IdentiScape: Tackling the Personal Online Identity Crisis

Petros Maniatis Mary Baker
{maniatis,mgbaker}@cs.stanford.edu
http://identiscape.stanford.edu/

Technical Report: CSL-TR-00-804
June 2000

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-9040

Abstract

Traditional systems refer to a mobile person using the name or address of that person's communication device. As personal communications become more diverse and popular, this solution is no longer adequate, since mobile people frequently move between different devices and use different communications applications. This lack of identifiers for mobile people causes problems ranging from the inconvenient to the downright dangerous: to locate a person, callers must use potentially multiple email addresses, cell phone numbers, land line phone numbers or instant messaging IDs; callers leave sensitive messages on shared voicemail boxes; and they send communications intended for the previous owner of a telephone number to the next owner. To solve this naming problem, we should be able to name *people* as the ultimate endpoints of personal communications, regardless of the applications or devices they use.

In this paper, we develop a naming scheme for mobile people: we derive its requirements and describe its design and implementation in the context of personal communications. IdentiScape, our prototype personal naming scheme, includes a name service which provides globally available identifiers that persist over time and an online identity repository service which can be locally owned and managed.

Keywords

Personal Mobility, Personal Naming, Personal Online Identity.

I. Introduction

While current systems traditionally target the communication device as the mobile endpoint, they do little to support the mobile person as the ultimate endpoint of communications. For instance, though current systems can consistently address a host as it moves across different networks [Per96], they offer no consistent way to address a person who moves across different applications and communication devices.

This lack of naming support for mobile people is exacerbated by the increasing diversity, popularity and availability of personal telecommunications. The multitude of methods we use to identify and authenticate ourselves to different communication applications and the multitude of ways in which we can reach others have created an enormous management problem, which inhibits personal mobility instead of promoting it.

First, it is increasingly difficult and time consuming to keep track of all of our names and authentication mechanisms in the networks we use, especially since we often need to identify ourselves differently according to the type of communication in which we wish to participate. For instance, Jane Mobile may have a different user name and password for her office email system, home email, office voice mail, home voice mail, cell phone voice mail and instant messaging system, all of which she uses regularly in different situations.

Second, it is time consuming to ensure that those who wish to reach us online have our current, most appropriate contact information. To be reachable easily, Jane Mobile needs to make sure we have all of her phone numbers, email addresses and other online identifiers. The task is even harder if she wishes to disclose different subsets of her contact information to different people. For example, she might not want her business contacts to reach her at a hotel room during vacation, but she might want to be reachable there by her next of kin.

Third, contacting people reliably has become more inconvenient and error-prone than ever. The use of application-specific identifiers to identify people is often inconvenient when we need to reach people through a variety of applications and devices, but it sometimes even causes serious problems. Sensitive messages are left on shared voicemail boxes, communications intended for the previous owners of a telephone number are delivered to the next owner, and restricted information about a person's place of work is embedded in that person's communication address.

This management problem has traditionally been attacked with sophisticated personal tracking mechanisms [JBK98], [RMS$^+$99], [uRe], [IMP], [eTu]. These mechanisms aim to maintain an up-to-date list of connectivity addresses where a mobile person can be reached at all times. The mobile person either updates this list manually, or is tracked automatically by arrays of interconnected devices. For example, Jane Mobile might manually update a web form, or be tracked automatically, every time she leaves her office to go chat with a coworker, so that phone calls directed at her are diverted from her office phone to the coworker's phone. All Dan Sender has to do, to be assured of reaching Jane directly, is to make use of Jane's tracking mechanisms, which he can access at a well-known network address, or through an invisible proxy listening in on Jane's communications.

The majority of such tracking solutions, though useful for minute-to-minute personal tracking, lack long-term perspective. They invariably assume that Dan Sender, in the example above, always has a way to name Jane, and that he can easily find her tracking mechanism.

Unfortunately, these assumptions are too optimistic. On one hand, people in the online world do not have unambiguous names. Dan cannot just lookup "Jane Mobile" and be sure to find the right person; even with a small fraction of the world population online, our first and last names are not unique identifiers.

On the other hand, application-, device-, or network-specific identifiers commonly used to identify people are transient and ambiguous. Email addresses change every time we change service providers or jobs. Telephone numbers change every time we change cellular carriers or residences. The `jane@example.com` that used to be Jane Mobile when Dan Sender contacted her in the past at her workplace, Example Inc., may not only be invalid, once Jane changes jobs, but may also now correspond to a different employee of Example Inc., who inherited Jane's email address.

We have termed this longer-term aspect of the personal mobility management problem the *personal online identity crisis*. Tackling this crisis is equivalent to naming the ultimate beneficiary of the personal telecommunications revolution, the mobile person.

IdentiScape, the project described in this paper, seeks to provide a solution for the personal online identity crisis. It complements the Mobile People Architecture [RMS+99], which encapsulates our short-term tracking mechanism. IdentiScape provides naming and directory services that

• allow callers to use one identifier to reach a person, regardless of the number of device and application endpoints associated with the recipient,
• provide persistence of identifiers over time to thwart identifier reuse, which causes confusion, and
• enable people to control the contents, fine-grained access rights, and storage of their own online identity information.

While the name service of IdentiScape is global, the identity information repositories it points to are locally managed, and they may reside anywhere for reasons of scalability, deployability and privacy.

The contributions presented in this paper include the identification of requirements for a personal online identity service; the design of IdentiScape, which fulfills these requirements as much as currently possible; a prototype implementation of the design; and a way to deploy the service without retrofitting current personal communications protocols.

The rest of the paper proceeds as follows: In Section II we explain in further detail the problems that arise from the Personal Online Identity Crisis and how these problems motivate our design requirements. In Section III we describe the design of our IdentiScape service prototype. In Section IV we explain fundamental beliefs behind the design of such a service, including the escapability of identities, and our beliefs that trust in a global name service is reasonable, that identity information repositories (in contrast) should be decentralized, and that encryption keys are unsuitable as identifiers for people. IdentiScape's implementation status is covered in Section V, and in Section VI we present related work. Finally, we conclude with future plans in Section VII.

II. The Personal Online Identity Crisis: Problems and Solutions

In this section we further describe the personal online identity crisis and explain the required features of an ideal solution. These features fall into two main categories: naming requirements and identity object requirements. Naming deals with the names we use to refer to people online, while identity objects are the collections of information about the people named by those names.

A. The Problems

Because people are not bit-based, they traditionally delegate online representative entities, such as email accounts, telephones, instant messaging profiles and customer profiles to represent them online. People interact with and direct their online representatives through peripheral devices such as keyboards, microphones, speakers and telephone sets, which are interfaces between the physical and online worlds.

A problem with this representation of people in the online world is that people and their online representatives do not usually have an explicit association to each other. Problems arise both from the number of online representatives for a person and from changes in online representatives.

TABLE I
Naming Requirements

Property	**Description**
(II-B.1) Ubiquity	I can transfer my names over different media I have the same name everywhere My names don't give out private information
(II-B.2) Persistence	My names are valid as long as I want them to be I control what my old names point to The person behind a name is always the same
(II-B.3) Human-centricity	I can define my name I can change my name
(II-B.4) Robustness	My name is not similar to anybody else's It is easy to catch simple typos in a name

Jane Mobile may be hard to reach reliably if she has many phone numbers and email addresses, and even worse problems occur when her phone numbers and email addresses change. For example, the telephone number of a defunct pizza parlor could be reassigned to Jane Mobile who subsequently receives many unwanted pizza orders. A more dangerous situation could arise if sensitive communications intended for Jane Mobile from her lawyers are directed to a phone number or email address she used to share with the party she is now suing.

