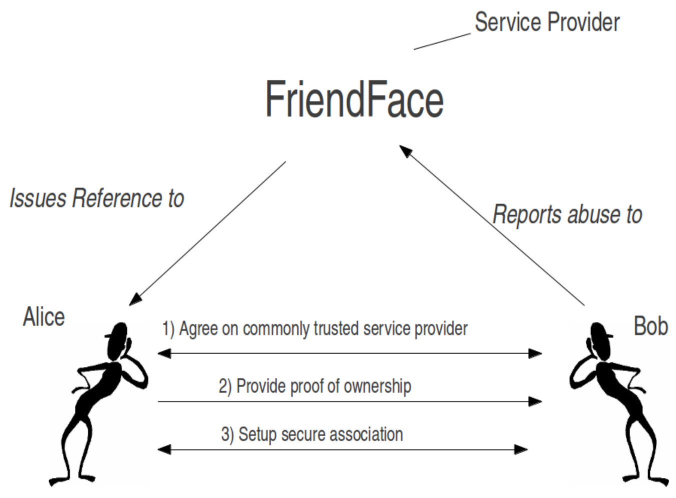
Service Provider
FriendFace
Issues Reference to
Reports abuse to
Alice
Bob
1) Agree on commonly trusted service provider
2) Provide proof of ownership
3) Setup secure association

Fig. 3.8: An overview of the use of references

accumulating a decent reputation on a website, such as eBay or Reddit for example, requires a relatively large investment of time. If we can make it so that the investment of time or money needed to successfully be able to, for example, send spam is more costly than the return on that investment then this should logically reduce the incentive for people to engage in this behaviour. For Bob however, they were investments he was going to make anyway. Bob is not likely to purchase an item he does not want or frequent a website that does not interest him. However, these transactions provide useful information about Bob to other people who he may wish to transact with in the future. Bob can obtain these references online from the issuing provider when he needs to bootstrap secure communications with someone he doesn't know. This means he doesn't need to store and protect them in the meantime.

For example, say Bob wishes to email Alice, but has no existing contact or relationship with her that he can use to introduce himself. He could present a Sobriquet Reference from a social news site or an e-commerce site, which indicates that he has undertaken a transaction with these sites that has required him to invest a reasonable amount of either time or money. Bob could stake his reputation on these references, knowing that Alice can complain to the issuing authority should he act maliciously and affect his ability to use them in the future. Alice also knows that the number of such references that Bob can obtain is limited so the frequency at which he can use references is limited. With these assurances in place it would seem reasonable that Alice would accept Bob's credential and continue the transaction. If he proves to engage in malicious behaviour despite Alice having the ability to issue a complaint about him, he will have to authenticate himself to her allowing Alice to filter his emails without much effort.

Privacy is of paramount importance as in this case Bob will be disclosing potentially personal information. He probably does not want the issuing authority to know that he is contacting Alice and he does not want the two of them to be able to collude to link his transactions with Alice to his transactions with the issuing authority. For this reason we have opted to use an anonymous credential system as they are designed to solve this very problem. As we saw in the second chapter, an anonymous credential is similar to a digital certificate but allows fields within the credential to be selectively revealed. This allows Bob to blind the sensitive values, which include his identity and Alice's public key, prior to having them certified by the issuing authority, during the issuing protocol. However, if Bob acts maliciously and receives a number of complaints then the issuing authority may wish to blacklist him from being able to obtain references in the future. This is possible under the anonymous credential system due to Brands [38].

Each reference contains the following information:

1. The public key of the issuing authority. In the example in Figure 3.8 this is the service provider Friendface.

2. The SGN of the person that the reference will be presented to. In the example above this would be Alice.

3. A timestamp indicating the issue date for the credential.

4. A set of values representing the transaction history that the reference is issued for. These are dependent on the issuing authority and are encoded differently on a per-issuer basis.

Additionally service providers will maintain a list of blocked pseudonyms that people wishing to obtain references must prove they are not on in order to obtain the reference. This list will be made available to an SP's users during that process, but as it only contains pseudonyms it will not infringe upon anyone's privacy.

3.8.1 Sobriquet Reference Protocols

In this section we will detail the protocols used to obtain and use Sobriquet References. The order they have been outlined in is the order in which they must occur, i.e. Alice must be issued a Reference before she can show it to Bob, and Bob must be shown the reference before he can issue a complaint about it. We make use of the Brands anonymous credential scheme we outlined in 2.7.1 for issuing and showing of these references, and both of these protocols closely mirror the corresponding issuing and showing protocols of the Brands scheme.

3.8.2 Issuing Protocol

What follows is our use of the anonymous credential issuing protocol [38] to obtain a credential for Alice on a public key of her choosing with Bob's public key and Alice's reference attributes encoded in the credential. We assume that Alice has an account with an issuing authority ahead of time, and that she meets the criteria of the issuing authority for obtaining references. All participants in the scheme that trust this issuing authority will have its public key stored as a trust anchor ahead of time.

The high-level steps are as follows:

1. Using the restrictive blinding protocol of Brands, Alice first blinds her own and Bob's public key. This is so that the issuing authority cannot link the Reference to her later.

2. She then authenticates herself to the issuing authority using whatever means is appropriate (this is left up to the issuing authority) and establishes a confidential channel. The rest of the protocol takes place over this channel.

3. She submits the blinded keys for signing by the issuing authority along with proofs of knowledge of the unblinded versions of those keys.

4. The issuing authority attempts to verify these proofs, exiting the protocol if the proofs fail to verify.

5. The IA encodes and adds the Reference Information to the Reference. This is the statement that the IA is making about Alice. The meaning of this value differs from issuer to issuer, and is presumed to be information that is known to people using and accepting those credentials. This information does not identify a person uniquely.

6. The IA sends the Reference to Alice.

3.8.3 Showing Protocol

This protocol specifies how a Sobriquet Reference would be used to bootstrap a secure channel between two entities. There are three distinct steps that Alice must take if she wants to present her reference to Bob. First she must obtain a reference from a provider that Bob trusts, she must then prove that she owns this reference, and finally Bob must be able to verify this proof and be satisfied that the reference is valid.

References may be issued by anyone and so there must be a way to determine which ones are trustworthy. Clearly there are entities who would be impacted negatively by

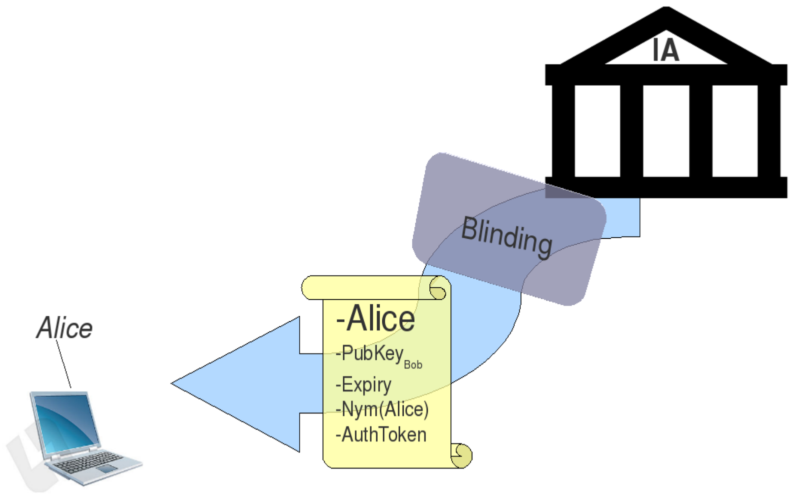

Fig. 3.9: The reference issuing protocol.

issuing bad references. Large organisations are usually very conscious of their public image and so would not be likely to abuse the use of references. However, there may be other lesser known issuers that may be trustworthy that should be taken into account. Every user of Sobriquet will have a list of acceptable issuers. We propose that such lists be compiled using a public peer review process, should be released fairly regularly, and be versioned. Each list should be made available under a permanent static URI with each version having a different URI, similar to how Document Type Declarations in XML are distinguished. While allowing multiple lists to be used may add to the complexity of the showing protocol, in reality the main Sobriquet implementations would likely quickly settle on a consensus for which lists make sense to use.

There is a similarity between the reference issuers in Sobriquet and the trust anchors in a PKI as they both certify public keys for the purpose of exchange. However, reference issuers are much more limited in the scope of what they can certify. In particular, even though the reference includes a public key, certification of that public key by a reference issuer does not bind the public key to any set of values that are associated with the holder of the public key. The public key is still assumed to have come from an undetermined source, with a notion of identity being ascribed to that key over time according to the principle of history based trust we discussed earlier.