B. Naming requirements

From the above problems we observe a need for people to be able to choose online identities that more closely represent them than particular communication devices and applications. In addition, these online identities should belong to their owners for as long as desired, despite changes in residence, jobs, or affiliation. Besides these requirements for human-centricity and persistence of identifiers, we believe identifiers in an ideal name service should be ubiquitous and robust, as described below and summarized in Table I.

Requirement II-B.1: Names are ubiquitous

The range of uses for a single personal identifier is so broad and unpredictable that the survivability of a personal naming scheme is directly proportional to its adaptability to new uses. A name must be at least as versatile as current identifiers, such as email addresses, and it cannot closely depend on a particular medium. For instance, though highly unique, a palm print would be a bad personal identifier, since it is heavily biased towards an image-based medium; transmitting it over a human-understandable audio channel would be, at best, time consuming.

Given current mobility trends, we believe names should be globally valid. It should not be necessary to choose different identifiers when in different countries or when performing a lookup on the name from within different organizations. While personal address books may build a custom, private layer of indirection on top of a personal online identification service, there should be a simple, intuitive way for two strangers to refer to the same third person, regardless of their own private aliases for that person.

Finally, name ubiquity also applies to content. Names should be usable even in insecure environments; they should not contain sensitive information. Such information should be stored separately within the identity object, not piggy-backed on the identity name. Thus, today's email addresses and phone numbers may not be appropriate identity names, because they contain information about the network organization or locality where a person can be reached, and this may be sensitive information.

Requirement II-B.2: Names are persistent

Current identifiers used by people, both directory-based and unstructured, are in large part transient. Most email addresses are only as permanent as their associated Internet Service Provider account; telephone numbers in most of the world are dependent on the carrier and the geographic area where the subscriber lives; and X.500 distinguished names [Uni97a] are dependent on the named person's affiliation and geographical location. The use of public encryption keys as personal names [RL] suffers from the same problem, since people need to change public keys periodically to avoid brute-force or known plaintext attacks.

Name transience becomes even more daunting when it is combined with name reuse. Naming schemes like email and telephony allow an identifier to be reassigned after its association with its previous owner has ended. This results in adverse surprises such as the previous example of pizza orders being sent to Jane.

Transient, reusable names affect both senders and receivers of personal communications. Senders can effectively never be certain of the actual recipient of a personal communication they send. In practice, they assume that it is likely to reach a person at an address if they have been recently successful at reaching the same person at that address. Though this assumption may usually be true, it would be dangerous to rely on it, especially when security is important. Even secure (encrypted) communications are not immune to the effects of transient naming, since they, too, usually rely on mappings from a transient reusable service-specific name (e.g., a Versign ID), to a transient secret (e.g., a public key). Furthermore, when successive references are far apart in time, transient naming means that a sender must establish an association between the intended receiver and that receiver's name from scratch, using offline, or insecure online means.

In the inverse communications direction, transient, reusable naming means that a person may miss important communications from senders who have not been informed of a move, or worse have important communications sent mistakenly to the wrong, potentially malicious recipient. To protect against this contingency, diligent mobile people must inform all their important correspondents — frequent or not — of their name changes, and they must trust them to update all their local copies of this information. Job, service provider, and residence changes must all be propagated to potential correspondents. When both the intended receiver and sender of a communication have recently moved, tackling name mobility can be virtually impossible to handle simply.

For these reasons we believe that historical persistence of names is important: a name once created should live indefinitely and should only point to a single identity throughout its history, to the extent practically possible.

Requirement II-B.3: Person controls name

Any reasonable naming scheme for people should allow people to choose their own names. Randomly or bureaucratically generated, otherwise meaningless names, though just as functional for name resolution purposes as meaningful names, are inappropriate to name people: they are hard to memorize and hard to identify with psychologically.

A naming scheme should also allow people to change their names as desired, since people change, and their legal names change over time. For example, while Jane may have called herself "Lazy Lady" in the 60's, she might no longer find that name representative of her identity, and she should be able to change it.

Multiple names should be allowed to point to the same identity. As people choose new names without wishing to abandon old names entirely, they point their multiple names to the same online identity. Similarly, people must be allowed to control the obsolescence of names they own, for example to "disappear" online.

TABLE II
Identity Object Requirements

Property	**Description**
(II-C.1) Contents controlled by user	Users control the contents of their identities A person can define new types of identity data A person can share data types with others
(II-C.2) Fine access control	Access control can be tailored to every attribute
(II-C.3) Information store is trusted	Users should be comfortable trusting their identity store

Requiring that names be both changeable and persistent, as per the previous requirement, is not a contradiction. Persistence mandates that a name have an unambiguous referent throughout its lifetime. Changeability mandates that an identity be allowed to have different names throughout its lifetime. The two are orthogonal.

Requirement II-B.4: Names are robust in the face of human error

Since names must be usable by people, we must anticipate human error in their use. A good name space for people should be sufficiently sparse to avoid name confusion caused by typographic mistakes or an absent mind. For example, if the name "Jane01" exists, then "Jane10" should probably be disallowed as a new name.

C. Identity Object Requirements

A personal online identity object is a publicly available collection of information about an identity. We believe there are three main requirements for the structure of such a store: its contents should be controlled by the person or people it represents, and should admit fine-grained access control. In addition, people need to exact as much assurance for the safety of their identity information as they wish (See Table II for a summary).

Requirement II-C.1: Users determine what information to store

The information stored in an online identity should be exactly what its owner wishes to portray. The identity object itself should mandate neither completeness nor correctness of the information stored. For instance, Jane might wish to include only business-related information in her identity, perhaps maintaining a different identity for her personal interactions; she might wish to claim a different age than that indicated by her birth certificate; and she might want to talk about aspects of herself unanticipated in the design of the service.

The information store should not limit the kinds of information that may be contained therein. This implies the ability to define and even share new types of data representing such information. If Jane wishes to include information about her body piercings in her identity, and even share the data schema she defined for this with others, she should be able to do so. Sharing at this level promotes community building, since it allows groups with similar interests to standardize a language for talking about those interests, without necessarily having to rely on external administrative help, or lengthy standardization processes.

Finally, an identity need not necessarily portray a single, entire physical person. It could also be a human role undertaken by a changing set of people. For example, a company may wish to give its web master position an online identity independent of the actual people behind the position.

Requirement II-C.2: People can exercise fine-grain access control over their identity objects

People place varying amounts of importance on the privacy of different pieces of their identities. For example, cellular phone numbers might be more private than land line phone numbers in

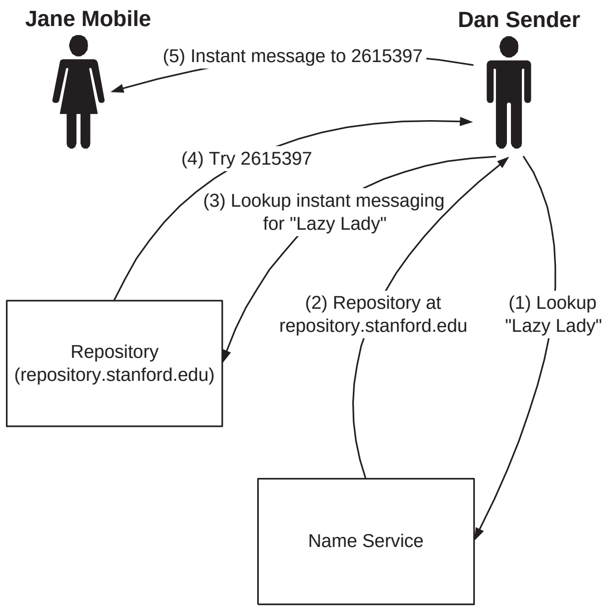
Jane Mobile
Dan Sender
(5) Instant message to 2615397
(4) Try 2615397
(3) Lookup instant messaging for "Lazy Lady"
(2) Repository at repository.stanford.edu
(1) Lookup "Lazy Lady"
Repository (repository.stanford.edu)
Name Service

Fig. 1. IdentiScape Design. We show the components participating in the identity service, and a typical, abstract protocol exchange between the service and a requester (Dan Sender), seeking the email address of Jane Mobile.

the US, since users pay for incoming cellular phone calls.

Identity objects should therefore offer their owners the ability to export views of their identity customized for specific people or groups of people. In Jane's case, she could export all of her business-related data only to her business acquaintances.

Requirement II-C.3: People should be assured of their privacy

An identity repository can keep unauthorized people from browsing a user's identity, but is the repository itself trustworthy? A person should be able to impose as strict security requirements on the network entity holding his identity object as desired, or as technically possible.

III. IdentiScape Design

In this section we describe the design of IdentiScape, a prototype personal online identity service. For a discussion of some of our design choices, please refer to Section IV.

To understand the fundamental functionality that IdentiScape provides, consider the following example, as illustrated in Figure 1. Dan Sender wants to send an instant message to his old friend Jane Mobile, whom he has not contacted in a while. With IdentiScape, all Dan has to do is input Jane's identity name into an instant messaging client and compose his message. No matter when or from where Dan performs this sequence of actions, the message will be sent where Jane wants it to arrive.

Ideally, Dan's instant messaging client must be able to find Jane's current instant messaging identifier, which is specific to the instant messaging protocol Dan is using. This is done in two phases in IdentiScape. First the client looks up Jane's identity name at a name service, perhaps authenticating Dan with the service first. The name service then returns a network pointer to an identity repository server that currently holds Jane's identity object if Dan is authorized to receive this information. Next, the client looks up Jane's identity name with that repository server, asking specifically for her instant messaging identifier. When that is returned, the client may go ahead with the transmission of Dan's message, using the regular instant messaging protocol it supports.

In the next few sections, we describe the components that make the above scenario work. In

Section III-A we present the infrastructure on which we rely for the design of IdentiScape, namely existing directory services and security services. In Section III-B we talk about Jane's identity name, what it looks like, and how it can be protected against human error. In Section III-C we describe the name service, its API and some preliminary insights on its deployment. In Section III-D we describe the repository, which actually holds the identity objects, and discuss its practical issues. In Section III-E we describe the identity objects themselves.

A. IdentiScape's infrastructure dependencies

In the design of IdentiScape we attempt to leverage currently specified designs as much as possible, and only make changes where we need additional support.

To design our identity repository and the structure of identity objects, we use the Lightweight Directory Access Protocol suite [WKH97], the de facto standard for directory services on the Internet. We chose LDAP, although it does not yet cover some of our required functionality, for three reasons. First, it is widely deployed and accepted. Second, solid implementations of LDAP exist, which we can use with some modifications. Third, the specification of the additional features we need are already in process, so we expect our needs and the functionality LDAP offers to converge soon.

For the security and privacy needs of IdentiScape, we choose to extend the mechanisms defined by the Simple Public Key Infrastructure [SPK], for their simplicity, broad applicability and existing implementation availability. We do not use the certificate and key distribution mechanisms of SPKI, because they rely on the concept of linked local name spaces, which are inappropriate for the reasons we present in Section IV-A. Furthermore, SPKI assumes that public keys are the primary principals in its mechanisms (see Section IV-D). We have extended SPKI to accommodate IdentiScape names.

B. The IdentiScape Name Space

Here we describe our choice of names that strikes a balance between the competing requirements of user control and robustness. We suspect that our prototype scheme may be naive, and certainly we expect further user feedback to cause us to evolve this design.

IdentiScape names (*identinames* for short) are based on free-form, case insensitive, Unicode [Con00] character strings, allowing numbers, letters in any supported script, and white space. An example of such a string is "Santas little helper". We have disallowed the use of any other punctuation for two reasons. First, to limit the similarity to commonly used application-specific identifiers such as email addresses, host names or telephone numbers, since this could cause ambiguity to end users ("is this an identiname or an email address?"). Second, to make sure that only linguistic means, which are more memorable to humans, are used to distinguish an identiname from others. Space provides a delimiter between components of the name. The components themselves can be picked arbitrarily; they do not have to belong to any human language.

The identiname consists of the basic string described above and an eight-bit checksum, appended to the basic string, and delimited with white space. The checksum is calculated on the UTF-8 [Yer98] encodings of the characters in the basic string, ignoring multiple instances of white space and counting all letters as lower-case. As a convenience to users of this scheme, to improve name memorability, we have defined a one-to-one mapping from an eight-bit number to a color or fruit name in English, which can be used interchangeably with the checksum. For example, the name derived from the basic string given above would be "Santas little helper blue" or, equivalently, "Santas little helper 216", since "blue" is the mnemonic for the number 216. This is a low-cost technique to limit casual name entry mistakes, without burdening name memorability unreasonably.

We limit the similarity between any two names in the name space, by requiring that no two names share more than half their words. Examples of basic strings we consider too similar to the name “Santas little helper” (minus the checksum) are “Santas helper little”, because it is a permutation of the original name, and “Santas little dog” which shares two words with the original name. Unfortunately, our heuristic can still be subverted by a determined adversary through exploitation of human error. Robust constructs guarding against human error in names, without limiting memorability, are an interesting and important topic for further investigation.

We expect the need for identinames to be on the order of two trillion identinames in the next 100 years, to avoid reuse. This figure is derived assuming instant global deployment, from estimated worldwide birth rates of about 150 million annually [Fun], and an expected use of at most 100 identinames per person.

The name space we propose can well accommodate this need without jeopardizing name memorability or robustness. To derive a rough sense of the size of the name space, consider a small subset of it, that which consists of any four distinct English words. If we only use nouns and verbs, which are roughly 20,000 and 14,000 respectively [SW89], we can construct approximately 1.4 trillion four-word names, observing the robustness limitations described above, as detailed in the Appendix. This measures a fraction only of the available name space, given that names of greater length, different languages and different formats are also allowed.

C. The IdentiScape Name Service

The IdentiScape name service is globally trusted and distributed. It maps an identiname to an identity repository that currently holds the identiname’s identity object. A mapping is protected with an access control list, which describes which other named identities may retrieve it. For example, if Jane only wants Dan to be able to find her identity object, she can insert Dan’s identiname in the ACL of her mapping at the name service. A mapping can be public, in which case anyone can access it. To enforce proper authorization, the name service provides commands to set up a secure, shared-key channel with an authenticated client.