Each list of issuers will contain a number of entries. Each entry will have at least a public identifier for the credential issuer, which will consist of the public key of the issuer, a URI for contacting them to obtain and complain about references, and a set of predicates. The predicates determine what values must be contained in each reference for it to be deemed acceptable. The verification process makes use of these predicates to ensure that a given reference is valid. These predicates would be expressed using a simple predicate language designed for the task. Say Alice obtains a reference from an e-commerce site to indicate that she has undertaken a monetary transaction with that site, an example predicate for that site may be "amount $>$ \$50". In other words if Alice cannot display a reference that indicates she undertook a transaction of at least \$50 with that site then her reference is deemed invalid.

When Alice contacts Bob he will provide her with the URI of the issuer list he accepts. This allows him to supply his own list if he would like to do so. Alice will then obtain a copy of that list from the URI if she doesn't already have a copy. From this list she will choose an issuer that she has a pre-existing relationship with. For instance, if her email service provider is on the list, she will obtain a reference from that provider. Assuming she meets the criteria set out for the provider to issue her with a credential she will obtain one according to the protocol set out in the previous section.

Successfully completing the protocol requires meeting three criteria:

- Proof of ownership of the credential - This requires proving knowledge of all the fields in the credential and follows the showing protocol of the anonymous credential system used.
- Satisfying a predicate for that issuer - Each reference will contain a number of fields. The list of issuers Bob consults to determine the validity of a reference

contains a list of predicates for each reference.

- Freshness - Sobriquet references cannot be revoked. Instead the date of issue is encoded into the credential. If the credential is too old it may be rejected. We have adopted this strategy, as proposed in [104], as opposed to a revocation list, since the maintenance of a revocation list has a significant overhead.

The showing protocols of anonymous credential systems consist of signed proofs of knowledge. If Alice shows her credential to Bob and reveals the field with her public key this will be the equivalent of an authenticated public key exchange. Bob can then use this public key to engage in the identifier exchange protocol listed above. In this way references allow for a secure relationship to be bootstrapped. However, even though both parties have been successfully authenticated there is one remaining part of the system to be described that is important to limiting the abuse of references.

3.8.4 Complaint Protocol

If Alice presents a reference to Bob and then proceeds to use the authenticated session to engage in undesirable conduct, for example spamming him, Bob can use the fact that her communications with him have been authenticated to ban her from communicating with him again. From that point on whenever someone authenticates themselves to her identifier Bob can choose to ignore any messages originating from them.

However the complaint protocol is an additional step that ensures that Alice cannot misuse her references to engage in such activity at a high rate. Since we contend that volume is necessary for such activity to turn a profit this aspect of Sobriquet essentially makes it unprofitable for people to engage in this activity.

After the showing protocol has taken place Bob will have obtained signed proofs from Alice that prove she is the holder of a reference. If Bob suspects she may be a spammer he may choose to engage in the complaint protocol. To achieve this he will send the proof to the issuer. The issuer can then blacklist Alice for a specified time period. This time period should be short so that Bob cannot affect her ability to obtain new credentials indefinitely. This in combination with the fact that credentials are not valid for a long period of time prevent misuse of this procedure.

3.9 Analysis

At the start of the chapter we outlined a number of requirements our solution must fulfill. These were:

1. No Trusted Third Parties for name allocation
2. Provide a way to securely identify people
3. Application Specific Identifier portability
4. Do not compromise the privacy of users in the way they identify themselves
5. Provide a means to establish secure communications between people who have never met

The first requirement was that the solution should not tie people unnecessarily to any identity providers or identifiers they cannot change. We achieved this by using identifiers based on a cryptographic namespace. To ensure our second requirement was met, we discussed how our identifiers are to be exchanged securely in the absence of infrastructure to ensure the authentication and confidentiality of the channels they are exchanged over.

Our name resolution protocol solved the issue of Application Specific Identifier portability, that was our third requirement. We proposed a proactive replication protocol that allows a user to perform a lookup to its local cache of SGN to name bindings most of the time. This should reduce the load on the Data Store considerably. The lazy lookup mechanism takes an optimistic view that its results are probably valid. If a failure occurs, such as an email bouncing for example, then a lookup may be performed.

Our fourth requirement was to avoid infringing the privacy of its users when authenticating themselves. The design of our authentication protocols plays a large part in achieving this goal. They attempt to avoid revealing identifying information about a person, they provide a means for establishing confidential channels to avoid having to have a user send everything in plain text, and finally they avoid leaving a digitally signed trail that allows a person's online interactions to be linked or a profile built about them.

The only component that is centralised in the infrastructure is the Data Store. No user is tied to using any particular Data Store and they can change the one they use at any time. Authentication takes place using information that is replicated across each of the user's devices. We favour automating the authentication process as in the Key Continuity Management (KCM) system. We extend the properties of KCM by solving the problem of authenticating unknown contacts, which addresses our fifth requirement.

By decentralising our solution our approach is scalable, as argued in 3.6.3. The use of public key cryptography should introduce a manageable amount of overhead. Use of a short group signature scheme, such as the one due to Boneh et al [36] could further reduce the overhead imposed by the cryptography element. By not requiring a third party to be present for the authentication process we eliminate a potential bottleneck. The persistent connection between the Data Store Browser and the Data Store will mostly be idling and so will not impose a large amount of overhead either.

3.9.1 Security Analysis

Fig. 3.10 shows an attack tree with possible attacks against the Sobriquet system. Nodes coloured the same represent the same attack. The first hierarchy of nodes from the centre are the components of the system that can be attacked. From there, the next level in the hierarchy depict goals an attacker may have. Finally, stemming from that are ways of achieving those goals. We will now discuss each of the attacks and their feasibility.

Man in the Middle Attack

We have already discussed in depth the type of channel that should be used for key exchanges, namely those keys that are used by Sobriquet to encrypt and sign information. When this exchange happens over an insecure channel, Sobriquet is vulnerable to a Man in the Middle attack. However, Sobriquet takes the optimistic view, in this instance, that an attacker is not present. If it becomes obvious that the identifiers have been compromised at a later date then each party can revoke access to the compromised keys and attempt another exchange.

DoS Data Store

Since clients depend on information that is only accessible from the Data Store this could prove to be a tempting target for attackers wishing to deny service to a particular user. The use of client caching mitigates this issue somewhat, since if the client already has most or all of the data it requires stored in this cache then the DoS will not be fully effective.

Password Guessing Attacks

Authentication between the Data Store and the client is through the use of a password. There are two ways this issue could be mitigated. First password authentication could be substituted by stronger forms of authentication. Second, the use of adequately strong passwords would ensure enough entropy so that a brute force attack on the authentication scheme would prove ineffectual. Since the password is the one piece of state the user must take care of themselves, we accept this trade off in exchange for reduced complexity of user authentication.

Steal References from Client Machine

Botnets could be set up to harvest references from client machines. This problem is reduced somewhat since references are bound to the SGN of the person they are to be showed to and they are obtained as they are needed. Targetting specific users in this way would require someone obtaining a reference to communicate with them, and the botnet being active on that persons machine. Also, references are single use so if a client attempts to show them twice then this will indicate an error.

Fake Reference Requirements

References depend on work having been done and certified by a trusted third party. This could either be faked by an attacker, or they could hire someone else to do this on their behalf. However, an issuing authority will only issue references at a certain rate to each person so even if an attacker circumvented the requirements they would still only obtain them at a low rate. Secondly, the complaint protocol allows misuse of references to be tracked and dealt with.

Push False Results

Since the Data Store accepts messages pushed to it from any source there is a risk that attackers may try and exploit this by pushing fake result sets for contacts to a user's Data Store. Only contact's result sets can be updated in this way. Other updates require authentication from users. Result sets must be signed by the private key of the SGN they correspond to. When result sets are pushed to a Data Store, the Data Store verifies the signed results against the SGN they will be stored under. If this verification step fails then the update is rejected.