With every identiname-to-repository mapping, the name service maintains a naming public key for the owner of the identiname, which carries the naming *authority* for that owner [EFL$^+$99]. A naming authority is something like a naming permission: a right to perform naming operations on behalf of its owner. This authority is created by the owner of an identity and delegated to the public/secret key pairs used to authenticate any operations at the name service. For example, Jane owns the right to operate on her name, that is, she owns her naming authority. Through a self-signed certificate, she delegates this authority to a public key pair that she generates. This is her master naming key pair, which Jane keeps out of circulation, and only produces when a certificate is challenged. She uses this key to sign the naming keys she uses regularly.

The name service is charged with the responsibility of maintaining the ownership of an identity throughout its history. It assumes that the naming authority is the closest bit-based entity to the owner of an identity. Therefore the name service keeps track of the public keys delegated with this authority to maintain ownership continuity. As naming public keys are replaced over time, because they expire or are compromised, the name service makes sure that the same naming authority is delegated on replacement naming keys, which is equivalent to maintaining the same ownership for a mapping. Unfortunately, if Jane gives her master naming key to Dan, she effectively passes ownership of her identity to Dan, as well. There is no technical means to prevent this from happening, as described in Section IV-C. In that respect, our design does not fulfill our persistent name requirement completely.

A new identiname for an identity object can be created by giving the name service a proposed new name, the location of the associated identity repository and the current naming public key of the owner’s naming authority.

Identity objects may move from repository to repository, when the owner changes identity service providers, for example. During an identity object migration, the name service must make sure that, at its new location, the identity object is controllable by the same naming authority. The API command for migration, in addition to the new repository location, contains a sequence number of migration requests. The sequence number is initialized when an identiname is created, and then incremented with every migration. This sequence number is used to thwart replay attacks.

The API provides lookup functionality in two flavors. First, a client can check whether an identiname or other identinames similar to it exist, without authentication. Existence requests do not return repository location information, since that is access controlled. They return either an affirmative or a negative answer, accompanied by any existing similar identinames. Second, a client can look up the repository location for an identity object, given its identiname. This lookup returns a transport address (IP address and port number), and the public naming key, if the operation is allowed by the access controls.

Finally, the naming API includes commands that deal with access control to the mappings (inspection or replacement). Only the owner of the mapping may use these commands.

Access to the name service through the Internet is straightforward through either a DNS lookup for `service.identiscape.com` or a service location search.

We believe that the size of the naming database will be manageable by storage systems of the near future. Assuming five kilobytes per name-to-repository mapping, and assuming 10 initial names per person, the storage needs for the current world population would be in the order of 30 terabytes, which can even now be handled by high-performance database servers.

We expect the rate of change of mappings within the name service also to be manageable. Assuming that world population remains under 20 billion for the next 100 years [Fun], and that each person uses at most 100 names during his lifetime of roughly 100 years, then we expect no more than 20 billion name changes or additions per year. Given that each name change will require the circulation of about 5000 bytes, we project the workload on the replicas of the name service to about three million bytes per second, which is well within the capabilities of current well-networked server environments, and should be certainly trivial before every man, woman and child on the planet uses IdentiScape.

D. The Identity Repository

The Identity Repository is a service responsible for maintaining identity objects and serving the information held within to authorized requesters. Identity repositories may also answer queries to unauthenticated requesters holding no special authority if the information in question is deemed public by its owner.

As mentioned in Section III-A, we use the Lightweight Directory Access Protocol functionality as the starting point for the identity repository. We describe below our modifications to the basic LDAP functionality, to suit our repository design.

To enforce proper authorization of repository clients, we use LDAP's bind and unbind primitives. They set up and tear down, respectively, a communication channel for a particular identity object, between a requester and the repository server, potentially authenticated with the requester's naming key. Note that a channel can only serve requests about a single identity object held at the repository.

We support LDAP's search and compare functionalities, with one restriction: only queries within a single identity object are allowed, even if the requester is authorized to look up information in other identity objects hosted in the same repository. The existence of an identity object at a specific repository should be revealed only by the name service, not by random wishful queries.

The LDAP modification primitive allows only the owner of an identity object to change its contents. LDAP's distinguished name modification primitive is replaced by a primitive to modify the primary identiname of an identity object.

For every identity object hosted, the identity repository holds the corresponding *hosting certificate*. This is a certificate signed with the owner's current secret naming key, authorizing the repository to hold the associated identity object. This is a delegation of the *hosting authority*. The repository can supply this certificate as proof of its right to host the named identity object.

Identity objects can be created and associated with a naming authority as described in Section III-C, or destroyed by the governing naming authority. Note that any naming authority can create an identity object within a repository, provided that any necessary service agreement with the repository maintainer is satisfied.

Since access control management within LDAP is still under discussion, we are implementing our own primitives, to view or change ACLs on attributes of an identity object. Only the owner of the object may access ACLs.

Ownership continuity is not enforced by the repository at any stage. This task is left to the name service. In this way it is possible to bypass the name service altogether, and use only the repository, if for example a person finds trusting the name service too burdensome. In that case, however, none of the benefits of name persistence and ubiquity can be reaped.

We use LDAP's mechanisms for scalability of large identity repositories. However, users might choose not to trust massively replicated identity repositories.

E. Personal Online Identity Objects

Our identity objects consist of an unordered set of named attribute/value pairs. The names of these pairs are unique within an identity object, but are otherwise unconstrained. Attribute/value pairs may be aggregated within named groups, which can be nested within one another to arbitrary depths.

Attribute types are selected from general or special-purpose attribute spaces, similar to name spaces in XML [Con99]. For example, Jane may be interested in an attribute space about body piercings containing attribute types for the diameter, material, color and location of jewelry. We have defined some initial attribute spaces for personal communication addressing, such as communications media available, communication addresses for every available medium, and basic identification information, such as first and last names, date of birth, etc. Anyone may extend the attribute spaces we have defined, or define new ones, and optionally publish them to be used by others. Given a link to its definition, an attribute space can be "mounted" inside an identity object, as a named group. Its specific attributes can then be populated with values specific to the owner of the identity object.

We do not use LDAP's support for data schema extensibility, since the definition of new arbitrary schemata is strongly discouraged in the specifications [WCHK97]. The specification favors packing encoded data extensions into textual fields, much like what MIME does for email. With this mechanism, entire sets of packed attributes and values have the access controls of the raw bit-field within which they have been packed. We find this restrictive, since it limits the amount of system-enforced access control a person has on the components of a non-standard data extension. We are currently addressing data schema extensibility using the XML name spaces mentioned above.

Values are non-empty, ordered lists of unstructured text. Binary data may be stored using MIME encoding facilities [FB96]. The repository makes no effort to parse, type-check or validate stored data, but values may be accompanied by validity certificates. A validity certificate provides a record of the attribute/value pair in question, along with the identiname of the object that contains it, cryptographically signed by an authority for that attribute. For example, Jane's identity object might contain an attribute/value pair for her social security number. To prove its validity, Jane may accompany her SSN with a certificate signed by the Social Security Authority, containing her identity name and her SSN.

Attribute/value pairs are associated with an access control list. The contents of an access control list are identinames of those allowed to read the values or detect the existence of the pair. Note that there is only one type of access right: read. Only the owner of an identity object may modify its contents, through the API described in Section III-D. By default, all attribute/value pairs are private. Pairs enclosed within a group inherit the access controls of the group, unless they have their own. No access control algebra takes place; an item's access is governed only by its own access control list, if one exists, or by the nearest enclosing group that has its own ACL.