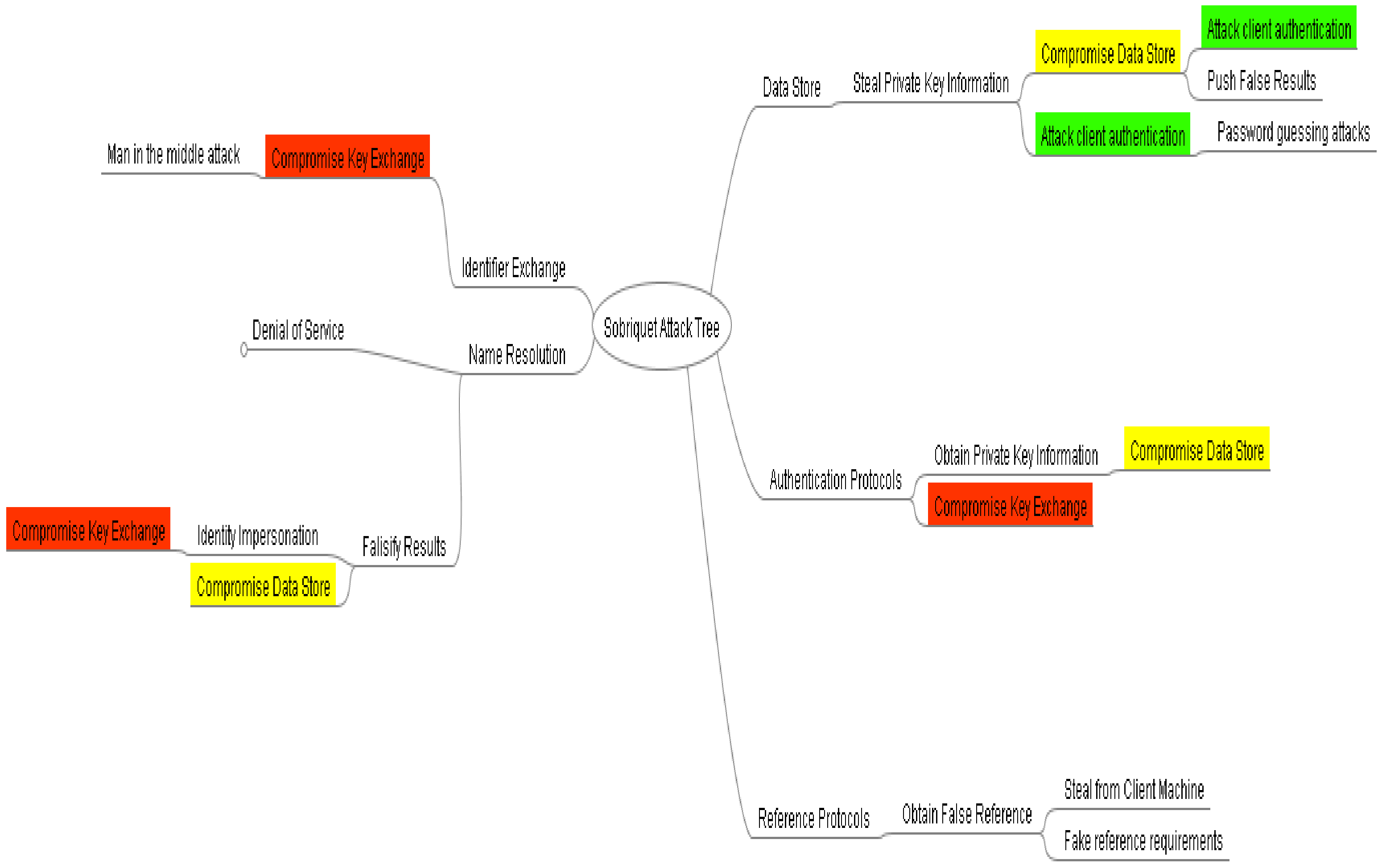

Fig. 3.10: An attack tree showing possible attacks on Sobriquet.

3.10 Conclusion

This chapter outlined the design of our naming and identification system. Section 3.1 outlined the problem statement at a high level. We then delved into the requirements in more depth in the rest of that section. We broke the description of our design into four sections. Section 3.3 took a look at Sobriquet Identifiers and how they are created and exchanged. The exchange protocol we outlined showed how a common channel is used for the exchange. This common channel could be location limited as in the case of Bluetooth, might be provided by a third party as in the case of using a web service, or may be completely unsecured. However, while we explained how this protects the exchange of identifiers without depending on dedicated infrastructure, we never explained how Alice and Bob could accept identifiers from each other in confidence without a pre-existing relationship of some description.

Section 3.7 took a look at the name resolution component of our system. We mentioned how we aimed to keep the design of this component simple, but resilient to failure. To this end we outlined our caching algorithm that proactively replicates information as it updates, with a view to reducing the overhead involved in the resolution process.

We outlined our Trust Establishment component in Section 3.8. Sobriquet aims to bring assurances of identity where there is no preexisting relationship between the participants. In this case, even though Alice and Bob may not have communicated with each other, they most likely have communicated with others or used online services. In other words, there are third parties who can attest to their trustworthiness. We realise that trust is an often overloaded term in the literature, and so a definition is necessary. For our purposes trust was defined as the probability of an entity engaging in undesirable behaviour. Unlike trust metric systems we do not attempt to quantify and assign a degree of trust to actors, but rather reduce the probability of actors behaving maliciously by making it expensive, in terms of monetary cost, to behave maliciously at scale.

Finally, we provided an analysis of our solution with respect to our requirements in Section 3.9. We reiterated the requirements we had set out in the beginning of the chapter, which in turn had come from our analysis of a use case in the first chapter. We also performed a threat analysis of our solution. This threat analysis considers those threats that we have specifically secured our system against.

4. IMPLEMENTATION

This chapter discusses an approach to implementing Sobriquet and outlines a proof of concept we have developed. To implement the system fully would require altering applications, which would be a large undertaking for a single developer. However, were application developers to take it upon themselves to implement use of the Sobriquet API within their applications, this would not require a substantial investment of time and effort on their part.

In this chapter we will discuss the implementation under three main sections. These are the naming system component, the authentication protocol, and secure identifier exchange protocol. These will be discussed in detail in the second, third, and fourth sections respectively. Finally we will give an overview of an example use case in the fifth section before concluding the chapter in the sixth section.

4.1 Implementation Design

The implementation of the system is designed to exist entirely in user space without requiring administrative privileges, for the most part, to allow it to be easily deployed for the purpose of evaluation. This requires configuration of user software which can be performed automatically in some cases, but requires user interaction in others. We will outline methods for avoiding the need for configuration, a trade off that was made to reduce complexity while allowing the implementation to remain portable.

4.2 Naming Component

The naming system component is composed of software that is deployed on each of the users' devices, the Data Store Browser, and the Data Store component that manages synchronisation issues. We will discuss these separately as well as discussing the choice of communication protocol between them.

4.2.1 Data Store

The Data Store component has two main functions:

1. to synchronise state between each of a person's devices. This state, outlined in the design chapter, is comprised of the information needed to authenticate a person's contacts, the contact's name to result set mappings, and time stamping information so we know at what point an update has been pushed.

2. to push updates of a person's name to result set mappings to their contact's Data Stores.

This component is best implemented as a web application. The reason for this is that most platforms come with libraries for HTTP communication, and most networks that have proxies and firewalls generally allow external HTTP access. We see the components of the web application in Fig. 4.1. Although the diagram depicts components as different machines, the various components are separated because they perform different logical tasks and it is convenient to depict them separately. In reality each of those components may be running on the same physical machine.

The front end server handles communication with the client, retrieving and storing information in the DB on their behalf. This server also handles authentication of the clients and provides the logic of the Data Store application.

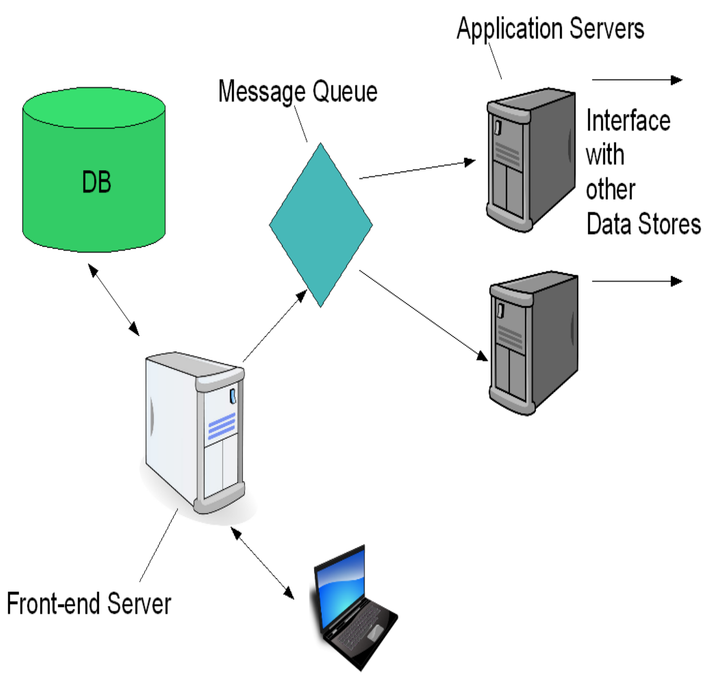

Fig. 4.1: The Data Store components.

Some functionality of the Data Store needs to be performed asynchronously, and the Data Store component relies on a number of back-end servers to do this. The front end server creates jobs for them to perform and adds them to a queue. The back end servers then remove jobs from the queue and complete them. These jobs generally involve communicating with other Data Stores, a task that may fail and need to be retried.

4.2.2 Communication protocols

Client to Server

We considered a few different transports for connecting clients and servers including stand alone TCP, XMPP, AMQP, and HTTP. Communication between clients and servers, in our architecture, uses the WebSockets protocol [64] when possible and falls back to regular HTTP when not it is not possible to use them. The WebSockets protocol facilitates bidirectional, full-duplex communication across a HTTP connection[1]. This allows servers to push data to clients in a standard way, which before was only possible with a set of techniques known as Comet, that provide this functionality by exploiting browser features in a non-standard way.