An identity object always has a singleton value for its own, primary identiname. Note however that many identinames may point to the same identity object. This can be invaluable, for example, in cases where an identity has a Kanji name as well as a Latin character name for those unfamiliar with Kanji.

IV. Architectural Discussion

In this section we present the major decisions we had to make while deriving out requirements and turning them into the design of our prototype identity service.

A. Trust of Local versus Global Services

An important design decision is whether we should build components as network-wide, trusted services or whether we should instead limit the trusted service base to be near the end user. Our requirements for ubiquitous names are much more easily satisfied with a global service, but many people are skeptical of globally trusted distributed name services. For example, Ellison [Ell99] believes that in the area of secure certificate distribution protocols (which share many goals with our work), globally trusted servers are at odds with the reality of services run by disinterested third parties. The third party's employees may not be motivated enough to avoid clerical errors, or may be bribed, feeling protected behind their corporate employer. Instead, suggests Ellison, trust in local personal or business contacts is more justified, since the people we know well are likelier to promote our and their own security interests. It is through information provided by these people, a *web of trust*, that we should attempt to resolve lookups.

We sympathize with this view but believe that it does not apply to the service we describe, for three reasons. First, we believe that placing trust in a professional, contractually obligated entity, which can afford the most reliable and secure technology and practices, is preferable in most cases to relying on peers who may have a limited, lay-person's security expertise, whose local name database may be mobile, and who do not necessarily stake their livelihoods on their reputations for answering name lookups correctly and in a timely manner.

Second, despite theories similar to the "six degrees of separation" conjecture, it is highly unlikely for a person to be able to satisfy all identification needs through the locally trusted name spaces of friends and coworkers. White pages are not yet on the endangered species list.

Third, for the near future, the entire Internet depends on an immense, distributed, globally trusted yet insecure naming service: the Domain Name System. Until the standardized secure extensions to DNS have been widely deployed, any argument against global distributed services is moot.

For these reasons, we design our name resolution service as a global, hierarchical service that could be provided by multiple competing entities.

B. Location of Identity Objects

While we believe that naming information may satisfactorily be provided by a global service, this is not the case with the identity objects pointed to by the global name service. Personal online identities are intended to contain anything from a person's tax or medical records to special purpose private encryption or signing keys or payment instruments. Allowing a global, "impersonal" service to accumulate such a powerful dossier on a person would constitute an intolerable privacy transgression. Instead, we believe people should be able to store their identity information wherever they choose: perhaps on a home computer or at a trusted service provider.

In our design, the name service resolves a name to any online address at which the identity information may be stored. This gives users the most control over their own information.

C. Escaping Identities

Although historically persistent names help avoid many potential identification errors, they do not provide absolute inescapability of identities [JM98]. It is always ultimately possible for a person to give his identity to another person, although governments, legal systems and the use of biometrics can make it difficult. For example, if Dan Sender decides he wants Jane's identity, and if Jane agrees to this, then Dan and Jane may change the information in Jane's repository to represent Dan instead.

A community might choose to reduce the chances of such exchanges by requiring all users to include DNA samples in their identities, signed by appropriate authorities, and then use these samples to derive users' authentication tokens, for instance. While our system does not prevent a government from setting up such a scheme, any requirement to do so is the domain of law enforcement, politics, and nightmarish science fiction, not of the underlying technical infrastructure.

Thus we state that our design provides historical persistence of mappings of names to identities, *unless* the old and new owners of the identity agree to the change. This does not fully satisfy our name persistence requirement. As with any identity directory service in current use, it is ultimately a judgment call a user must make as to whether the information returned actually satisfies the request.

D. Names versus Public Keys

Another important issue in the design of a naming scheme for personal online identities is that of encryption key distribution. Since public keys produced by popular public key cryptography algorithms are mathematically guaranteed to be globally unique with near-absolute certainty, it would seem intuitive to use such public keys as names for people. After all, knowledge of a secret is as close as we can get to identifying (i.e., authenticating) a real person in the physical world. Separating the name from the authenticator may appear to be a step in the wrong direction.

We believe this reasoning does not apply to a naming scheme designed primarily for people. First, reasonably secure public keys tend to be in the range of a few thousand bits long. Even if a cryptographically secure hash function is applied to a public key — which increases the chances of name collisions — the result is still a long, meaningless string of digits to a human. We believe that a name that is not memorable will not survive as an identifier for people.

Second, public keys must change regularly, to deter brute-force, or known plaintext attacks. This means that if a public key were used as a person's name, that name would have to change regularly. Propagating regularly changed names to all those who need them is one of the fundamental problems we set out to solve with this work.

Third, not everyone is interested in privacy; there do exist people who take pride in having nothing to hide from anyone. Such people might not care even to generate a public/private encryption key pair, let alone be identified by it.

Unfortunately, rejecting the use of public keys as names carries its own cost. Most notably, a service that maintains the association between an identity name and its public encryption keys must be deployed and trusted. Since we already incur this cost to maintain name-to-repository mappings, this extra association is relatively inexpensive.

V. Implementation Status

For the IdentiScape prototype, we have implemented the components of the identity service, a client-side library and an identity browser. We have also extended our mobile personal communications platform to support identinames.

The prototype name server and the identity repository are currently monolithic, insecure and non-replicated. We have extended SPKI to support identinames as a first step in the implementation of the security features of IdentiScape. The programming environment is Java 1.2. We choose Java because it is convenient for rapid prototyping.

The client-side libraries for the name service and the repository are also Java-based, and currently support communication over Java's Remote Method Invocation (RMI) platform only. Using the libraries, we have built an identity browser, which can also be used by an identity owner to modify information held at the repository. To explore the adaptivity of current personal communication protocols to IdentiScape, we have built a simple email client that can handle email messages addressed with identinames. Email messages exchanged over the Simple Mail Transfer Protocol must have RFC 822-style addresses to be properly routed. The mailer extracts such addresses from IdentiScape. We encapsulate the identiname source and destination of a message in extended RFC 822 headers, so that IdentiScape-aware agents can use them.

To demonstrate identiname-based communication routing, we have extended the personal proxy, the prototypical implementation of the Mobile People Architecture [RMS$^+$99] to route communications based on sender and recipient identinames. In a simple scenario, Jane Mobile's personal proxy, when sending out a communication from her to Dan Sender, looks up his identiname and searches for an address of the applicable application type (e.g., email), to forward the communication. In the opposite direction, when Dan wants to send a communication to Jane, he can either use an IdentiScape-enabled client, such as the emailer described above, or first look up her identiname, which resolves to an application address. Jane has set up her identity object to contain the addresses of her personal proxy. The proxy itself knows where Jane is at a finer time grain, so it can filter and optionally reformat and forward the communication to her current network point of attachment.

VI. Related Work

The area of personal naming, identification and privacy has long been active. The project with the closest match in goals with ours is that undertaken by Specialist Task Force 157 of the European Telecommunications Standards Institute (ETSI), charged with finding "the Personal Identifier of the 21st Century" [PP00]. Though attacking the problem at a more abstract level, the task force is largely following a path parallel to ours.

In the rest of this section, we present related work which shares some of our goals.

A. Directory Services

In the Internet, by far the most popular, widely deployed directory is the Domain Name Service [Moc87]. Its original purpose was to serve mappings between human-readable machine names and IP network addresses. However, the service is currently being used for application-level routing (MX records), service location (SRV records), and public key distribution as well [DNS].

Although DNS is slowly moving to a more secure deployment, it is not the right service to offer personal identification. The names used, though ubiquitous, contain affiliation information, are not persistent, are minimally controllable by the named entity, and are notorious for their lack of robustness. Furthermore, DNS does not support user-initiated data schema extensibility or access control.

Among general-purpose directory services, X.500 [Uni97a] and the Lightweight Directory Access Protocol (LDAP) [WKH97] are the most representative examples. The X.500 standards suite describes an extremely versatile directory specification, covering issues from data schemata, entry organization, access control and extensibility, to authentication, replication, fault tolerance and caching. Its name space consists of a hierarchical sequence of attributes of the named person, including organizational affiliation, country of registration, etc. In that respect, X.500 names do not meet our naming requirements, since they are relatively transient and minimally controllable by their owner.

LDAP was proposed as a transitional, simpler protocol to access X.500 directories, than that defined by the X.500 suite. Its overwhelming success resulted in the specification of a standalone LDAP directory service, largely paralleling X.500 functionality. Unfortunately, standalone LDAP still does not include user-modifiable access controls, data schema extensibility, or reasonable replication. Most of the missing functionality is in the process of being specified and standardized, so we expect to use a variety of LDAP for our identity repositories, eventually.

B. Online Presence Tracking and Routing

IdentiScape targets the Personal Online Identity Crisis, which is the large-scale, coarse-grain aspect of the bigger, personal mobility management problem. At a finer grain, this problem is addressed by several presence tracking and person-level routing research or commercial projects. We combine IdentiScape with one of these, the Mobile People Architecture [RMS+99], to provide a more complete solution for personal mobility tracking.

The Iceberg [JBK98] and TOPS [AGK+99] projects differ from our approach in that they rely on deep infrastructure changes for their deployment. We limit ourselves to changes that can be deployed as much as possible on top of current infrastructure, instead.

Several commercial ventures [uRe], [eTu] attack the presence tracking problem by creating communities, within which contact information is centrally managed. Unlike IdentiScape, these do not permit tight user control on identity repositories.

The Internet standards community is currently in the process of specifying a large-scale presence tracking mechanism [IMP]. Data formats for the transfer of transient contact information have been defined in [Con]. Finally, the Sendmail mail server [Sen] already incorporates mechanisms for directory-based, email routing.

C. Security Infrastructure

The security field that shares the most goals with our work is that of *Public Key Infrastructure*. PKI addresses the problems facing distribution and maintenance of public keys, which, along with their private counterparts, are the main tools commonly used to ensure integrity and authenticity of communications. One of the tasks of PKI is that of associating a real-world actor, such as a person, a role, a group of people or a company, with a public/private key pair. Having a secure association of this sort, one can answer questions like "with which public key should I encrypt my message to ensure that only X can read it?" or "who has cryptographically signed this message?".

The oldest scheme available is based on the X.509 standard [Uni97b], which belongs to the X.500 suite. In its latest version (version 3 ratified in 1997), it allows both X.500 names and non-X.500 names to be used within certificates. Therefore, X.509v3 certificates can be used with any naming scheme, including identinames. Furthermore, X.509v3 inherits from X.500 its data schema extensibility and sharing. The IETF has worked on an Internet PKI based on X.509v3 certificates (PKIX) [PKI].

The Simple Public Key Infrastructure (SPKI) initiative [SPK] has taken a different approach. It considers a public key to be the closest online entity to a physical person. SPKI names are, in fact, public keys. The naming scheme used is borrowed from the Simple Distributed Secure Infrastructure (SDSI) project [RL] at MIT, and is based on the concept of linked local name spaces. According to this idea, people are likely to assign local names to the public keys of those with whom they communicate frequently. For example, Dan Sender contacts "jane", his local name for Jane Mobile, when he needs to reach her. If, however, he is interested in reaching Jane's dentist, whom Jane locally identifies as "dentist", Dan can use the construct "Jane's dentist". As long as he can securely access Jane's information online, Dan can resolve a local name chain to Jane's dentist's public key. For the reasons described in Section IV-A, we do not find this naming scheme appropriate for a personal online identity service.

Finally, the Pretty Good Privacy (PGP) project [Zim95] was specifically intended to secure email communications. The names within PGP certificates refer to email addresses, along with an alias, which might or might not be unique. Email addresses are under minimal user control, expose potentially sensitive information and may be short-lived. Consequently, we do not believe they constitute a sufficient global personal naming solution.

VII. Conclusion

This work addresses the growing identity management problem facing mobile people online. We describe a set of requirements we consider essential for a practical personal online identity system, including ubiquity and persistence for identity names, and user-controlled privacy and extensibility for identity objects. We further explain the design decisions we make regarding global trust, the separation of authentication from naming, and the escapability of currently practical online identification technologies. We then describe how we apply these requirements to the design of a prototype identity service, called IdentiScape.

We plan to evaluate our prototype over this next year by deploying the integrated system of the personal proxy and the identity service in our department for day-to-day communications. Through this deployment we hope to gain further insight into issues such as users' acceptance of our naming scheme, scalability of the name service, manageability of the repositories, and the practicality of our security infrastructure extensions. We also hope to derive quantitative utility metrics beyond our qualitative ones for evaluating personal identification systems.

VIII. Acknowledgments

This work has been supported in part by a generous gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo). Petros Maniatis is supported by a USENIX Scholar Fellowship.

We would like to thank Mema Roussopoulos for her help in extending the routing mechanisms in MPA to support IdentiScape, Keith Coleman and Jim Norris, for their work on the certificate infrastructure, and Sudipto Guha and Toni Hondroulis for their help with measuring the size of our name space. We are also grateful to Kevin Lai, Ed Swierk, Guido Appenzeller, Ichiro Okajima, Mike Pluke, Derek Pollard, Dan Boneh, Oliver Krone and Victoria Bellotti for useful discussions and their invaluable feedback to the ideas presented herein.

References

[AGK+99] N. Anerousis, R. Gopalakrishnan, C.R. Kalmanek, A.E. Kaplan, W.T. Marshall, P.P. Mishra, P.Z. Onufryk, K.K. Ramakrishnan, and C.J. Sreenan. TOPS: An Architecture for Telephony Over Packet Networks. *IEEE Journal of Selected Areas in Communications*, 17(1), January 1999.

[Con] Internet Mail Consortium. vCard - the electronic business card version 2.1. http://www.imc.org/pdi/vcard-21.txt,.

[Con99] World Wide Web Consortium. Recommendation: Namespaces in XML. http://www.w3.org/TR/REC-xml-names/, January 1999.

[Con00] The Unicode Consortium. *The Unicode Standard, Version 3.0.* Addison-Wesley, 2000.

[DNS] The IETF domain name system security (DNSSEC) working group charter. http://www.ietf.org/html.charters/ dnssec-charter.html.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. RFC 2693: SPKI certificate theory, September 1999.

[Ell99] C. Ellison. RFC 2692: SPKI requirements, September 1999.

[eTu] eTurn.com. Link your World. http://www.eturn.com/.

[FB96] N. Freed and N. Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies, November 1996.

[Fun] United Nations Population Fund. 6 billion: A time for choices. http://www.unfpa.org/swp/1999/contents.htm.

[IMP] The IETF instant messaging and presence protocol (IMPP) working group charter. http://www.ietf.org/html.charters/ impp-charter.html.