The WebSockets protocol is designed to work through most web proxies, which makes it useful as a transport for use behind restrictive middleboxes. This was a major deciding factor in our choice to use WebSockets as the communication protocol. Furthermore, use of a new protocol would have meant that it would have had to circumvent firewalls itself, much like Skype does for example, which is a needless increase in complexity. The stateless GET and PUT semantics of the HTTP protocol map well to the functions provided by the Data Store. The WebSockets protocol reduces the complexity of building our application substantially. Finally, web applications are traditionally designed to be stateless, for reasons of scalability and robustness in the face of failure and network error. The HTTP protocol has influenced our design in this way and we have attempted to design both components to maintain as little state as possible. Where state has to be stored it is stored persistently and synchronised across all clients.

REST API

As we mentioned in the previous chapter, authentication between the client and server is performed using a shared secret, e.g. a password. To provide access to the API for a given user, the server must have an authenticated session with that client established. Once the session has been established the server knows the user table entries that the client has access to. The following PUT and GET calls operate on those entries that the user has access to.

Operations on the user table map to the relative URI */users/USERID* and opera-

[1] Unfortunately some proxies do not yet support it and so it may be necessary on occasion to resort to using regular HTTP

tions on the contacts table map to the URI */users/USERID/contact/CONTACTID/*, where USERID and CONTACTID are secure hashes, computed using RIPEMD-160 [58], of the SGN and the contact's SGN respectively. To update an entry a PUT request is made with a serialised key-value record. To update a result set record an additional verification of the record against the corresponding public key (SGN) is required. Additionally the Data Store exposes the URI */contact/CONTACTID* for PUT operations, where the CONTACTID is once again the hash of the contact's SGN. Since PUTs on this URI will require a digitally signed result set that can be verified with the public key that already exists in the database, this functionality is safe to expose. This PUT request is used by other Data Stores when pushing updates on behalf of one of their clients.

Server to Server

To simplify the design we tried to keep all components in the system as loosely coupled as possible. To achieve this we split them up into separate processes, which interact using a message passing protocol. Message passing protocols often offer a light-weight, asynchronous alternative to RPC mechanisms. AMQP is one such protocol, with a robust server implementation in RabbitMQ [8] and client libraries available for most languages. For security, TLS can be used between clients and the message queue itself.

The servers only communicate with each other on behalf of the clients. As such they are mostly stateless and need only maintain and ensure delivery of messages in their message queue. In order to maintain resistance to failure each server maintains a persistent queue of messages and may be killed or restarted at any time without message loss.

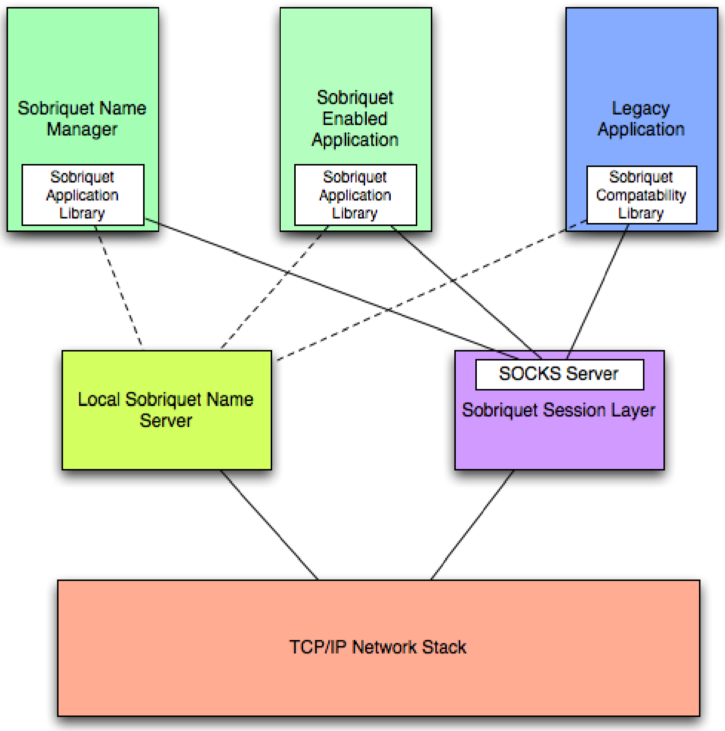
Sobriquet Name Manager
Sobriquet Application Library
Sobriquet Enabled Application
Sobriquet Application Library
Legacy Application
Sobriquet Compatability Library
Local Sobriquet Name Server
SOCKS Server
Sobriquet Session Layer
TCP/IP Network Stack

Fig. 4.2: The Sobriquet client side architecture.

4.2.3 Client-side

The client side software of the naming system component maintains a Local Cache of the information stored on the server on its behalf. State is maintained across each of the client's devices. They each synchronise with the server to ensure that each of the devices has the most up to date state. This synchronisation takes place over a persistent WebSockets connection to the server, which each client maintains.

As depicted in the figure 4.2, the client side software makes use of two libraries to allow applications to take advantage of Sobriquet. The first is the Sobriquet Application Library (SAL). This allows applications to take advantage of the functionality provided by Sobriquet and is the preferred mode of operation since application developers can integrate the system fully into the application. The second mode is to use the Sobriquet Compatibility Library (SCL). This enables legacy applications, which have not been ported or cannot be ported to use the Sobriquet Application Library, to use the functionality provided by Sobriquet.

An application that uses the Sobriquet Application Library makes calls to the Sobriquet name resolution (Section 4.2.4), authentication (Section 4.4), and identifier exchange (Section 4.3) functions. These are integrated into the application through the Sobriquet library, which communicates with the main Sobriquet process that runs on each client. The Sobriquet Compatibility Library provides for Sobriquet naming and authentication by intercepting the socket calls made by an application as it attempts to connect to a host. Sobriquet intercepts socket calls such as *socket*, *bind*, *getnamebyaddress*, etc. and provides support to legacy applications by way of its compatibility layer. The Sobriquet Compatibility Library is just a thin layer that intercepts socket calls on a given platform and as such the API it exposes is dictated by the socket API on each platform.

4.2.4 Coexisting Namespaces

Sobriquet names are integrated into the DNS namespace for backwards compatibility purposes. This allows people to use names such as *"rob.sbq"* to refer to people. The TLD *.sbq* is used as a suffix for Sobriquet names and all queries for names under this TLD are routed to the Sobriquet name resolution process on the user's machine. Since a Sobriquet name can map to any number of results the name resolution library selects an appropriate subset of results given the context of the application that is performing the query. For instance if the query originated from a web browser then the result must be a web address.

When the query is performed in SAL mode then the application itself is responsible for handling the results. If there are multiple results it can choose to select an appropriate one, or present all the results to the user and allow them to select the appropriate results. However, when the SCL is in use the Sobriquet Name Server is responsible for returning the result in the correct form. To achieve this it retrieves the context information from the query, loads an appropriate plugin, which then performs further processing on the results in order to return a usable result.

4.3 Secure Identifier Exchange

Sobriquet also provides an API for performing the Secure Identifier Exchange protocol. This API abstracts the exchange detailed in Section 3.5.1, and also handles obtaining and using references.

The API may be used in conjunction with a reference issuing authority or without. As shown in Fig. 4.3 the role of the issuing authority is to provide a secure channel for the initiator and responder when only an insecure one exists. The API is layered so that it may be used across different transports. This is to reflect the fact that the exchange should take place, when possible, over a common "secure" channel, that is a channel that both the initiator and recipient share that performs its own authentication. A good example of this is a web service that offers HTTPS access. In this case the initiator and recipient can exchange data over the HTTPS channel with the service provider, assuming that they both trust the provider to some extent. We argue that this "trust" is no different from the trust a person has in a service they use. Furthermore all information exchanged over this channel can be verified after the fact using a different channel if necessary.

The API for providing Secure Identifier Exchange in Sobriquet consists of a number of different layers. These are shown in Fig. 4.3. A Sobriquet enabled application makes use of the high level Secure Identifier Exchange API. This API provides the functions necessary to exchange identifiers over a common channel. This API provides the following core functions: *sbqidEstablishChannel*, *sbqidCreateNewIdentifier*, *sbqid-SendNewIdentifier*, *sbqidVerifyNewIdentifier*, and *sbqidStoreNewIdentifer*.

These provide the following functionality:

- **sbqidEstablishChannel:** Establishes a channel on behalf of initiator over which the identifier can be exchanged. This channel will either be through an Issuing Authority or directly to the other participant in the exchange. This takes

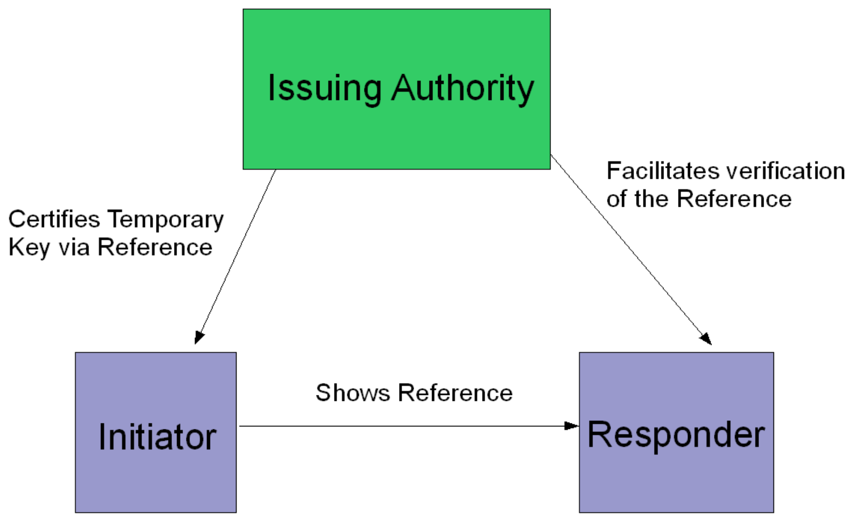
Issuing Authority
Certifies Temporary Key via Reference
Facilitates verification of the Reference
Initiator
Shows Reference
Responder

Fig. 4.3: The relationship between the participants and the issuing authority.

as arguments a URI of the target, and returns a handle (unique integer value) to identify this channel.

- **sbqidCreateNewIdentifier:** Creates a new identifier that will be assigned to and sent to a person for them to use for authentication in future sessions with the initiator. This takes as arguments a Sobriquet Memorable Name and a Contact Group in the form of two strings, and returns the SGN corresponding to that Contact Group and the created group membership private key for the new member.

- **sbqidSendNewIdentifier:** Sends the identifier across the established channel. This takes as arguments the handle created by a call to sbqidEstablishChannel and the values returned by sbqidCreateNewIdentifier. This function abstracts the whole process of performing the exchange as detailed in 3.5.1.

- **sbqidReceiveNewIdentifier:** Receives a new identifier from the creator. The argument for this function is a callback function, the handle created by the sbqidEstablishChannel function, and an SMN. The callback may be the sbqid-

StoreNewIdentifer function in a simple case, or may be a function used by an application to perform some validation on the received values before storing them.

- **sbqidStoreNewIdentifer:** Stores the identifiers received from the target. This takes as arguments an SGN, a group membership private key both of which are received from the other participant in the exchange and are to be used by the user from that point on to identify themselves to the other participant. This also takes an SMN and a Contact Group. These parameters will be passed in from sbqidReceiveNewIdentifier.

The Common Transport API provides a common interface to the protocol specific layers beneath it. This is the layer that binds the two together and defines what interfaces the protocol specific layers must be capable of providing.

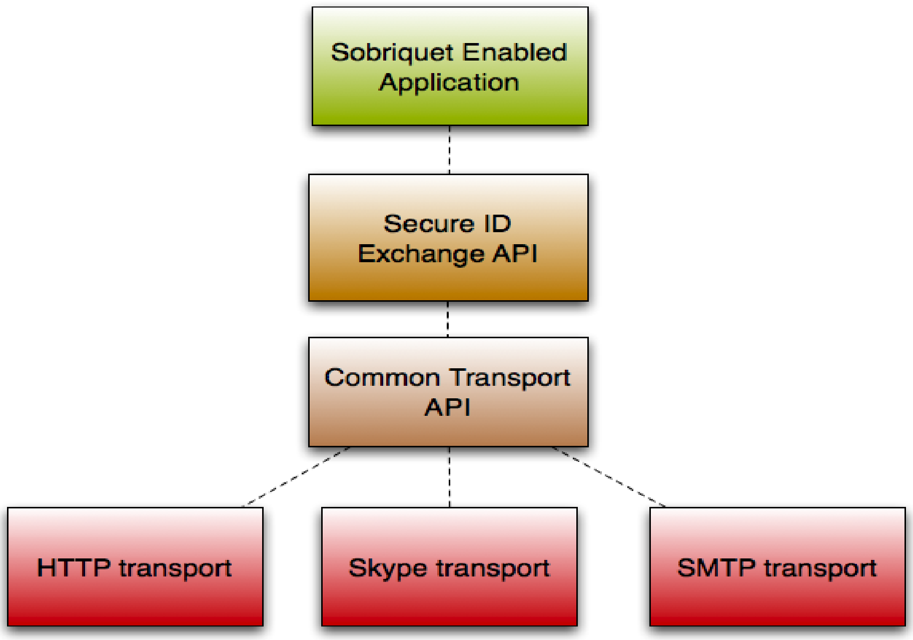

Fig. 4.4: The layers in the design of the Secure ID Exchange API.

4.4 Authentication Protocols

The authentication protocol was designed to be used in two ways. The first signs messages used in store and forward protocols, such as email. To make this possible the implementation's API exposes functions for authenticating and verifying messages, *sbqSign* and *sbqVerify*. The functions take as arguments a message either to be signed in the case of the signing function or the signed message to be verified in the case of the verify function, and the Sobriquet Memorable Name of the person that it is intended for, both of which are strings. The functions do a look up in the local cache for the SMN to obtain the corresponding keys. This function maps to the security protocol in 3.4.2.

4.5 Proof of Concept

To prove that the system was viable we implemented a proof of concept. This consisted of a cryptography library that implemented the group signature scheme and an implementation of the Sobriquet Browser and Compatibility Layers.

The cryptography library included the complete functionality of the group signature scheme including the setup, join, sign, verify, and open functions. The scheme we implemented was the one due to Camenisch and Groth. The implementation furthermore was capable of simulating message exchanges and could provide the non-interactive protocol. The overhead from the cryptography library was roughly in line with what we expected from reading the literature of the group signature scheme we implemented.

Our implementation led to an analysis of the cryptographic primitive that underpins our protocols. Using our group signature scheme implementation we generated a group public key, and added one member to the group. Since the functions of the scheme are independent of the number of members of the group, this should give us an adequate test of the overhead of the operations in general. We ran this implementation on an

Function	Average (ms)	100 iterations (ms)	per second
sign	29.3	2930	34
verify	37.0	3700	27
open	1.6	160	625

Tab. 4.1: Time taken to perform cryptographic functions of the group signature scheme.

AMD Athlon(tm) 64 Processor 3700+. The results of the tests are as shown in Table 4.5.

This implementation is, as a proof of concept, lacking optimisation. We should be able to get better performance by applying some easy optimisation fixes. For instance, some values during the signing function can be pregenerated. These were not in our implementation and as such the time taken by the signing function could be reduced. As a benchmark we also ran the speed test as part of the OpenSSL suite for RSA-2048. With this we achieved 263 signs per second, with each taking roughly 3.8 milliseconds, and 9634 verifies per second, with each of those taking approximately 0.104 milliseconds. Our group signature implementation is thus, when it comes to signing at least, roughly an order of magnitude slower than RSA.

As far as message sizes go, the group signature scheme is larger than RSA. Using the recommended parameters for the scheme as set out in the paper, we calculated the group public key to be 931 bytes, whereas an individual signature is 696 bytes, and a group member private key is 450 bytes. While the outputs of this group signature scheme are larger than that of RSA they are not prohibitively large in general usage. All of the above would, for instance, fit within a single Ethernet frame. In applications where signature length is an issue, the scheme due to Boneh et al. [36] claims to produce signatures that are "approximately the size of a standard RSA signature with the same security". We have not implemented this scheme, but implementing it within the context of Sobriquet would be a useful avenue of investigation.

We also implemented the Sobriquet Compatibility Layer in the form of a shared library. This was successfully injected into an existing application; Firefox. This shared library intercepted the DNS functions of that application, and routed the queries to another local process. This process running on the same machine sent back fake results to simulate a lookup to the local cache in our Sobriquet design. This shows the viability of the Sobriquet Compatibility Layer to bring the name resolution functionality of the system to existing applications without recompilation.

We furthermore developed a simple HTTP proxy in Python with the goal of extending it to use the Sobriquet cryptography libraries. To this end we provided a wrapper for the cryptographic libraries suitable for this use. The wrapper was automatically generated from the existing C library using SWIG [11], which allowed access to the group signature scheme functions and data structures from within Python. The library was shown to be usable in this way as an existing test application written in C was ported to Python.

The next stage for the implementation is to finish the implementation and deploy it in a networked environment. We would like to do this to prove the operation of our system as outlined in our use case. This extended implementation could then be used, for example, in user trials to see what usability obstacles exist in the deployment of Sobriquet in a real world application.