[JBK98] A. D. Joseph, B. R. Badrinath, and R. H. Katz. The Case for Services over Cascaded Networks. In *Proceedings of WOMMOM '98*, October 1998.

[JM98] Deborah G. Johnson and Keith Miller. Anonymity, pseudonymity, or inescapable identity on the net. In *Proceedings of the ethics and social impact component on Shaping policy in the information age*, pages 37–38, May 1998.

[Moc87] Paul V. Mockapetris. RFC 1034: Domain names — concepts and facilities, November 1987.

[Per96] C. Perkins. RFC 2002: IP mobility support, October 1996.

[PKI] The IETF public key infrastructure (X.509) (PKIX) working group charter. http://www.ietf.org/html.charters/ pkix-charter.html.

[PP00] Mike Pluke and Derek Pollard. Looking for the personal identifier of the 21st century. Personal communication, January 2000.

[RL] Ronald R. Rivest and Butler Lampson. A simple distributed security infrastructure (SDSI). http://www.toc.lcs.mit.edu/ cis/sdsi.html.

[RMS+99] Mema Roussopoulos, Petros Maniatis, Edward Swierk, Kevin Lai, Guido Appenzeller, and Mary Baker. Person-level routing in the Mobile People Architecture. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, pages 165–176, Boulder, CO, USA, October 1999. USENIX Association.

[Sen] The Sendmail Consortium. http://www.sendmail.org/.

[SPK] The IETF simple public key infrastructure (SPKI) working group charter. http://www.ietf.org/html.charters/ spki-charter.html.

[SW89] J. A. Simpson and Edmund S. Weiner, editors. *The Oxford English Dictionary.* Oxford University Press, 2nd edition, 1989.

[Uni97a] International Telecommunication Union. Recommendation X.500. Data Networks and Open System Communications – The Directory: Overview of concepts, models and services, 1997. Version 8/1997.

[Uni97b] International Telecommunication Union. Recommendation X.509. Data Networks and Open System Communications – The Directory: Authentication Framework, 1997. Version 8/1997.

[uRe] uReach.com. The All-In-One Communications Service. http://www.ureach.com/.

[WCHK97] M. Wahl, A. Coulbeck, T. Howes, and S. Kille. RFC 2252: Lightweight Directory Access Protocol (v3): Attribute syntax definitions, December 1997.

[WKH97] M. Wahl, S. Kille, and T. Howes. Lightweight Directory Access Protocol (v3). RFC 2251, 1997.

[Yer98] F. Yergeau. RFC 2279: UTF-8, a transformation format of ISO 10646, January 1998.

[Zim95] Philip R. Zimmermann. *PGP source code and internals.* MIT Press, 1995.

Appendix

I. Derivation of the size of the name space

The objective of this appendix is to prove the lower bound for the size of the name space described in Section III-B. To do that, we have to derive a lower bound for the name space structure described, namely, the set of all four-word combinations, where no word can be used more than once in the same combination, and no two combinations can share more than two words. Words can be chosen from the set of 34,000 English nouns and verbs.

Fig. 2. An algorithm for the four-generator

```
$Four(E)$
1. If $E$ has fewer than 4 elements, return the empty set
2. Partition $E$ into two halves, $E_{left}$ and $E_{right}$
3. Do $Four(E_{left})$
4. Do $Four(E_{right})$
5. Do $Two(E_{left}, E_{right})$
6. Return the union of all three sets
```

Fig. 3. An algorithm for the two-generator

```
$Two(E, D)$
1. If $E$ and $D$ have in total fewer than 4 elements, return the empty set
2. Partition $E$ into two halves, $E_{left}$ and $E_{right}$
3. Partition $D$ into two halves, $D_{left}$ and $D_{right}$
4. Do $Two(E_{left}, D_{left})$
5. Do $Two(E_{left}, D_{right})$
6. Do $Two(E_{right}, D_{left})$
7. Do $Two(E_{right}, D_{right})$
8. Do $One(E_{left}, E_{right}, D_{left}, D_{right})$
9. Return the union of all five sets
```

The appendix is organized as follows: First we describe an algorithm that generates combinations of the type described above; then we prove that the algorithm is sound, i.e., that it only generates correct combinations; then we derive the mathematical expression that yields the number of combinations generated by our algorithm, given the number of available words, and use this expression to obtain the number of combinations, i.e., a lower bound on the size of the name space we described above.

A. The algorithmic generator

In this section we describe a simple algorithm that can produce combinations of four distinct elements from a set E, such that no two combinations produced share more than two elements. We call combinations that share more than two elements *incompatible*.

The algorithm is built using three different combination generators:

• The *four-generator*, given a set of elements E, constructs a set $Four(E)$ of combinations of four distinct elements from E, such that no two combinations in $Four(E)$ are incompatible.
• The *two-generator*, given two disjoint sets E and D, constructs a set $Two(E, D)$ of combinations of four distinct elements, such that exactly two elements from each combination come from E, and exactly two elements come from D. As with $Four(E)$ above, no two combinations in $Two(E, D)$ are incompatible.
• The *one-generator*, given four disjoint sets E, D, C, B, constructs a set $One(E, D, C, B)$ of combinations of four distinct elements, such that exactly one element from each combination comes from each one of the sets E, D, C and B. Again, no two combinations in $One(E, D, C, B)$ are incompatible.

The three generators are described in pseudocode in Figures 2, 3 and 4 respectively.

The problem we need to solve is actually an instance of the four-generator, given the set of all available English nouns and verbs. For now, since there is no special handling in the pseudocode for element sets that cannot be split in two equally-sized halves, we assume that all sets have a cardinality that is a power of 2.

Fig. 4. An algorithm for the one-generator

```
$One(E, D, C, B)$
1. If $E$, $D$, $C$, and $B$ have in total 4 elements, return their elements into a
single combination.
2. Partition $E$ into two halves, $E_{left}$ and $E_{right}$
3. Partition $D$ into two halves, $D_{left}$ and $D_{right}$
4. Partition $C$ into two halves, $C_{left}$ and $C_{right}$
5. Partition $B$ into two halves, $B_{left}$ and $B_{right}$
6. Do $One(E_{left}, D_{left}, C_{left}, B_{left})$
7. Do $One(E_{left}, D_{left}, C_{right}, B_{right})$
8. Do $One(E_{left}, D_{right}, C_{left}, B_{right})$
9. Do $One(E_{left}, D_{right}, C_{right}, B_{left})$
10. Do $One(E_{right}, D_{left}, C_{left}, B_{right})$
11. Do $One(E_{right}, D_{left}, C_{right}, B_{left})$
12. Do $One(E_{right}, D_{right}, C_{left}, B_{left})$
13. Do $One(E_{right}, D_{right}, C_{right}, B_{right})$
14. Return the union of all eight sets
```

B. Soundness of the algorithmic generator

In this section we prove that the algorithm described above is sound, i.e., that the set of combinations it produces has no incompatible combinations. We will use induction to prove soundness.

The four-generator in Figure 2 constructs its result from the union of three sets: two recursive invocations of itself on smaller input and one invocation of the two-generator:

$$Four(E) = Four(E_{left}) \cup Four(E_{right}) \cup Two(E_{left}, E_{right}) \tag{1}$$

Theorem 1: The four-generator described in Figure 2 and Expression 1 is sound.