5. CONCLUSIONS AND FUTURE WORK

We began by stating our belief that solving the issue of identity on the Internet is an important problem. The rise of identity management systems in the past few years shows that this view is held by many. We note Kim Cameron's statement that without solving this problem we will end up in a situation where people may lose confidence in the infrastructure. Centralised identity infrastructures have so far failed to solve this problem. We take the view that a centralised key management infrastructure should be used if present, but if not then the next best thing is to perform key exchange however possible and attempt to reduce the ability for people to tamper with this process through the use of out of band verification where possible. We analysed a number of systems in the second chapter from the point of view of our requirements, and detailed the reasons they did not meet these requirements. In the course of this investigation we came across elements of existing systems that we incorporated into our design.

With these observations in mind we proposed a solution to naming and authenticating people on the Internet. Our solution placed an emphasis on three different properties: decentralisation, privacy, and usability. We aimed for a decentralised infrastructure that actively avoided the use of identity providers and tried not to tie people into using third parties they could not change. We prioritised privacy as a means of protecting people who do not have an adequate understanding of online privacy issues, but also as a means of protecting service providers from embarrassing data leaks that could potentially open themselves up to legal action under data privacy laws. Finally our solution sought to automate operation as much as possible so as to reduce the level of user involvement required for proper operation of the system.

Our aim is to complement the use of Application Specific Identifiers as identifiers that are used to refer to people. We hope that Sobriquet identifiers would be exchanged by people who wish to communicate with one another in much the same way that email addresses are now. Depending on the mode of exchange used, these identifiers will take on different meanings. For instance two people who meet up and exchange identifiers will have a strong assurance of who those identifiers refer to, while identifiers exchanged over an insecure network will assure participants that the parties present are the same as the ones that were present at the exchange. Identifiers are public keys and are used for encryption and signing in the authentication protocols specified as part of our identity management solution. We refer to the type of authentication Sobriquet provides as history-based authentication. That is that the quality of authentication is dependent on the history of interactions between the people involved.

We refer to the identifiers used as part of the authentication protocols as Sobriquet Global Names and the naming system component maps them to a set of results. Every SGN refers to a specific Contact Group. A Contact Group is a set of people who are allowed access to the mapping between a particular SGN and its result set. The use of a group signature scheme as the basis for SGN allocation allows a person to prove that they are a member of a certain Contact Group without revealing any further identification information. Third parties such as the Data Store can verify such a proof without obtaining the identity of the prover. This provides privacy preserving access control to the Data Store. While we place no constraints on what is contained in these result sets we expect them to quite often be Application Specific Identifiers.

Our trust establishment framework sought to bootstrap authenticated communications between people who have never met. This allows for people to interact without disclosing more identity information than they want to, while providing both parties with a level of assurance that the other party in the transaction cannot create new identifiers and misuse them at will. The solution to this problem was motivated by the assumption that for many undesirable activities, such as spam, to be economically viable spammers need to be able to send emails at a high frequency. If they cannot then it is harder for them to realise a profit. We reduce their capacity to send emails at a high frequency by tying their ability to communicate anonymously to a real world cost. The fact that this cost has been incurred is attested to by a mutually trusted third party.

The work we presented demonstrated a need for a system such as Sobriquet. If it saw deployment Sobriquet could bring the following significant benefits to online interactions. First it would bring authentication and confidentiality to personal communications systems where currently it is lacking. Second people would be more easily able to manage their interactions with others online by applying filters and routing preferences to the way they are contacted. The use of identity in this case would allow them to express high-level but powerful rules expressing these preferences. Finally, adopting Sobriquet would reduce the cost of trying and using new applications online. This is currently a major obstacle in the deployment of new systems. As evidenced by the number of third party applications supporting systems like Facebook Connect and OpenID this obstacle is of concern to application developers. Deploying Sobriquet would allow this process to be automated, thereby eliminating these obstacles. As moving between different application providers would no longer be an issue from an identity stand point, providers would be forced to compete on features.

5.1 Future Work

There were a number of additions we would have liked to have made to the Sobriquet design and that we feel warrant further investigation. The first of these, and the next logical step to take with Sobriquet would be to ascertain its usability through user trials. Only when this has been demonstrated could it be considered for deployment. One possible way to go about encouraging its adoption would be to integrate it into the popular browsers. This could be done using their plugin system. If a user goes to a site that requires Sobriquet authentication they could be redirected to a download page. The use of social networks as a means of spreading its use warrants investigation. We believe that bringing strong notions of identity to the web could be facilitated by sites such as Facebook. The fact that it may be used as communications platform, allows third party applications, and is in use by a large number of users means that it could be used as a channel for exchanging Sobriquet identifiers. Facebook already has its own identity solution called Facebook Connect. However this solution is similar to OpenID and as such ties the user to the Facebook platform. Given that Facebook allows third party applications it should be possible to deploy a Sobriquet-based solution that pitches itself as an alternative to Connect.

Our solution attempts to require as little user involvement as possible. We hope that this provides for usability gains. However the only way to know for sure would be to undertake usability trials. Issues that warrant investigation in these trials would include ways of effectively presenting errors to the user, visual aids that distinguish between identifiers that were verified and those that were not, and ways of integrating the use of Sobriquet identifiers with other authentication systems. The studies undertaken by the authors of [69] could be a useful starting point.

We conjecture that the use of a proactive replication protocol would reduce the load on the Data Store in our design. This claim could be further substantiated by an analysis of the usage patterns of a number of users of our system. We could further ascertain where the latency is incurred in different Internet applications. This could be done by leveraging the Sobriquet implementation's integration into applications to measure the time between resolution of the Sobriquet identifier and invocation of the authentication protocol. Sobriquet could be extended to try and reduce latency where possible. One approach may be to have it perform DNS lookups ahead of time similar to the way some modern web browsers pre-resolve the domain names of links in a page so that the next one loads more quickly.

We advocate the use of existing secure channels as a way of exchanging cryptographic identifiers while reducing the ability of an active attacker to subvert the exchange. Our approach builds in the abstractions for implementing interoperation with these channels, but does not currently integrate with any. One mode of exchange we advocate is the use of location limited channels such as Bluetooth or NFC (Near Field Communication). These exchanges would benefit from a means of automatic configuration in the form of a secure Zeroconf like protocol that handles establishing a channel, undertaking an exchange, and provides a simple way for users to verify the exchange. Implementing such a protocol would provide a better idea of the challenges involved in the use of these channels.

Finally, we believe the issue of bringing identity and authentication to cross-layer interactions would be of benefit. Sobriquet could help in this vein by providing a top-down chain of authentication that facilitated this functionality through the layers. As an example it could make sense for an IPsec implementation to make use of the authentication provided by Sobriquet to establish secure associations between user devices. This would require an investigation into the requirements to make this possible at each layer.

BIBLIOGRAPHY

[1] "Google Voice - One phone number, online voicemail, and enhanced call features," Web site: [Last accessed: 18/10/2011]. [Online]. Available: https://www.google.com/voice

[2] "Intel Q45 Express Chipset Overview," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.intel.com/products/desktop/chipsets/q45/q45-overview.htm

[3] "Liberty alliance," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.projectliberty.org/

[4] "LID," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://lid.netmesh.org/

[5] "Microsoft U-Prove," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://connect.microsoft.com/site1188

[6] "OpenID," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://openid.net/

[7] "PhoneGnome Homepage - http://www.phonegnome.com," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.phonegnome.com

[8] "RabbitMQ," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.rabbitmq.com/

[9] "RDF/XML Syntax Specification (Revised)," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[10] "Shibboleth," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://shibboleth.internet2.edu/

[11] "Simplified wrapper and interface generator," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.swig.org/

[12] "The Distributed Computing Environment Portal," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.opengroup.org/dce/

[13] "The NFC Forum," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.nfc-forum.org

[14] "Yadis," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://yadis.org/

[15] "Yahoo! OpenID Usability Research," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://developer.yahoo.com/openid/bestpractices.html

[16] "Gnutella protocol v0.4," Web site: [Last accessed: 18/10/2011], 2001. [Online]. Available: http://www9.limewire.com/developer/gnutellaprotocol0.4.pdf

[17] "eBay Inc. Online Auction and Shopping Website." 2011. [Online]. Available: http://www.ebay.com/

[18] ABC news, "China's Facebook Status: Blocked," Web site: [Last accessed: 18/10/2011], July 2009. [Online]. Available: http://abcnews.go.com/blogs/headlines/2009/07/chinas-facebook-status-blocked/

[19] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in *SOSP '99: Proceedings of*

the seventeenth ACM symposium on operating systems principles. New York, NY, USA: ACM, 1999, pp. 186–201.

[20] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS Security Introduction and Requirements,” RFC 4033 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFC 6014. [Online]. Available: http://www.ietf.org/rfc/rfc4033.txt

[21] ——, “Protocol Modifications for the DNS Security Extensions,” RFC 4035 (Proposed Standard), Internet Engineering Task Force, March 2005, updated by RFC 4470. [Online]. Available: http://www.ietf.org/rfc/rfc4035.txt

[22] ——, “Resource Records for the DNS Security Extensions,” RFC 4034 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 4470, 6014. [Online]. Available: http://www.ietf.org/rfc/rfc4034.txt

[23] P. Argyroudis, R. McAdoo, S. Toner, L. Doyle, and D. O’Mahony, “Analysing the Security Threats Against Network Convergence Architectures,” in *Proceedings of 3rd International Symposium on Information Assurance and Security (IAS’07).* Manchester, UK: IEEE Computer Society, August 2007, pp. 241–246.

[24] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, “A practical and provably secure coalition-resistant group signature scheme.” Springer-Verlag, 2000, pp. 255–270.

[25] T. Aura, “Cryptographically Generated Addresses (CGA),” 2003, pp. 29–43. [Online]. Available: http://dx.doi.org/10.1007/10958513_3

[26] A. Back, “Hashcash - a denial of service counter-measure,” Web site: [Last accessed: 18/10/2011], Tech. Rep., 2002. [Online]. Available: http://www.hashcash.org/

[27] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, “A layered naming architecture for the internet,” in *SIGCOMM ’04:*

Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM, 2004, pp. 343–352.

[28] S. A. Baset and H. G. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1–11, Apr. 2006. [Online]. Available: http://dx.doi.org/10.1109/INFOCOM.2006.312

[29] T. Berners-Lee, R. Fielding, and L. Masinter, "RFC 3986, Uniform Resource Identifier (URI): Generic Syntax," 2005. [Online]. Available: http://rfc.net/rfc3986.html

[30] T. Berson, "Skype security evaluation," Web site: [Last accessed: 18/10/2011], 2005. [Online]. Available: http://download.skype.com/share/security/2005-031%20security%20evaluation.pdf

[31] A. Bhargav-Spantzel, J. Woo, and E. Bertino, "Receipt management- transaction history based trust establishment," in *DIM '07: Proceedings of the 2007 ACM workshop on Digital identity management.* New York, NY, USA: ACM, 2007, pp. 82–91.

[32] A. D. Birrell, R. Levin, M. D. Schroeder, and R. M. Needham, "Grapevine: an exercise in distributed computing," *Commun. ACM*, vol. 25, no. 4, pp. 260–274, 1982. [Online]. Available: http://dx.doi.org/10.1145/358468.358487

[33] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote Trust-Management System Version 2," RFC 2704 (Informational), Internet Engineering Task Force, September 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2704.txt

[34] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in

In Proceedings of the 1996 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 1996, pp. 164–173.

[35] P. Bondi and F. Desclaux, “Silver needle in the skype,” Blackhat Europe, March 2006. [Online]. Available: http://home.dei.polimi.it/giacomaz/courses/Telematica/materiale/Skype/bh-eu-06-biondi-up.pdf

[36] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” in *Advances in Cryptology—CRYPTO 2004*, ser. Lecture Notes in Computer Science, vol. 3152. Berlin: Springer-Verlag, 2004, pp. 41–55, available at http://www.cs.stanford.edu/~xb/crypto04a/.

[37] S. Bradner, L. Conroy, and K. Fujiwara, “RFC6116: The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM),” p. 4, 2011.

[38] S. A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* Cambridge, MA, USA: MIT Press, 2000.

[39] ——, “A technical overview of digital credentials,” Web site: [Last accessed: 18/10/2011], 2002. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.16.3617&rep=rep1&type=pdf

[40] E. Brickell, J. Camenisch, and L. Chen, “Direct anonymous attestation,” in *Proceedings of the 11th ACM conference on Computer and communications security*, ser. CCS ’04. New York, NY, USA: ACM, 2004, pp. 132–145. [Online]. Available: http://doi.acm.org/10.1145/1030083.1030103

[41] D. Brickley and L. Miller, “Foaf vocabulary specification,” http://xmlns.com/foaf/spec/, November 2007. [Online]. Available: http://xmlns.com/foaf/spec/

[42] J. Camenisch and J. Groth, “Group signatures: Better efficiency and new theoretical aspects,” in *In proceedings of SCN 04, LNCS series.* Springer, 2005, pp. 120–133.

[43] K. Cameron, “The laws of identity,” Web site: [Last accessed: 18/10/2011], May 2005. [Online]. Available: http://www.identityblog.com/stories/2004/12/09/thelaws.html

[44] D. W. Chadwick, *Understanding X.500 - the directory.* Chapman and Hall, 1994.

[45] D. Chaum and E. van Heyst, “Group signatures,” in *Advances in Cryptology EUROCRYPT ’91*, D. Davies, Ed. Springer-Verlag, 1991, pp. 257–265.

[46] D. Chaum, “Security without identification: transaction systems to make big brother obsolete,” *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[47] I. Clarke and S. D. C. Mellish, “A distributed decentralised information storage and retrieval system,” University of Edinburgh, Tech. Rep., 1999.

[48] F. C. Commission, “In the matter of telephone number portability - first report and order and further notice of proposed rulemaking,” July 1996.

[49] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “RFC5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” 2008.

[50] R. Cox, A. Muthitacharoen, and R. Morris, “Serving DNS Using a Peer-to-Peer Lookup Service,” in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS ’01. London, UK: Springer-Verlag, 2002, pp. 155–165. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687812

[51] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles.* New York, NY, USA: ACM, 2001, pp. 202–215.

[52] P. B. Danzig, K. Obraczka, and A. Kumar, "An analysis of wide-area name server traffic: a study of the internet domain name system," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 4, pp. 281–292, 1992.

[53] T. Deegan, J. Crowcroft, and A. Warfield, "The main name system: an exercise in centralized computing," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 5–14, 2005.

[54] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC '87. New York, NY, USA: ACM, 1987, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/41840.41841

[55] T. Dierks and E. Rescorla, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," Tech. Rep., Aug. 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246

[56] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976. [Online]. Available: citeseer.ist.psu.edu/diffie76new.html

[57] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Des. Codes Cryptography*, vol. 2, pp. 107–125, June 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00124891

[58] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," in *Proceedings of the Third International Workshop on*

Fast Software Encryption. London, UK: Springer-Verlag, 1996, pp. 71–82. [Online]. Available: http://dl.acm.org/citation.cfm?id=647931.740583

[59] J. R. Douceur, “The Sybil Attack,” in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS ’01. London, UK: Springer-Verlag, 2002, pp. 251–260. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687813

[60] D. Doval, “Self-organizing resource location and discovery,” Ph.D. dissertation, University of Dublin, Trinity College, 2003. [Online]. Available: http://www.dynamicobjects.com/manifold/archives/thesis.pdf

[61] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, “SPKI Certificate Theory,” September 1999. [Online]. Available: http://tools.ietf.org/html/rfc2693

[62] C. Ellison and C. Inc, “Establishing identity without certification authorities,” in *6th USENIX Security Symposium*, 1996, pp. 67–76.

[63] European Telecommunications Standards Institute, “ETSI EG 202 072 V1.1.1 (2002-09),” 2002.

[64] I. Fette and A. Melnikov, “The WebSocket protocol,” draft-ietf-hybi-thewebsocketprotocol-17, Internet Engineering Task Force, Sept. 2011. [Online]. Available: http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17

[65] B. Ford, “UIA: A global connectivity architecture for mobile personal devices,” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, Sept. 2008. [Online]. Available: http://www.brynosaurus.com/pub/net/phd-abs.html

[66] B. Ford, J. Strauss, C. Lesniewski-laas, S. Rhea, F. Kaashoek, and R. Morris, “Persistent personal names for globally connected mobile devices,” in *In Proc. of OSDI 2006*, 2006.

[67] M. Foster, T. McGarry, and J. Yu, "Number Portability in the Global Switched Telephone Network (GSTN): An Overview," RFC 3482, February 2003. [Online]. Available: http://tools.ietf.org/html/rfc3482

[68] A. O. Freier, P. Karlton, and P. C. Kocher, "The ssl protocol — version 3.0," Internet Draft, Transport Layer Security Working Group, November 1996.

[69] S. L. Garfinkel and R. C. Miller, "Johnny 2: a user test of key continuity management with s/mime and outlook express," in *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security.* New York, NY, USA: ACM, 2005, pp. 13–24.

[70] S. Gilbertson, "Six apart shuts down vox," Web site: [Last accessed: 18/10/2011]. [Online]. Available: http://www.webmonkey.com/2010/09/six-apart-shuts-down-vox/

[71] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," 2006. [Online]. Available: citeseer.ist.psu.edu/guha06experimental.html

[72] E. Hammer-Lahav, "RFC 5849: The OAuth 1.0 Protocol," 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5849.txt

[73] International Telecommunication Union, "The international public telecommunication numbering plan," ITU-T Recommendation E.164, May 1997.