The base case for the four-generator occurs when E has exactly 4 elements; then both $Four(E_{left})$ and $Four(E_{right})$ yield the empty set, from line 1 in Figure 2. Therefore, in the base case, the four-generator degenerates into a single application of the two-generator. We will assume that the two-generator is sound for now — we will prove its soundness in Theorem 2. This means that the base case of the four-generator has no incompatible combinations.

Since the base-case application of the four-generator has no incompatibilities, we can form the inductive hypothesis: assume that neither of the sets $Four(E_{left})$ and $Four(E_{right})$ in expression 1 have incompatibilities; prove that $Four(E)$ has no incompatibilities.

Uniting $Four(E_{left})$ and $Four(E_{right})$ does not yield any incompatibilities: they draw their elements from disjoint sets, therefore no combinations in one will share any elements with any combinations in the other. Furthermore, no combination in $Two(E_{left}, E_{right})$ is incompatible with any combination in either $Four(E_{left})$ or $Four(E_{right})$, since any combination in $Two(E_{left}, E_{right})$ has exactly two elements from each partition of E, and to be incompatible with combinations from either of the four-generators, it would have to contain at least three elements from the same half. From this we conclude that, given the soundness of the two-generator, the inductive hypothesis holds, i.e., the four-generator is sound.

□

We proceed to prove the soundness of the two-generator. From Figure 3, the two generator constructs its result from the union of five sets: four recursive invocations of itself on smaller input and one invocation of the one-generator.

$$\begin{aligned} Two(E,D) \quad = \quad & Two(E_{left},D_{left}) \cup Two(E_{left},D_{right}) \cup \\ & Two(E_{right},D_{left}) \cup Two(E_{right},D_{right}) \cup \\ & One(E_{left},E_{right},D_{left},D_{right}) \end{aligned} \qquad (2)$$

We temporarily assume that the one-generator is sound; we will prove its soundness in Theorem 3 below.

Theorem 2: The two-generator described in Figure 3 and Expression 2 is sound.

The base case for the two-generator occurs when E and D have exactly two elements each; then all four of the two-generator invocations on the right side of expression 2 yield the empty set, from line 1 in Figure 3. Therefore, in the base case, the two-generator degenerates into a single application of the one-generator which has no incompatibilities.

We can form now the inductive hypothesis for the two-generator: assume that none of the sets $Two(E_{left},D_{left})$, $Two(E_{left},D_{right})$, $Two(E_{right},D_{left})$ and $Two(E_{right},D_{right})$ contains incompatible combinations; prove that $Two(E,D)$ contains no incompatible combinations.

None of the recursive invocations of the two-generator contains any combination that is incompatible with any combination in any of the other two-generator invocations. This is so, because no two-generator on the right side of expression 2 draws elements for its combinations from the same two sets. Since each two-generator draws exactly two elements from each of its input sets, there can be at most two elements in common between any combination of any two-generator in expression 2 with any combination of any other two-generator. Furthermore, no combination in $One(E_{left},E_{right},D_{left},D_{right})$ has a combination that is incompatible with any of those produced by the two-generators. This is so, because the one-generator has exactly one element from each partition of sets E and D. Since no two-generator invocation utilizes more than two partitions of sets E and D, it is guaranteed that any combination from the one-generator can have at most two elements in common with any of the combinations of the two-generators. Therefore, the inductive hypothesis holds, which means that the two-generator is sound, given that the one-generator is sound.

□

We prove now the final assumption that the one-generator is sound. The recursive step in Figure 4 can be rewritten as

$$\begin{aligned} One(E,D,C,B) \quad = \quad & One(E_{left},D_{left},C_{left},B_{left}) \cup \\ & One(E_{left},D_{left},C_{right},B_{right}) \cup \\ & One(E_{left},D_{right},C_{left},B_{right}) \cup \\ & One(E_{left},D_{right},C_{right},B_{left}) \cup \\ & One(E_{right},D_{left},C_{left},B_{right}) \cup \\ & One(E_{right},D_{left},C_{right},B_{left}) \cup \\ & One(E_{right},D_{right},C_{left},B_{left}) \cup \\ & One(E_{right},D_{right},C_{right},B_{right}) \end{aligned} \qquad (3)$$

Theorem 3: The one-generator described in Figure 4 and Expression 3 is sound.

The base case occurs when E, D, C and B have a single element each. From line 1 in Figure 4, this one-generator results in a single combination, one consisting of each one of the singular elements of all four input sets. Since there is only one combination in the set, the base case of the one-generator has no incompatibilities.

In the inductive hypothesis, we aim to prove that a one-generator produces no incompatibilities, if none of its recursive invocations produces any incompatibilities. None of the recursive invocations share more than two sets with any other recursive invocation; this means that no combination in any of the one-generators shares more than two elements with any combination of any of the other one-generators. Therefore, the union of all smaller one-generators has no incompatibilities, which means that the inductive hypothesis holds, and the one-generator is sound.

□

Although completeness of our generating algorithm is straightforward to prove, we omit the proof since it does not contribute to the measurement of our sample name space, in Section III-B.

C. The size of the generated space

Since we are interested in the actual number of elements in $Four(E)$, we will translate the set recurrences from Expressions 1, 2 and 3 into set-size recurrences. We define $\mathcal{FOUR}(n) = |Four(E)|$, when $|E| = n$, $\mathcal{TWO}(n) = |Two(E, D)|$, when $|E| + |D| = n$ and $\mathcal{ONE}(n) = |One(E, D, C, B)|$, when $|E| + |D| + |C| + |B| = n$.

$$\begin{aligned} \mathcal{FOUR}(n) &= 2 \times \mathcal{FOUR}(n/2) + \mathcal{TWO}(n) && (4) \\ \mathcal{TWO}(n) &= 4 \times \mathcal{TWO}(n/2) + \mathcal{ONE}(n) && (5) \\ \mathcal{ONE}(n) &= 8 \times \mathcal{ONE}(n/2) && (6) \end{aligned}$$

Recall that n is restricted to powers of 2 only.

The recurrence relations 4, 5 and 6 can be easily solved, from the bottom up, by using substitution. For example, in relation 6, we replace $\mathcal{ONE}$ on the right side with its definition successively, until we obtain the base case, i.e., $\mathcal{ONE}(4) = 1$. The resulting closed forms, in order of derivation are as follows.

$$\begin{aligned} \mathcal{ONE}(n) &= n^3/64 && (7) \\ \mathcal{TWO}(n) &= n^2(n-2)/32 && (8) \\ \mathcal{FOUR}(n) &= n^3/24 - n^2/8 + n/12 && (9) \end{aligned}$$

for $n = 2^k$, where k is a positive integer greater than or equal to 2.

The name space size we must measure in Section III-B would be $\mathcal{FOUR}(34000)$, if only we could apply the $\mathcal{FOUR}()$ function on numbers that are not powers of 2. To overcome this limitation, we can instead use the closest smaller power of 2 ($2^{15} = 32,768$) to derive a slightly smaller number of combinations for the name space. This number, $\mathcal{FOUR}(32768)$, describes the number of combinations we get if we ignore $34,000 - 32,768 = 1,232$ elements of the $34,000$ available. Since we are seeking a lower bound, this waste of valid combinations does not affect the validity of our estimate. The number we seek, then, is at least $\mathcal{FOUR}(32768) = 1,465,881,288,703$, or roughly 1.4 trillion.