[74] I. T. U. (ITU), "The Directory and Authentication Framework," nov 1988, reedition of CCITT Recommendation X.509 published in the Blue Book, Fascicle VIII.8 (1988). [Online]. Available: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.509-198811-S!!PDF-E&type=items

[75] J Laganier and L Eggert, "RFC 5204: Host Identity Protocol (HIP) Rendezvous Extension," 2008. [Online]. Available: http://tools.ietf.org/html/rfc5204

[76] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "DNS performance and the effectiveness of caching," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 589–603, 2002.

[77] B. Kaliski, "RFC 2898 - PKCS #5: Password-Based Cryptography Specification Version 2.0," September 2000. [Online]. Available: http://tools.ietf.org/html/rfc2898

[78] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: an empirical analysis of spam marketing conversion," in *CCS '08: Proceedings of the 15th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2008, pp. 3–14.

[79] M. Kassoff, C. Petrie, L. ming Zen, and M. Genesereth, "Semantic email addressing: Sending email to people, not strings," in *In: AAAI 2006 Fall Symposium on*, 2006.

[80] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321 (Draft Standard), Internet Engineering Task Force, October 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5321.txt

[81] Kohnfleder, "Towards a Practical Public Key Cryptosystem," Bachelor's thesis, MIT Department of Electrical Engineering, May 1978.

[82] B. W. Lampson, "Designing a global name service," in *PODC '86: Proceedings of the fifth annual ACM symposium on Principles of distributed computing.* New York, NY, USA: ACM, 1986, pp. 1–10.

[83] B. Laurie, "OpenID: Phishing Heaven," Web site: [Last accessed: 18/10/2011], Jan. 2007. [Online]. Available: http://www.links.org/?p=187

[84] S. Lind and P. Pfautz, "RFC5067: Infrastructure ENUM Requirements," RFC 5067 (Informational), Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5067.txt

[85] V. Lortz and M. Saaranen, "Deviceprotection:1 service for upnp version 1.0," Web site: [Last accessed: 18/10/2011], 2011. [Online]. Available: http://upnp.org/specs/gw/UPnP-gw-DeviceProtection-v1-Service.pdf

[86] P. Maniatis and M. Baker, "A historic name-trail service," in *Fifth IEEE Workshop on Mobile Computing Systems & Applications WMCSA*, 2003.

[87] P. Maniatis, M. Roussopoulos, E. Swierk, K. Lai, G. Appenzeller, X. Zhao, and M. Baker, "The mobile people architecture," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 36–42, 1999.

[88] S. P. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, University of Stirling, April 1994. [Online]. Available: http://homepage.mac.com/smarsh2003/SteveMarsh/Publications_files/Trust-thesis.pdf

[89] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, ser. IPTPS '01. London, UK: Springer-Verlag, 2002, pp. 53–65. [Online]. Available: http://dl.acm.org/citation.cfm?id=646334.687801

[90] M. Mealling and R. Daniel, "The Naming Authority Pointer (NAPTR) DNS Resource Record," RFC 2915 (Proposed Standard), Internet Engineering Task Force, September 2000, obsoleted by RFCs 3401, 3402, 3403, 3404. [Online]. Available: http://www.ietf.org/rfc/rfc2915.txt

[91] P. Mockapetris, *RFC 1034 Domain Names - Concepts and Facilities*, Internet Engineering Task Force, Nov. 1987. [Online]. Available: http://tools.ietf.org/html/rfc1034

[92] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 123–133, 1988.

[93] P. Mockapetris, "Domain names - concepts and facilities," RFC 882, November 1983. [Online]. Available: http://www.ietf.org/rfc/rfc882.txt

[94] ——, “Domain names: Implementation specification,” RFC 883, November 1983. [Online]. Available: http://www.ietf.org/rfc/rfc883.txt

[95] R. Moskowitz and P. Nikander, “Host identity protocol (hip) architecture,” May 2006. [Online]. Available: http://tools.ietf.org/html/rfc4423

[96] National Institute of Standards and Technology, “FIPS 180-2: Secure Hash Standard (SHS),” 2002.

[97] Philip Zimmerman, “PGP v2.x Manual Volume 1,” Web site: [Last accessed: 18/10/2011], 1994. [Online]. Available: http://www.cl.cam.ac.uk/PGP/pgpdoc2.ps.gz

[98] R Fielding and J Gettys and J Mogul and H Frystyk and L Masinter and P Leach and T Berners-Lee, “RFC2616: Hypertext Transfer Protocol – HTTP/1.1,” http://www.ietf.org/rfc/rfc2616.txt, 1999.

[99] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, T. Scavo, H. Lockhart, and B. Campbell, “Security Assertion Markup Language (SAML) V2.0 Technical Overview,” March 2008. [Online]. Available: http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[100] B. Raman, R. H. Katz, and A. D. Joseph, “Universal inbox: Providing extensible personal mobility and service mobility in an integrated communication network,” in *In Proc. of the Workshop on Mobile Computing Systems and Applications (WMSCAï£¡00*, 2000, pp. 95–106.

[101] V. Ramasubramanian and E. G. Sirer, “Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays,” in *NSDI’04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation.* Berkeley, CA, USA: USENIX Association, 2004, p. 8. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251183

[102] ——, “The design and implementation of a next generation name service for the internet,” in *SIGCOMM ’04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2004, pp. 331–342.

[103] P. Resnick, “RFC 5322 - Internet Message Format,” Online, 2008. [Online]. Available: http://tools.ietf.org/rfc/rfc5322.txt

[104] R. Rivest, “Can we eliminate certificate revocation lists?” in *In Financial Cryptography.* Springer-Verlag, 1998, pp. 178–183.

[105] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs),” RFC 3489, March 2003.

[106] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Lecture Notes in Computer Science*, 2001, pp. 329–350.

[107] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-to-end arguments in system design,” *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.

[108] A. Singla and C. Rohrs, “Ultrapeers: Another step towards gnutella scalability,” Web site: [Last accessed: 18/10/2011], December 2001. [Online]. Available: http://rfc-gnutella.sourceforge.net/Proposals/Ultrapeer/Ultrapeers.htm

[109] A. Sotirov, “Creating a rogue CA certificate.” [Online]. Available: http://phreedom.org/research/rogue-ca/

[110] P. Srisuresh and M. Holdrege, “RFC2663: IP Network Address Translator (NAT) Terminology and Considerations,” August 1999. [Online]. Available: http://tools.ietf.org/html/rfc2663

[111] M. Stiegler, "An introduction to petname systems," Web site: [Last accessed: 18/10/2011], February 2005. [Online]. Available: http://www.skyhunter.com/marcs/petnames/IntroPetNames.html

[112] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[113] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *In Proceedings of ACM SIGCOMM*, 2002, pp. 73–86.

[114] J. Touch, D. Black, and Y. Wang, "Problem and Applicability Statement for Better-Than-Nothing Security (BTNS)," RFC 5387 (Informational), Internet Engineering Task Force, November 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5387.txt

[115] T. I. T. Union, "First Mobile Phones, Now Mobile Numbers ITU allocates code for Universal Personal Telecommunications Number," Web site: [Last accessed: 18/10/2011], December 2001. [Online]. Available: http://www.itu.int/newsroom/press_releases/2001/31.html

[116] P. Vixie, "Extension Mechanisms for DNS (EDNS0)," RFC 2671 (Proposed Standard), Internet Engineering Task Force, Aug. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2671.txt

[117] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)," RFC 2136 (Proposed Standard), Internet Engineering Task Force, April 1997, updated by RFCs 3007, 4035, 4033, 4034. [Online]. Available: http://www.ietf.org/rfc/rfc2136.txt

[118] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," http://www.w3.org/TR/soap12-part1/.

[119] M. Walfish, H. Balakrishnan, and S. Shenker, “Untangling the web from dns,” in *NSDI’04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation.* Berkeley, CA, USA: USENIX Association, 2004, pp. 17–17.

[120] B. Wellington, “Secure Domain Name System (DNS) Dynamic Update,” RFC 3007 (Proposed Standard), Internet Engineering Task Force, November 2000, updated by RFCs 4033, 4034, 4035. [Online]. Available: http://www.ietf.org/rfc/rfc3007.txt

[121] W. H. Winsborough and N. Li, “Towards practical automated trust negotiation,” in *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002).* IEEE Computer Society Press, June 2002, pp. 92–103.

[122] T. Wu, “RFC 2945: The SRP Authentication and Key Exchange System,” Sept. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2945.txt

[123] T. Ylonen and C. Lonvick, “The Secure Shell (SSH) Authentication Protocol,” RFC 4252 (Proposed Standard), Internet Engineering Task Force, January 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4252.txt

[124] K. Zeilenga, “Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map,” RFC 4510 (Proposed Standard), Internet Engineering Task Force, June 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4510.txt